THE CASE FOR AGGRESSIVE PARTIAL PRELOADING IN BROADCASTING PROTOCOLS FOR VIDEO-ON-DEMAND

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

Darrell D. E. Long
Department of Computer Science
University of California
Santa Cruz, CA 95064

Abstract

Broadcasting protocols for video-on-demand usually consume over fifty percent of their bandwidth to distribute the first ten to fifteen minutes of the videos they distribute. Since all these protocols require the user set-top box to include a disk drive, we propose to use this drive to store the first five to twenty minutes of the ten to twenty most popular videos. This will provide low-cost instant access to these videos.

1. INTRODUCTION

An important factor in the high cost of video-on-demand services is the very high bandwidth these services require. Assuming that the videos are in MPEG-2 format, each user request will require the delivery of around 5 Megabits of data per second. Hence a video server allocating a separate data channel of data to each request would need an aggregate bandwidth of 5 Gigabit/second to accommodate 1,000 concurrent users.

This situation has resulted in many proposals aimed at reducing the bandwidth requirements of video-on-demand services. Despite all their differences, all these proposals are based on the same idea, namely, sharing as many data as possible among overlapping requests for the same video. Hence, most of these proposals assume that customers receive their videos through a set-top box (STB) capable of (a) simultaneously receiving data from several video channels and (b) storing in a local buffer the video data it receives out of sequence.

Unfortunately this approach does not work well for the first few minutes of each video because the customer STB has very little or no time to collect the required data. As a result, distributing the fist few minutes of a video takes a very large fraction of the total bandwidth required to distribute the video. Consider, for instance, the case of a video distributed through Juhn and Tseng's *fast broadcasting* protocol [4]. The fast broadcasting protocol requires a bandwidth equal to seven times the video consumption rate to guarantee a maximum waiting time of 57 seconds for a two-hour video. Fifty-seven percent of this bandwidth is used to distribute the first 15 minutes of the video, that is one eighth of the duration of the video. A similar observation would apply to a *pagoda broadcasting* protocol [5] operating under the same conditions.

A common characteristic of these two broadcasting protocols is that they require enough buffer space in the customer STB to store up to 60 percent of each video being watched. In the current state of the technology, this implies a disk drive in each STB. Most disk drives sold today can store at least 30 Gigabytes of data, that is, more than 13 hours of video data in MPEG-2 format. Thanks to this huge capacity, it become possible to preload in the customer STB the first ten to twenty minutes of the most popular videos. We could, for instance, the first 10 minutes of the top 80 videos or the first 20 minutes of the top 40 videos. While the system would require a higher level of coordination between the customer STB and the video server, it would provide instant access to all the preloaded videos while reducing by more than 50 percent the bandwidth requirements of the protocol.

2. PREVIOUS WORK

Broadcasting protocols anticipate customer demand and distribute the various segments of each video according to a deterministic schedule. The *pyramid broadcasting* [8] protocol was the first broadcasting protocol that required the customer STB to include enough buffer space to store up to one half of each video being broadcast. This allowed the STB to receive the video data out of order and allowed the video server to transmit less frequently the later portions of each video. Among the other broadcasting protocols following the same approach, we should mention *skyscraper broadcasting* [2], *harmonic broadcasting* [3], *fast broadcasting* [4], and *pagoda broadcasting* [5].

One of the most intuitive broadcasting protocols is Juhn and Tseng's *fast broadcasting* (FB) protocol [4]. FB allocates to each video k channels whose bandwidths are all equal to the video consumption rate b. It then partitions each video into $2^k - 1$ segments S_1 to S_{2^k-1} of equal duration d. As Figure 1 indicates, the first channel continuously rebroadcasts segment S_1, the second channel transmits segments S_2 and S_3, and the third channel transmits segments S_4 to S_7. More generally, channel j with $1 \le j \le k$ transmits segments $S_{2^{j-1}}$ to S_{2^j-1}.

Pagoda broadcasting (PB) [5] improves upon the FB protocol by allocating segments to pairs of consecutive channels, which allows packing more segments into the same number of channels. As shown on Figure 2, a PB protocol using three channels would pack nine segments

First Channel	S_1	S_1	S_1	S_1
Second Channel	S_2	S_3	S_2	S_3
Third Channel	S_4	S_5	S_6	S_7

Figure 1. Fast broadcasting with three channels.

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

Figure 2. Pagoda broadcasting with three channels.

into these three channels, that is two segments more than FB. Hence the maximum waiting time for a video of duration D is $D/9$ instead of $D/7$. More generally, the maximum waiting time for a video of duration D broadcast over n channels is given by $d = D/[2\times5^{((n-1)/2)} - 1]$ for n odd, and $d = D/[4 \times5^{((n-2)/2)} - 1]$ for n even.

