LIMITING THE CLIENT BANDWIDTH OF BROADCASTING PROTOCOLS FOR VIDEOS ON DEMAND

Jehan-François Pâris[1]
Department of Computer Science
University of Houston
Houston, TX 77204-3475
E-mail: paris@acm.org

Darrell D. E. Long[2]
Department of Computer Science
Jack Baskin School of Engineering
University of California
Santa Cruz, CA 95064
E-mail: darrell@cse.ucsc.edu

KEYWORDS
video-on-demand, broadcasting protocols, fast broadcasting, new pagoda broadcasting.

ABSTRACT

Broadcasting protocols can lower the cost of video-on-demand services by more efficiently distributing all videos that are simultaneously watched by many viewers. The most efficient broadcasting protocols require a customer set-top box capable of capturing data from five to seven video channels at the same time.

We show how to modify existing broadcasting protocols so that their client bandwidth would never exceed three to four channels and apply our method to the fast broadcasting and the new pagoda broadcasting protocols. Our data show that this modification has only a moderate effect on the overall performance of the two protocols because their server bandwidth never increases by more than 15 percent.

1. INTRODUCTION

In the last five years, there have been 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. The savings can be considerable because these videos are expected to account for about 40 percent of the total consumer demand (Dan et al. 1996).

Broadcasting protocols differ from all other proposals by their *proactive* approach: rather than waiting for customer requests, they continuously rebroadcast each video according to a deterministic schedule. The simplest broadcasting strategy is to retransmit each video on several dedicated data channels at equal time intervals. The major problem with this approach is the number of channels per video required to achieve a reasonable waiting time. Much more impressive results can be achieved by increasing the functionality of the set-top box (STB) connecting the customer television set with the VOD service (Viswanathan and Imielinski 1996). With a STB capable of simultaneously receiving several channels and storing their contents until they are consumed, we can use much less bandwidth to obtain the same waiting times. For example, achieving a maximum waiting time of two minutes for a two-hour video only requires a bandwidth equal to five times the consumption rate of the video (Pâris 1999).

This approach raises two issues. First, it assumes that the STB has enough local storage to store up to one half of each video being watched. In the current state of memory technology, this implies the presence of a hard drive in each STB. Second, the approach requires a STB capable of receiving data from several channels at the same time. This cannot be done without significant increases in the cost of the STB network interface.

The *skyscraper broadcasting* protocol (Hua and Sheu 1997) has already addressed these two issues. Unlike all other broadcasting protocols, skyscraper broadcasting never requires the customer STB to receive data from more than two channels at the same time. In addition, the protocol controls the amount of data the STB must store while the customer is watching a video. This approach has a major drawback, namely a very significant increase in the server bandwidth required to distribute the videos. Hence, the potential savings in STB costs achieved by skyscraper broadcasting would require bigger, more expensive video servers and a costlier network infrastructure.

There are at least two major arguments in a favor of a less radical approach. First, we can safely predict that within one or two years it will become difficult to buy a standard size disk drive that cannot store at least an entire video. Hence, the issue of reducing the STB storage requirements of broadcasting protocols will become moot. Buffering the entire contents of each video in the STB would also allow purely local implementations of pause and rewind interactive controls without any server intervention. Second, a recent study by Eager, Vernon and Zahorjan (Eager et al. 1999) indicates that proactive video distribution protocols that limit their client bandwidth to two concurrent channels will always require much more server bandwidth

[1] This research was supported in part by the Usenix Association and the Texas Advanced Research Program under grant 003652-0124-1999.
[2] This research was supported by the National Science Foundation under grant PO-10152754.

than protocols allowing concurrent downloads from three or four concurrent channels.

The solution we propose reduces the client bandwidth requirements of existing broadcasting protocols to three or four concurrent channels rather than designing new protocols. This will allow us to build upon the strengths of existing broadcasting protocols such as fast broadcasting (Juhn and Tseng 1998) and new pagoda broadcasting (Pâris 1999). As we will see, this less radical approach will also result in much more moderate increases of the server bandwidth than using a skyscraper broadcasting protocol.

2. BROADCASTING PROTOCOLS

The simplest video broadcasting protocol is *staggered broadcasting* (Dan et al. 1996). It consists of broadcasting on separate data channels multiple copies of the same video at staggered starting times. Staggered broadcasting is simple to implement and does not require any changes to the customer STB. Unfortunately, it requires a fairly large number of channels per video to achieve a reasonable waiting time. Consider, for instance, a two-hour video, approximately the average duration of a feature movie. Guaranteeing a maximum waiting time of 10 minutes would require starting a new instance of the video every 10 minutes and a total of 12 channels.

