An Interactive Broadcasting Protocol for Video-on-Demand

Jehan-François Pâris
Department of Computer Science
University of Houston
Houston, TX 77204-3475

paris@acm.org

Abstract

Broadcasting protocols reduce the cost of video-on-demand services by distributing more efficiently videos that are likely to be simultaneously watched by several viewers. Unfortunately, they do not allow the customer to pause, move fast forward or backward while watching a video.

We present an interactive pagoda broadcasting protocol that provides these functions at a very reasonable cost. Our protocol is based on the pagoda broadcasting protocol and requires a set-top box buffer large enough to keep in storage all video data until the customer has watched the entire video. As a result, rewind and pause interactions do not require any server intervention. To minimize the bandwidth requirements of fast forward interactions, the server only transmits the segments that are not available on any of the server broadcasting channels.

We evaluate the overhead of these fast forward operations through a probabilistic model. Our data indicate that the most costly fast forward operations are those starting at the beginning of the video and jumping to the beginning of the second half of the video while most fast-forward operation taking place during the second half of the video require little or no additional data.

1. Introduction

Despite all the attractiveness of the concept, video-on-demand (VOD) [16] has yet to succeed on the marketplace. The main reason for this lack of success is the high cost of providing video services. As a result, VOD cannot yet compete on a price basis with its two more entrenched rivals, namely pay-per-view and video-cassette rentals.

This situation has led to numerous proposals aiming at reducing the cost of providing video-on-demand (VOD) services. Many, if not most, of these proposals have focused on finding ways to distribute the top ten or twenty so-called "hot" videos more efficiently. Broadcasting protocols [3] distribute each video according to a fixed schedule that is not affected by the presence–or the absence–of requests for that video. Hence the number of viewers watching a given video does not affect their bandwidth requirements.

With the sole exception of staggered broadcasting, all broadcasting protocols share the common limitation of not offering any interactive action capability. They require the viewers to watch each video in sequence as in a theater. Unlike VCRs, they do not provide controls allowing the viewers to *pause* the video and interrupt its viewing, to move *fast forward* or backward (*rewind*).

While staggered broadcasting provides some interactive control capability, it only allows viewers to jump from one staggered stream to another [2]. The sole advantage of this solution is its simplicity. Its major disadvantages are its high bandwidth requirements and its lack of precision: given a video of duration D distributed over n broadcasting channels, staggered broadcasting only allows users to move forward or backward in increments of D/n times units. In addition, reducing the granularity of the jumps is very costly in terms of bandwidth.

We present a more flexible, much less expensive solution. With the sole exception of staggered broadcasting, all broadcasting protocols require a *set-top box* (STB) capable of

a) Simultaneously receiving video data from several channels and
b) Storing these data in its local buffer until the customer watches them.

The amount of data to be stored in the STB buffer varies with each distribution protocol but can be as high as 50 to 60 percent of the video duration. We propose to increase the STB buffer size in such a way that the STB will never have to discard any video data until the customer has watched the entire video. This would allow the STB to handle locally all *pause* and *rewind* commands. *Fast forward* interactions would still require the intervention of the video server and would be handled by a few *contingency streams* transmitting on demand the missing video data.

First Channel	S_1	S_1	S_1	S_1	S_1	S_1
Second Channel	S_2	S_4	S_2	S_5	S_2	S_4
Third Channel	S_3	S_6	S_8	S_3	S_7	S_9
Fourth Channel	repeats S_{10} to S_{14} and S_{20} to S_{29}					
Fifth Channel	repeats S_{15} to S_{19} and S_{30} to S_{49}					
Sixth Channel	repeats S_{50} to S_{99}					

Figure 1: How pagoda broadcasting maps 99 segments into six channels.

To investigate the feasibility of our approach, we have evaluated the bandwidth overhead caused by adding interactive controls to an existing proactive video distribution protocol, namely, the pagoda broadcasting protocol [13]. We found that the most costly fast forward operations were those starting at the beginning of the video and jumping to the beginning of the second half of the video. Even in this case, transmitting only the missing video data reduces the cost of the operation by 50 percent.

The remainder of our paper is organized as follows. Section 2 reviews previous work. Section 3 introduces our *interactive pagoda broadcasting protocol* and Section 4 contains our analysis of the protocol performance. Finally, Section 5 has our conclusions.