Partial preloading [6] loads in the customer STB the first few minutes of the top 10 to 20 videos in order to provide zero-delay access to these videos and reduce somewhat the server bandwidth of the broadcasting protocol distributing the remainder of the video.

3. AGGRESSIVE PARTIAL PRELOADING

Over the last few years, disk drive capacities have been doubling every twelve months. One can now find 30 Gigabyte hard drives at retail prices below one hundred dollars. We can store on one of these hard drives more than 13 hours of video data in MPEG-2 format. Given the current evolution of disk technology, we can expect to find one year from now drives of twice that capacity at the same price.

We propose to use this storage capacity to preload in the customer STB a significant fraction of each video being broadcast. As a result, we will provide instant access to these videos while only having to allocate one, two or three broadcast channels per video.

Theoretical limitations of the approach

Let us compute first which minimum fraction x of a given video we must preload in order to ensure that the remainder of the video could be distributed using a given number of channels.

Consider a video of duration D whose xD first minutes are preloaded in the client STB. Let Δt represent a small time interval at a location t within the video and let us assume that $t > xD$. To avoid client underflow, the contents of this time interval must be broadcast at a minimum bandwidth b/t where b is the video consumption rate.

Summing over all intervals as Δt approaches 0, we see that the minimum bandwidth required to transmit the non-preloaded part of the video is be given by

$$\int_{xD}^{D} \frac{b}{t} dt = b(\ln D - \ln xD) = -b \ln x$$

Assume now that we want this bandwidth to be equal to a fixed number k of channels of equal bandwidth b. The minimum fraction x will then be the solution of the equation $-b\ln x = kb$ and we will have

$$x = \frac{1}{e^k} \qquad (1)$$

In particular we need to prefetch at least $1/e$ of a given video to ensure that the remaining $(e-1)/e$ fraction can be broadcast using only one channel.

The sole problem with this approach is that it would require partitioning the last $(1-x)D$ minutes of the video into a very large number of very small segments and broadcast each segment at exactly the required bandwidth. We present here two more practical solutions, one based on Juhn and Tseng's fast broadcasting protocol and the second on our pagoda broadcasting protocol.

A fast broadcasting protocol with partial preloading

Consider a video of duration D whose xD first minutes are preloaded in the client STB. To avoid client underflow, the next segment of the video must be fully received before the customer has finished watching the preloaded portion of the video. Hence this segment must be repeated at least once every xD minutes. If this segment is broadcast over a channel whose bandwidth is equal to the video consumption rate b, the maximum duration of the segment will also be equal to xD. This means that we need to preload the first half of a video in the STB in order to be able to broadcast the remainder of the video on a single video channel.

Adding a second video channel would allow us to broadcast on that channel two segments of duration xD for a total of three segments. The video could then be partitioned into four equal size segments of duration $xD = D/4$. Hence, we would need to preload one fourth of the video in the STB in order to be able to broadcast the remainder of the video on two video channels. More generally, we would need to preload the first $D/2^k$ minutes of a video to be able to broadcast the remainder of the video on k channels. This is $(e/2)^k$ times the theoretical minimum given by equation (1).

A pagoda broadcasting protocol with partial preloading

A major limitation of the previous protocol is that all segments that are broadcast on the same channel are repeated at the same frequency. Consider a video of duration D partitioned into n equal-size segments of duration $d = D/n$. Assume then that the first m segments of that video are preloaded in the customer STB. Client underflow will be avoided as long as each segment S_i with $i > m$ is fully received before it is needed. Hence, S_i should be repeated at least once every $(i-1)d$ minutes.

Subchannel	0	1	2	…	10	11
First Segment	S_{145}	S_{157}	S_{170}	…	S_{317}	S_{343}
Last Segment	S_{156}	S_{169}	S_{183}	…	S_{342}	S_{370}

Figure 3. The first channel of a PB protocol with 144 preloaded segments.

The pagoda broadcasting (PB) protocol with partial preloading partitions all its broadcasting channels into fixed-size slots whose duration will be equal to the duration d of one video segment.