The past four years have seen the development of many efficient broadcasting protocols starting with the *pyramid broadcasting* protocol (Viswanathan and Imielenski 1996). All these protocols divide each video into *segments* that are simultaneously broadcast on multiple data channels. One of these channels transmits only the first segment of the video. The other channels transmit the remaining segments. When customers want to watch a video, they wait for the beginning of the first segment on the first channel. While they start watching that segment, their STB starts downloading enough data from the other channels to allow it to play each segment of the video in sequence. Given the fairly large number of existing broadcasting protocols, we will focus the remainder of our discussion on the two broadcasting protocols that are the most relevant to our proposal, namely fast broadcasting and new pagoda broadcasting.

Fast broadcasting (FB) (Juhn and Tseng 1998) allocates to each video to be broadcast k data channels whose bandwidths are equal to the video consumption rate b. It then partitions the 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}. Define a *slot* as a time interval equal to the duration of a segment. To prove the correctness of the protocol, we need only to observe that each segment i with $1 \le i \le 2^{k-1}$ is rebroadcast at least once every i slot. Then any client STB starting to receive data from all broadcasting channels will always receive on time all segments on time.

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. The first three channels for fast broadcasting (FB)

The *new pagoda broadcasting* (NPB) (Pâris 1999) protocol improves upon the FB protocol by using a more complex segment-to-channel mapping. As seen in Figure 2, the NPB protocol can pack nine segments into three channels whereas the FB protocol could only pack seven segments. Hence the segment size will be equal to one ninth of the duration of the video and no customer would ever have to wait more than 14 minutes for a two-hour video.

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. The first three channels for the NPB protocol

3. OUR APPROACH

A common feature of all broadcasting protocols is that the data that are broadcast on each channel periodically repeat themselves. In addition, the channels containing the first segments of the video repeat themselves more frequently than the channels containing the later segments. Consider, for instance, the case of the fast broadcasting protocol. Looking back at Figure 1, we can see that the contents of channels 1, 2 and 3 respectively repeat themselves every 1, 2 and 4 slots.

Let us focus our attention now to what happens when a VOD customer starts watching a video. With the sole exceptions of staggered broadcasting and skyscraper broadcasting, all broadcasting protocols assume that the customer STB will immediately start receiving data from all k channels that are broadcasting segments of that video. Client bandwidth and server bandwidth will then be equal. However, they will not remain equal very long because the STB will quickly start to drop the channels that have started to repeat themselves.

This observation provides the basis for our new method. Rather than letting the STB receive data from all k channels that are broadcasting video segments, we will only allow it to receive data from the first m of these k channels. Downloading data from the $k - m$ remaining channels will be progressively allowed as STB starts dropping some of the first m channels. The STB will start receiving data from channel $m + 1$ when it is finished with the first channel. It will then start receiving data from channel $m + 2$ when it is finished with the second channel and the process will continue until the STB is finished with channel $k - m$ and starts receiving data from channel k.

This new approach will have direct consequences for the segment-to-stream mapping. The correctness criterion for a conventional broadcasting protocol is to have each

segment S_i repeated at least once every i slots. It will continue to apply to the first m of the k channels that we will use to broadcast the video. Consider now a segment S_i that is mapped into one of the remaining $k - m$ channels of the video, say, channel l. The STB will not be able to download segments from this channel until it is finished with channel $k - m$, say after slot n_{k-m}. To be received on time by the STB, segment i will now have to appear at least once within the slot interval starting with slot $n_{k-m} + 1$ and ending with slot i. To guarantee that this will always happen, segment S_i will now have to be repeated at least once every $i - n_{k-m}$ slots instead of every i slots.

This more restrictive correctness criterion will result in an increase of the maximum waiting time of the protocol. Since we need now to repeat more frequently the segments that are mapped into the last $k - m$ channels of the video, we will not be able to map as many segments into these channels and will have to partition the video into fewer segments. Recalling that the maximum waiting time is always equal to the size of segment S_1, we see that limiting the client bandwidth without increasing the server bandwidth will always result in increasing the maximum waiting time of the protocol.

We will now explain how to use our approach to limit the client bandwidth of FB and NPB protocols to three or four channels. To simplify the computation of n_{k-m}, we will always assume that the STB uses a greedy downloading policy that always downloads the first arriving instance of a segment.

3.1 Fast broadcasting protocols limiting the client bandwidth

Let us first consider a FB protocol limiting the client bandwidth to three channels (FB-3). As Table 1 shows, its first three channels are identical to the first three channels of the original FB protocol. Hence segment S_8 will be the first segment mapped into channel 4.