2. Previous Work

The simplest video broadcasting protocol is *staggered broadcasting* [16]. A video broadcast under that protocol is continuously retransmitted over k distinct video channels at equal time intervals. The approach does not necessitate any significant modification to the set-top box (STB) but requires a fairly large number of channels per video to achieve a reasonable waiting time. Consider, for instance, a video that lasts two hours, which happens to be close to the average duration of a feature movie. Guaranteeing a maximum waiting time of five minutes would require starting a new instance of the video every five minutes for a total of 24 full-bandwidth streams.

The past five years have seen the development of many more efficient broadcasting protocols [3]. Most of these protocols assume that the client set-top box has enough local storage to store at least one half of each video being watched. We can subdivide these protocols into two groups. The protocols in the first group are based on Viswanathan and Imielinski's *pyramid broadcasting* protocol [15]. They include Aggarwal, Wolf and Yu's *permutation-based pyramid broadcasting* protocol [1], Hua and Sheu's *skyscraper broadcasting* protocol [5] and Juhn and Tseng's *fast broadcasting* protocol [7].

While these protocols require less than half the bandwidth of staggered broadcasting to guarantee the same maximum waiting time, they cannot match the performance of the protocols based on the *harmonic broadcasting* (HB) protocol [6, 12]. These protocols divide videos into many equally sized segments and allocate a separate data stream to each segment. In addition, they require the STB to start receiving all these streams when the customer starts watching the first segment of a video. As a result, each segment can be broadcast using the minimum bandwidth required to ensure on time delivery of its data

Even though the total bandwidth requirements for HB and its variants are quite small, the multitude of streams these protocols involve complicates the task of the STB's and the servers. Like HB, *pagoda broadcasting* (PB) [13] uses fixed-size segments. It assigns to each video m video channels whose bandwidths are all equal to the video consumption rate and partitions these m channels into slots of equal duration. The protocol then tries to find the segment mapping that maximizes the number n of segments that can be packed into these m channels while satisfying the condition that segment S_k, for $1 \le k \le n$, is repeated at least once every k slots. Figure 1 shows how the PB protocol maps 99 segments into six channels. Since the segment size is then equal to 1/99 of the duration of the video, the maximum customer waiting time for a two-hour video is 7200/99 = 73 seconds.

Most research on interactive video-on-demand has focussed on distribution protocols that allocate their video streams in a dynamic fashion. Li *et al.* proposed in 1996 to use contingent streams to handle interactive VOD operations [9]. More recent work has focused on minimizing the duration of these contingent streams by merging them as soon as possible with other streams [10, 11, 8, 4]. Poon *et al.* have proposed a single-rate multicast double-rate unicast protocol supporting full VCR functionality [14].

3. The Interactive Pagoda Broadcasting Protocol

The *interactive pagoda broadcasting* (IPB) protocol is based on the assumption that most customers of a video-on-demand service will watch videos that they had not watched before. Hence these customers will use the

pause and *rewind* controls much more frequently than the *fast-forward* control.

Like the pagoda broadcasting protocol, the IPB protocol assigns to each video m video channels whose bandwidths are all equal to the video consumption rate. It then partitions these m channels into slots of equal duration and assigns to each slot one segment of the video in a way that guarantees that segment S_k, for $1 \le k \le n$, is repeated at least once every k slots.

As we mentioned earlier, most broadcasting protocols require a customer STB that can store over one half of each video being watched on their disk drive. Assuming a video distributed in MPEG-2 format with an average bandwidth of 5Megabits/s, this means that 2.25 Gigabytes of storage are needed to store the first hour of a two-hour video. The cheapest way to provide this buffer capacity is to add a disk drive to the STB. Most new disk drives being sold today have capacities of at least seven Gigabytes. A disk drive of that size would allow the STB to keep any previously played segment of the video being watched until the end of that video. As a result, the STB could handle all pause and rewind requests locally without any server intervention.

Implementing unrestricted fast forward interactions will require additional bandwidth unless we require each video to be completely downloaded by the customer STB before being viewed. This is not an acceptable solution as it would either result in unacceptable delays for the customer or require an inordinate amount of bandwidth. A better solution would be to allocate to each fast forward request a *contingency* video stream and use it to transmit to the customer STB the segments that it does not have and will not receive on time from the m broadcasting channels. One possible option would be to let these contingency streams use the extra server bandwidth that must be kept available to handle server disk failures.