Consider a PB using a large number of small segments. Let us assume that the protocol preloads the first 144 segments in the customer STB. This means that the first segment to be broadcast is segment S_{145} and this segment must be repeated at least once every $144d$ minutes. Since 144 is the square of 12, we organize the slots in the first channel into 12 *subchannels* in such a way that slot j belongs to the subchannel j (mod 12). Each subchannel has thus 1/12 of the slots and 1/12 of the bandwidth of the channel. As Figure 3 shows, subchannel 0 continuously retransmits the 12 segments S_{145} to S_{156} ensuring that each segment is repeated exactly once every 144 slots. Subchannel 1 continuously retransmits the 13 segments S_{157} to S_{169} ensuring that each segment is repeated exactly once every 156 slots with successive subchannels transmitting increasing numbers of segments. Finally, subchannel 11 continuously retransmits the 28 segments S_{343} to S_{370}. As a result, the video is partitioned into 370 segments of equal duration $D/370$. Hence we need to preload 144/370 of the video to be able to broadcast the remaining 226 segments on a single channel.

The second, third and fourth channels are allocated using the same method. Since 371 is not a square and the closest square 361 is the square of 19, the slots of the second channel are organized into 19 subchannels in such a way that slot j belongs to the subchannel j (mod 19). As Figure 4 shows, the slots in subchannel 0 will continuously retransmit segments $S_{371}$to S_{389} ensuring that each segment is repeated exactly once every 361 slots. Successive subchannels will retransmit increasing numbers of segments and subchannel 18 will continuously retransmit segments S_{918} to S_{965}. Hence, we only need to preload 144/965 of the video to be able to broadcast the remaining 821 segments on two video channels.

With a third broadcasting channel, we could partition each video into 2562 segments and would have to preload 144/2562 of the video. Adding a fourth channel would allow partitioning the video into 6855 segments. At this stage, each segment would contain 1.05 seconds of video data for a two-hour video. Assuming an average bandwidth of 5 Mb/s, each segment would still occupy around 656 kilobytes.

Table 1 summarizes our findings and compares the performance of our two protocols with the theoretical minima

Subchannel	0	1	2	…	17	18
First Segment	S_{371}	S_{390}	S_{410}	…	S_{873}	S_{918}
Last Segment	S_{389}	S_{409}	S_{429}	…	S_{917}	S_{965}

Figure 4. The second channel of a PB protocol with 144 preloaded segments.

Table 1. Comparing the two protocols

Number of Broadcasting Channels	*Duration of preloaded fraction of video for a two-hour video*		
	Theoretical Minimum	*Fast Broadcasting*	*Pagoda Broadcasting*
1	44 min 8 s	60 min	46 min 42 s
2	16 min 14 s	30 min	17 min 54 s
3	5 min 58 s	15 min	6 min 44 s
4	2 min 12 s	7 min 30 s	2 min 31 s

we derived earlier. As one can see, the performance of the pagoda broadcasting protocol with partial preloading is not far of the theoretical minimum. Preloading the first 6 minutes and 44 seconds of a two-hour video allows us to broadcast the remainder of the video on three channels while providing instant access to the video. This is exactly one half of the bandwidth required by the original pagoda broadcasting protocol to achieve a maximum waiting time of 73 seconds. Even preloading as little as the first 2 minutes and 31 second of each video would still allow us to broadcast the remainder of the video on only four channels. This is still much less than any existing broadcasting protocol.

There is an obvious trade-off between the number of videos we want to offer and how many minutes of each video we can preload in the customer STB. Assuming a disk drive capable of storing 800 minutes of video data, we could elect to store the first 6 minutes and 44 seconds of 118 videos, more data from fewer videos, or less data from more videos.

Distributing the preloaded segments

We have not discussed so far how the preloaded segments of each video are to be distributed to the customer STB's. This task will be assigned to one or two dedicated channels that will continuously broadcast the initial segments of the videos that are currently offered for viewing. Any change in this set will require each STB to download the initial segments of the new videos being offered and to store them on its hard drive. The mechanism allowing the VOD server to notify the STB's that they have new data to download could be as simple as agreeing upon some predefined time.

4. A DYNAMIC PROTOCOL WITH PARTIAL PRELOADING

The two protocols we have presented use static broadcasting schedules that are not affected by request arrivals. Hence, their bandwidth requirements are not affected by the request arrival rate. *Dynamic broadcasting* protocols [1, 7] improve upon other broadcasting protocols by adapting their broadcasting schedule to actual request arrivals. Hence they require much less bandwidth at times when there are fewer requests for the video.

To show that our approach also applies to dynamic broadcasting protocols, we present here a dynamic broadcasting protocol with partial preloading based on the fast broadcasting protocol. We will assume that our protocol uses k channels per video and partitions each video into 2^k segments of equal duration $d=D/2^k$. Since the first segment of each video will be preloaded in the customer STB, segment S_2 will be broadcast *on demand* on the first channel, that is, only when there is a request requiring that segment. Similarly segments S_3 and S_4 will be broadcast on demand on the second channel, and. more generally, segments $S_{2^{j-1}+1}$ to S_{2^j} will be broadcast on demand on the jth channel for $1 \le j \le k$.