Since the FB-3 protocol limits the client bandwidth to three channels, the STB will not be able to receive data from channel 4 until it is finished with channel 1. Because channel 1 endlessly repeats segment S_1, this would introduce a one-slot delay. As a result the contents of channel 4 will have to be repeated every $8 - 1 = 7$ slots and we will be able to map 7 segments into that channel. Hence channel 4 will contain segments S_8 to S_{14} and S_{15} will be the first segment into channel 5.

Because the STB will not be allowed to receive data from channel 5 until it is finished with channel 2, the process will start with a delay of two slots. Hence, the contents of channel 5 will have to be repeated every $15 - 2 = 13$ slots and channel 5 will contain segments S_{15} to S_{27}. Note that partitioning a video into 27 segment guarantees that the maximum client waiting time will never exceed 1/27 of the video duration. Five channels thus suffice to guarantee a maximum waiting time of 4 minutes and 27 seconds for a two-hour video.

Table 1. The first ten channels for the FB-3 protocol

Channel	Delay (slots)	Segments	Total number of segments
1	0	S_1	1
2	0	S_2 to S_3	3
3	0	S_4 to S_7	7
4	1	S_8 to S_{14}	14
5	2	S_{15} to S_{27}	27
6	4	S_{28} to S_{51}	51
7	8	S_{52} to S_{95}	95
8	15	S_{96} to S_{176}	176
9	28	S_{177} to S_{325}	325
10	52	S_{326} to S_{599}	599

Table 2. The first ten channels for the FB-4 protocol

Channel	Delay (slots)	Segments	Total number of segments
1	0	S_1	1
2	0	S_2 to S_3	3
3	0	S_4 to S_7	7
4	0	S_8 to S_{15}	15
5	1	S_{16} to S_{30}	30
6	2	S_{31} to S_{59}	59
7	4	S_{60} to S_{115}	115
8	8	S_{116} to S_{223}	223
9	16	S_{224} to S_{431}	431
10	31	S_{432} to S_{599}	832

We could achieve even lower waiting times by adding more channels. We will not detail the segment mapping process for these channels as it is well summarized in Table 1. Starting with channel 7, the delays become slightly more complicated to compute. The STB will not be allowed to receive data from channel 7 until it is finished with channel 4. The new factor to take into account is that the downloading process for channel 4 had to wait until the STB had finished downloading segment S_1 from the first channel. So the total delay for channel 7 is $1 + 7 = 6$ slots.

There is little to say about the FB protocol limiting the client bandwidth to four channels (FB-4). As Table 4 shows, its first four channels are identical to the first channels of the original FB protocol. As a result, the FB-4 protocol will be able to pack eight segments in channel 4, that is one more than was possible under the FB-3 protocol. Since the STB will now be allowed to receive data from channel 5 as soon as it is finished with channel 1, we will also be able to pack more segments in channel 5.

3.2 New pagoda broadcasting protocols limiting the client bandwidth

Let us consider first an NPB protocol limiting the client bandwidth to three channels (NPB-3). As Table 3 indicates, the first three channels are identical to the first three channels of the original NPB protocol. Hence segment S_{10} will be the first segment mapped into channel 4.

Since the NPB-3 protocol limits the client bandwidth to 3 channels, the STB will not be able to receive data from channel 4 until it is finished with channel 1. Because channel 1 endlessly repeats segment S_1, this would introduce a one-slot delay. As a result, segment S_{10} will have to be repeated every $10 - 1 = 9$ slots and, more generally, segment S_i will have to be repeated once every $i - 1$ slots. To map the maximum number of segments into channel 4, we will partition the channel into sets of three consecutive slots and assume they constitute rows of a large matrix representing the segment-to-slot mapping (Pâris 1999). We will allocate the first column of that matrix to segments S_{10} to S_{12}, the second column of the matrix to segments S_{13} to S_{16} and the third column of the matrix to segments S_{17} to S_{21}:

S_{10}	S_{13}	S_{17}
S_{11}	S_{14}	S_{18}
S_{12}	S_{15}	S_{19}
...	S_{16}	S_{20}
...	...	S_{21}

The segment-to-slot mapping for the first 15 slots of channel 4 will then repeat segments S_{10} to S_{12} once every 9 slots, segments S_{13} to S_{16} once every 12 slots and segments S_{17} to S_{21} once every 15 slots.