The next question is to decide whether the server or the STB will determine which segments are to be included in the contingency stream. Making this task the responsibility of the server would require the server to keep track of the state of the storage units of all STBs currently receiving the video. This could be a daunting task for very popular videos and would complicate server recovery. Conversely, giving this responsibility to the STB would prevent the sharing of segments among contingency streams. We propose instead to share this duty between the STB and the server. After each fast-forward interaction, the STB will send to the server a request specifying the video segments it does not have and does not expect to receive on time. When the server receives the request, it checks if some of the requested segments are not already included in one of the existing contingency streams. Having done that, it verifies that it has sufficient bandwidth to satisfy the customer request. If this is the case, the server replies to the STB with a message informing it of the scheduled time slots and channel allocations of all the missing segments.

While our solution was strongly influenced by the work of Carter *et al.* [4], it differs in at least one major point. In order not to increase the size of the STB drive, these authors assume that the video server will handle all interactive commands. We believe our solution will require much less additional bandwidth without significantly increasing the cost of the STB.

4. Analytical Study

As we mentioned in the previous section, our IPB protocol lets the STB handle all pause and rewind operations without server intervention. As a result, only fast forward operation can affect the server workload. We will thus focus our analysis on the cost of fast forward operations. Unable to find any reliable data on the frequency and the extents of fast-forward operations, we decided instead to measure directly the average costs of fast forward operations skipping a given amount of video data starting at a given position within the video.

To ensure a steady flow of images, the IPB must ensure that the k^{th} segment of video, say segment S_k, will be broadcast at least once every k slots. As a first approximation, we can thus assume that there is at least a probability $1/k$ of finding segment S_k in any arbitrary slot. Consider now a viewer watching segment S_j and wanting to fast forward to some later segment S_k. The probability $p_{k,j}$ that either segment S_k is already in the STB buffer or can be received on time by the STB will satisfy the inequality

$$p_{k,j} \ge \frac{j+1}{k}.$$

Similarly the probability $p_{k+l,j+l}$ that segment S_{k+l} will either be already on the STB buffer or received on time by the STB will satisfy the inequality

$$p_{k+l,j+l} \ge \frac{j+l+1}{k+l}.$$

Not transmitting the segments that the client already has or will receive on time from the m broadcasting channels will reduce the average number of segments transmitted by the contingency stream by at least

$$\sum_{l=0}^{n-k} \frac{j+l+1}{k+l}$$

where n is the number of segments in the video. Hence, the average cost of a fast forward from within segment j to within segment k will never exceed

$$\sum_{l=0}^{n-k}(1-\frac{j+l+1}{k+l}) = \sum_{l=0}^{n-k}\frac{k-j-1}{k+l}$$
$$= \sum_{l=k}^{n}\frac{k-j-1}{l}$$
$$= (k-j-1)(H(n)-H(k-1))$$

where $H(n)$ is the n-th harmonic number.

In reality, mapping constraints force the protocol to broadcast some segments slightly more frequently than they should. As shown on Figure 1, segment S_5 is broadcast once every four slots even though it only has to be broadcast every five slots. Similarly, all segments in channel 6 are broadcast once every 30 slots. In general, segment S_k, will be broadcast once every s(k) slots with $s(k) \le k$. The probability $p_{k,j}$ that either segment S_k is already in the STB storage unit or it will be received on time by the STB is then given by

$$p_{k,j} = \min(\frac{j+1}{s(k)},1) .$$

Similarly the probability $p_{k+l,j+l}$ that segment S_{k+l} will either be already on the STB drive or received on time by the STB is given by

$$p_{k+l,j+l} = \min(\frac{j+l+1}{s(k+l)},1) .$$

Hence, the average cost of a fast forward from within segment j to within segment k will be equal to

$$\sum_{l=0}^{n-k}(1-\min(\frac{j+l+1}{s(k+l)},1)) = (n-k+1)-\sum_{l=0}^{n-k}\min(\frac{j+l+1}{s(k+l)},1)$$

where n is the number of segments in the video.

Define θ as the index of the lowest-numbered segment that is repeated at the same periodicity as the last segment of the video S_n:

$$\theta = \min\{k \mid s(k) = s(n)\} .$$

Since $s(k)$ is normally a monotonic non-decreasing function of k, we will have $s(k)=s(n)$ for all $\theta \le k \le n$. Since $s(\theta) \le \theta$, this means that $s(k) \le \theta$ for all $\theta \le k \le n$.