To simplify our analysis, let assume that each channel uses an *all-or-nothing* scheduling policy: either it schedules all its segments or it schedules none of them. Then channel 1 will schedule a broadcast of segment S_2 if there has been at least one request for the video during the last d minutes. Channel 2 will schedule a broadcast of the two segments S_3 and S_4 if there has been at least one request for the video during the last $2d$ minutes. More generally, channel i will schedule a broadcast of the 2^{i-1} segments $S_{2^{j-1}+1}$ to S_{2^j} if there has been at least one request for the video during the last 2^{i-1} d minutes.

Assuming that requests to the video are exponentially distributed with average rate λ, the total bandwidth that will be required to broadcast the video will be given by:

$$B = \sum_{i=1}^{k} (1 - e^{-2^{i-1}\lambda d})$$

Replacing d by $D/2^k$, we obtain:

$$B = \sum_{i=1}^{k} (1 - e^{-2^{i-k-1}\lambda D}) \qquad (2)$$

Figure 5 represents the bandwidth requirements of our dynamic fast broadcasting protocol with partial preloading with two, three and four channels dedicated to the broadcast of a two-hour video. Request arrival rates are expressed in arrivals per hour and bandwidths are expressed in multiples of the video consumption rate. As one can see, the dynamic protocol requires significantly less bandwidth that its static counterpart when the request arrival rate remains below seven to eight requests per hour. It is thus best suited to the distribution of videos whose popularity is either unpredictable or highly variable [7].

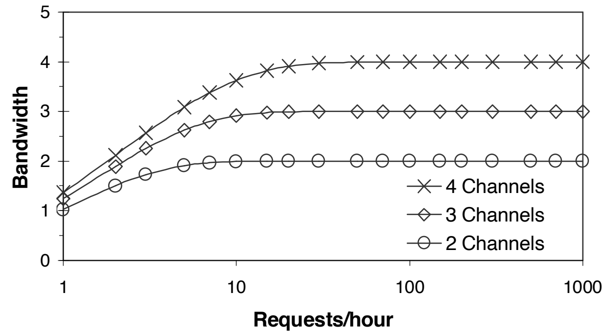

Figure 5. Bandwidth requirements of a dynamic fast broadcasting protocol with partial preloading.

5. CONCLUSIONS

We have presented two broadcasting protocols that preload the first five to twenty minutes of the videos they distribute. Both protocols provide instant access to these videos while requiring considerably less bandwidth than any other broadcasting protocol. We found that the pagoda broadcasting protocol with partial preloading required much less preloaded data per video than the fast broadcasting protocol with partial preloading to achieve the same bandwidth savings.

We have also shown how the same approach could apply to dynamic broadcasting protocols.

ACKNOWLEDGEMENTS

J.-F. Pâris was partially supported by the Texas Advanced Research Program under grant 003652-0124-1999 and the National Science Foundation under grant CCR-9988390. D. D. E. Long acknowledges the support of the National Science Foundation under grant CCR-9988363.

REFERENCES

[1] Eager, D. L. and M. K. Vernon. Dynamic skyscraper broadcast for video-on-demand. *Proc. 4th Int. Workshop on Advances in Multimedia Information Systems*, pages 18–32, Sep. 1998.

[2] Hua, K. A. and S. Sheu. Skyscraper broadcasting: a new broadcasting scheme for metropolitan video-on-demand systems. *Proc. ACM SIGCOMM '97 Conf.*, pages 89–100, Sept. 1997.

[3] Juhn, L. and L. Tseng. Harmonic broadcasting for video-on-demand service. *IEEE Trans. on Broadcasting*, 43(3):268–271, Sep. 1997.

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

[5] Pâris, J.-F. A simple low-bandwidth broadcasting protocol for video on demand, *Proc. 7th Int. Conf. on Computer Communications and Networks*, pages 690–697, Oct. 1999.

[6] Pâris, J.-F., D. D. E. Long and P. E. Mantey. A zero-delay broadcasting protocol for video on demand. *Proc. 1999 ACM Multimedia Conf.*, pages 189–197, Nov. 1999.

[7] Pâris, J.-F., S. W. Carter and D. D. E. Long. A universal distribution protocol for video-on-demand. *Proc. Int. Conf. on Multimedia and Expo 2000*, Vol. 1, pages 49–52, July 2000.

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