Because the STB will not be allowed to receive data from channel 5 until it is finished with channel 2, the process will start with a delay of 2 slots. Hence, an arbitrary segment S_i will have to be repeated every $i - 2$ slots. We will partition the channel into groups of six consecutive slots and repeat:

a) segments S_{22} to S_{27} once every 18 slots,

b) segments S_{28} to S_{35} once every 24 slots,

c) segments S_{22} to S_{27} once every 18 slots,

d) segments S_{28} to S_{35} once every 24 slots,

e) segments S_{36} to S_{40} once every 30 slots, and

f) segments S_{41} to S_{46} once every 36 slots.

With five channels, we can partition the video into 46 segments, which guarantees that the maximum client waiting time will never exceed 1/46 of the video duration, that is, 2 minutes and 36 seconds for a two-hour video. We could achieve even smaller viewing delays by increasing the total bandwidth allocated to each video. For instance, adding a sixth channel would allow partitioning each video into 107 segments and achieving a maximum viewing delay of 67 seconds for a two-hour video.

Table 3. The first eight channels for the NPB-3 protocol

Channel	Delay (slots)	Segments	Total number of segments
1	0	S_1	–
2	0	S_2, S_4 andS_5	–
3	0	S_3, S_6 to S_9	9
4	1	S_{10} to S_{21}	21
5	4	S_{22} to S_{46}	46
6	6	S_{47} to S_{107}	107
7	16	S_{108} to S_{249}	249
8	40	S_{250} to S_{582}	582

Table 4. The first eight channels for the NPB-4 protocol

Channel	Delay (slots)	Segments	Total number of segments
1	0	S_1	–
2	0	S_2, S_4, S_8,and S_9	–
3	0	S_3, S_6, S_7, S_{12} to S_{14}, S_{25} and S_{26}	–
4	0	S_5, S_{10}, S_{11}, S_{15} to S_{24}	26
5	1	S_{27} to S_{62}	62
6	8	S_{63} to S_{140}	140
7	24	S_{141} to S_{318}	318
8	24	S_{319} to S_{791}	791

Table 4 summarizes the segment to channel mapping for the NPB protocol limiting the client bandwidth to four channels (NPB-4). Its first four channels are identical to the first channels of the original NPB protocol. One particularity worth mentioning results from the fact that the NPB protocol repeats that the contents of channels 3 and 4 every 24 slots. Hence, the STB will start receiving data from both channels 7 and 8 at the same time.

4. DISCUSSION

Figure 3 displays the server bandwidth requirements of the FB-3 and FB-4 protocols as well as those of the original FB protocol and the skyscraper broadcasting protocol with a maximum width of 52. Bandwidths on the *x*-axis are expressed in channels, that is, in multiples of the video consumption rate. The corresponding maximum customer waiting times are all expressed as fractions of the video duration.

As one can see, both FB-3 and FB-4 protocols require much less server bandwidth than skyscraper broadcasting to achieve the same maximum waiting times. The bandwidth