Consider now a fast-forward interaction starting from a segment S_j such that $j \ge \theta$. At that time, the customer STB would have already received *all* segments of the video since none of them would have been broadcast less frequently than once every θ slots. Therefore it would not require additional video data from the video server and would not result in any bandwidth overhead.

Theorem 1: *Any fast-forward interaction starting from a segment S_j such that $j \ge \theta$ would not cause any bandwidth overhead.*

Consider, for instance, the case of the IPB protocol with six channels. Segments S_{50} to S_{99} are all broadcast once every 50 slots, which means that $\theta = 50$. Hence any fast-forward operation taking place during the second half of the video will be handled by the STB without any server intervention. Similarly IPB with five channels broadcasts segments S_{30} to S_{49} once every 30 slots. Any fast forward operation taking place during the last 40 percent of the video would also be handled without any server intervention.

Figure 2 shows the influence of the number of skipped segments on the cost of a fast forward operation for the IPB protocol with 5 channels and 49 segments. Each segment thus represents 2 minutes and 27 seconds of playtime for a two-hour video. The X-axis represents the number of skipped segments during the fast forward operation while the Y-axis represents the number of additional video segments sent by the video server as a result of the fast-forward operation. Each curve corresponds to a different starting point.

As we can see, the most expensive fast forward operations start while the customer is watching the first segment of the video and skip 23 segments thus bringing the viewer to the beginning of segment S_{25}, that is, just before the beginning of the second half of the video. The contingency stream that ensures on time delivery of segments S_{25} to S_{49} has an average duration of 13 segments, that is, 31 minutes and 50 seconds of data for a two-hour video.

Even here, the protocol achieves significant bandwidth savings by excluding from the contingency stream the segments that the STB has already received or can receive on time from the five broadcast channels. A protocol that would always transmit segments S_{25} to S_{49} would have sent 26 segments, that is, exactly double of what our protocol requires.

Fast forward operations taking place later in the video require considerably less bandwidth, mostly because the STB is more likely to have already received the segments or to be able to receive them on time from the five broadcasting channels. For instance, fast forward operations starting while customers watch the 16th segment of the video never require more than an average of 3.0333 segments, that is, 7 minutes and 26 seconds of data for a two-hour video. Fast forward operations starting while customers watch the 24th segment of the video are almost free. All fast forward operations starting after that require no server intervention.

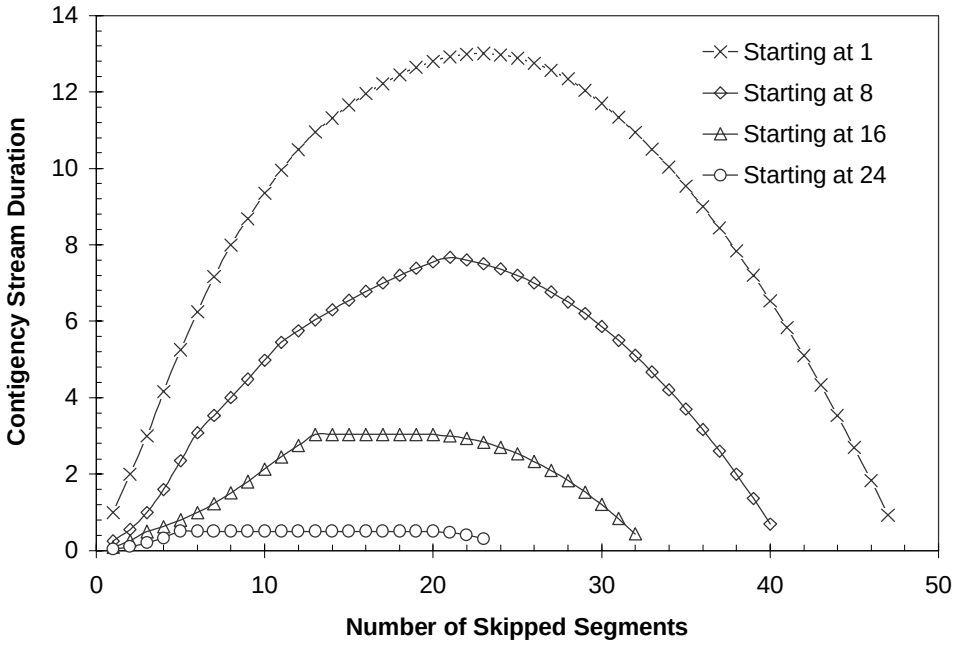

Figure 2: Cost of a fast forward as a function of its starting point and the number of segments skipped for IPB with 5 channels and 49 segments.

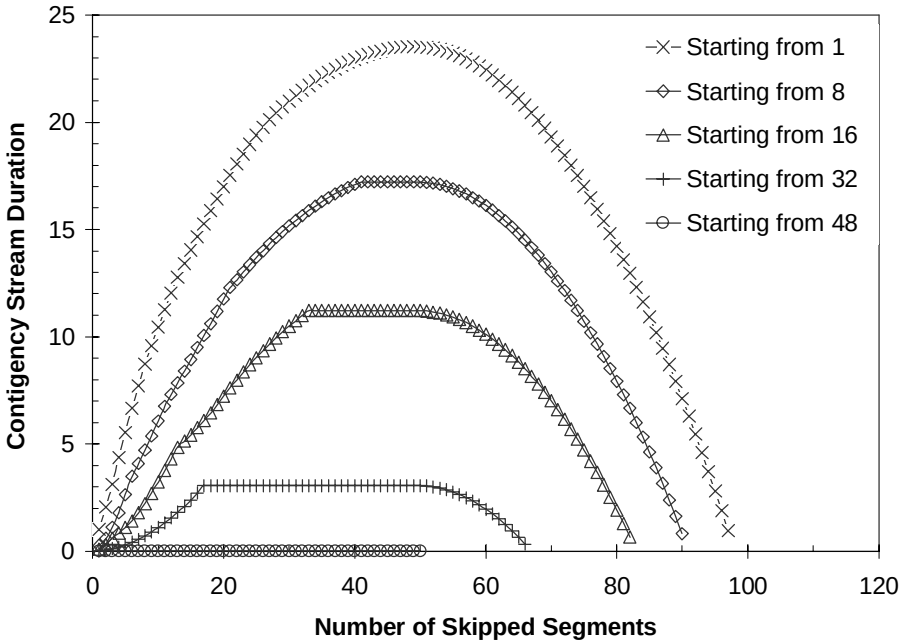

Figure 3: Cost of a fast forward as a function of its starting point and the number of segments skipped for IPB with 6 channels and 99 segments.

As Figure 3 indicates, similar conclusions hold for the IPB protocol with 6 channels and 99 segments. The most expensive fast forward operations start while the customer is watching the first segment of the video and skip 49 segments thus bringing the viewer to the beginning of segment S_{51}, that is, slightly after the beginning of the second half of the video. The contingency stream that ensures on time delivery of segments S_{51} to S_{99} has an average duration of 23.52 segments, that is, 28 minutes and 31 seconds of data for a two-hour video. This is 48 percent of the 49 segments that a protocol that always transmits segments S_{51} to S_{99} would have required. Fast forward operations starting while customers watch the 48^{th} segment of the video are almost free. All fast forward operations starting after the 49^{th} segment of the video require no server intervention.

We need however to point out that these results only hold for fast forward operations that are not preceded by any other interactive request. Fast forward operations taking place after one or more pause or rewind requests will require less bandwidth as the STB will have accumulated more segments in its buffer. On the other hand, cascading fast forward operations, especially when a customer performs incremental fast-forwards throughout an entire video, will require considerably more bandwidth.

Our model did not either consider how segment sharing among contingency streams could reduce the average number of segments per contingency stream. We do not believe this impact to be significant as long as fast forward operations remain infrequent.

5. Conclusion

Rather than answering individual requests, broadcasting protocols distribute each video according to a fixed schedule that is not affected by the presence—or the absence—of requests for that video. As a result, they do not provide controls allowing the viewers to *pause* the video and interrupt its viewing, to move *fast forward* or backward (*rewind*). The sole exception is staggered broadcasting, which allows viewers to jump from one staggered stream to another [2]. The biggest disadvantage of this solution is its high cost..