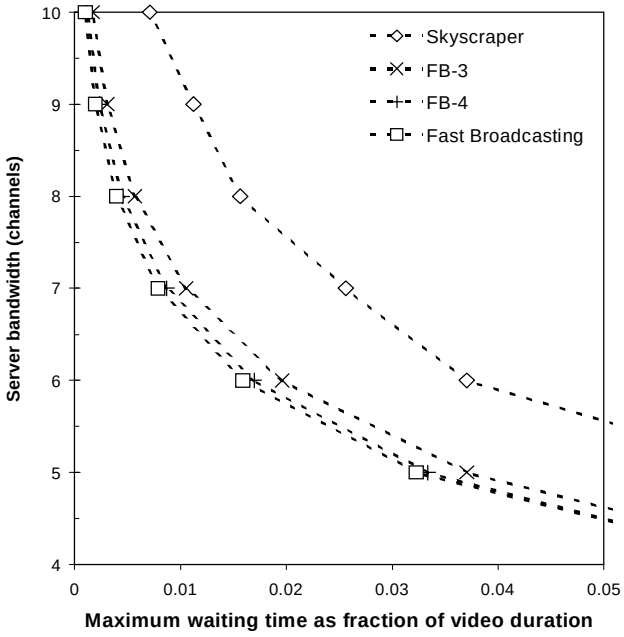

Figure 3. Compared server bandwidth requirements of the FB-3 and FB-4 protocols

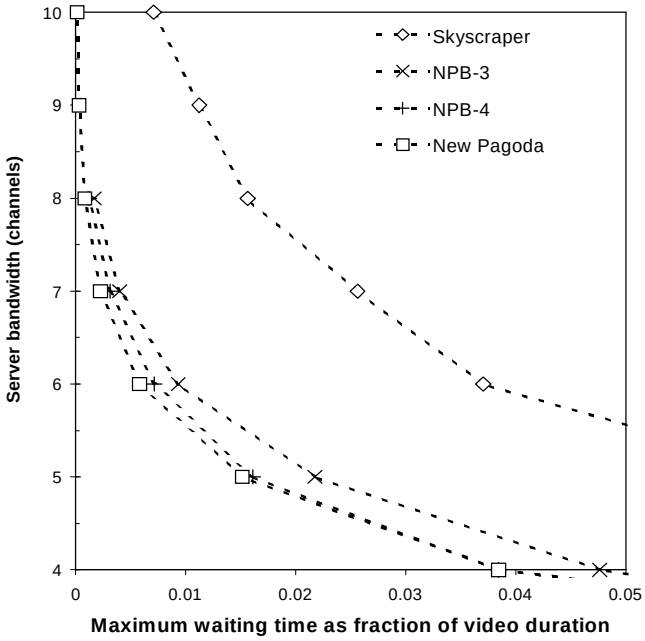

Figure 4. Compared server bandwidth requirements of the NPB-3 and NPB-4 protocols

requirements of both protocols always remain very close to those of the original FB protocol with the FB-3 protocol being only slightly worse than the FB-4 protocol.

As Figure 4 indicates, similar observations can be made about the NPB-3 and the NPB-4 broadcasting protocols. The bandwidth requirements of the NPB-4 protocol always remain very close to bandwidth requirements of the NPB protocol while the NPB-3 protocol never exceed these by more than 15 percent. The only major difference lies in the much wider gap between the performances of skyscraper broadcasting and the three pagoda-based protocols. Skyscraper broadcasting requires 10 channels to achieve a maximum waiting time of less than a minute for a two-hour video, that is 1/7200 = 0.000139 times the video duration, while all three pagoda-based protocols achieve lower waiting times with only eight channels.

Our data clearly indicate that it is possible to reduce the client bandwidth of broadcasting protocols to three channels without significantly affecting their server bandwidth. It remains to be seen whether the much higher server bandwidth requirements of skyscraper broadcasting are an artifact of the protocol or a direct result of its very low client bandwidth.

5. CONCLUSIONS

One of the most promising approaches for reducing the cost of video-on-demand services is to broadcast continuously the most frequently requested videos. The sole serious drawback of this approach is that most broadcasting protocols require a customer set-top box capable of simultaneously capturing data from five to eight video channels.

We have shown how to modify existing broadcasting protocols so that their client bandwidth would never exceed three to four channels. We have applied the method to the fast broadcasting and the new pagoda broadcasting protocols. We found that the fast broadcasting protocol was somewhat less affected by the restriction than the new pagoda broadcasting protocol, whose server bandwidth could increase by up to 15 percent.

More work is still needed to apply the method to other broadcasting protocols, among which we should mention the recent GEBB protocol (Hu et al. 1999).

REFERENCES

Eager, D. L.; M. K. Vernon; and J. Zahorjan. 1999. "Minimizing bandwidth requirements for on-demand data delivery." In *Proc. 5th Int. Workshop on Advances in Multimedia Information Systems* (Indian Wells, CA, Oct), pages 80–87.

Dan, A.; D. Sitaram; and P. Shahabuddin. 1996. "Dynamic batching policies for an on-demand video server." *Multimedia Systems*, 4 no. 3 (June):112–121.

Hu, A.; I. Nikolaidis; and P. van Beek. 1999. "On the design of efficient video-on-demand broadcast schedules." In *Proc. 7th Int. MASCOTS Symposium (*College Park, MD, Oct.), pages 262–269

Hua, K. A. and S. Sheu. 1997. "Skyscraper broadcasting: a new broadcasting scheme for metropolitan video-on-demand systems." In *Proc. ACM SIGCOMM '97 Conference* (Sep.), pages 89–100.

Juhn, L. and L. Tseng. 1998. "Fast data broadcasting and receiving scheme for popular video service. *IEEE Transactions on Broadcasting*, 44, no.1 (Mar.):100–105.

Pâris, J.-F. 1999. "A simple low-bandwidth broadcasting protocol for video on demand." In *Proc. 8th ICCCN Conference* (Boston-Natick, MA, Oct.), pages 690–697.

Viswanathan, S. and T. Imielinski. 1996. "Metropolitan area video-on-demand service using pyramid broadcasting." *Multimedia Systems*, 4 no. 4:197–208.