We have presented here an interactive broadcasting protocol that allows the customer greater control at a much lower cost. Our *interactive pagoda broadcasting* protocol (IPB) is based on the pagoda broadcasting protocol. Unlike the original pagoda broadcasting protocol, the IPB protocol requires a STB buffer large enough to keep in storage all video segments for the whole duration of the video. As a result, the STB can handle all *rewind* and *pause* interactions without any server intervention. To minimize the bandwidth requirements of *fast forward* interactions, the server only transmits the segments that are not already stored in the STB buffer and are not available on any of the VOD broadcasting channels.

We have evaluated the actual cost of these fast forward operations through a probabilistic model. We found that the most costly fast forward operations were those starting at the beginning of the video and jumping to the beginning of the second half of the video while most fast-forward operation taking place during the second half of the video required little or no additional data.

Keeping in mind that rewind and pause operations do not require any server intervention, we can safely conclude that the bandwidth requirements of our IBP protocol will remain reasonable as long as fast forward operations will remain infrequent.

Acknowledgments

This research was supported by the Texas Advanced Research Program under grant 003652-0124-1999 and the National Science Foundation under grant CCR-9988390.

References

[1] C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "A permutation-based pyramid broadcasting scheme for video-on-demand systems," In *Proceedings of International Conference on Multimedia Computing and Systems*, pages 118–126, Hiroshima, Japan, June 1996.

[2] K. C. Almeroth and M. H. Ammar, "The use of multicast delivery to provide a scalable and interactive video-on-demand service," *IEEE Journal on Selected Areas in Communications*, 14(50):1110–1122, Aug. 1996.

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

[4] S. W. Carter, D. D. E Long and J.-F. Pâris, "An efficient implementation of interactive video-on-demand," In *Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 172–179, San Francisco, CA, Aug.-Sept. 2000.

[5] K. A. Hua and S. Sheu, "Skyscraper broadcasting: a new broadcasting scheme for metropolitan video-on-demand systems," In *Proceedings of SIGCOMM 97Conference*, pages 89–100, Cannes, France, Sep. 1997.

[6] L. Juhn and L. Tseng, "Harmonic broadcasting protocols for video-on-demand service," *IEEE*

Transactions on Broadcasting, 43:268–271, Sep. 1997.

[7] L. Juhn. and L. Tseng, "Fast data broadcasting and receiving scheme for popular video service," *IEEE Transactions on Broadcasting*, 44(1):100–105, Mar. 1998.

[8] N. Kamiyama and V. O. K. Li, "An efficient deterministic bandwidth allocation method in interactive video-on-demand systems," In *Proceedings of the 1998 Global Communication Conference*, vol. 2, pages 664–671, Nov. 1998.

[9] V. O. K. Li , W. Liao , X. Qiu , and E. Wang, "Performance model of interactive video-on-demand systems,¨ *IEEE Journal on Selected Areas in Communications*, 14(6):1099–1109, Aug.. 1996.

[10] W. Liao and V. O. K. Li, "The split-and-merge (SAM) protocol for interactive video-on-demand systems," *IEEE Multimedia* 4(4): 51–62, 1997.

[11] W. Liao, V. O. K. Li: "Synchronization of distributed multimedia systems with user interactions," *ACM Multimedia Systems Journal*, 6(3): 196–205, 1998.

[12] J.-F. Pâris, S. W. Carter and D. D. E. Long, "Efficient broadcasting protocols for video on demand," In *Proceedings of the 6th International Symposium on Modeling, Analysis, and Simulation of Computing and Telecom Systems*, pages 127–132, Montreal, Canada, July 1998.

[13] J.-F. Pâris S. W. Carter, and D. E. Long, "A hybrid broadcasting protocol for video-on-demand," In *Proceedings of the 1999 Multimedia Computing and Networking Conference*, pages 317–326, San Jose, CA, Jan.. 1999.

[14] W.-F. Poon, K.-T. Lo and J. Feng, "Design and analysis of multicast delivery to provide VCR functionality in video-on-demand systems," In *Proceedings of the 2nd International Conference on ATM*, pages 132–139, June 1999.

[15] S. Viswanathan and T. Imielinski, "Metropolitan area video-on-demand service using pyramid broadcasting," *ACM Multimedia Systems Journal*, 4(4):197–208, Aug. 1996.

[16] J. W. Wong, "Broadcast delivery," *Proceedings of the IEEE*, 76(12), 1566–1577, Dec. 1988.