A hybrid broadcasting protocol for video on demand

Jehan-François Pâris[a], Steven W. Carter[b], and Darrell D. E. Long[b]

[a]Department of Computer Science
University of Houston
Houston, TX 77204-3475 USA

[b]Department of Computer Science, Jack Baskin School of Engineering
University of California, Santa Cruz
Santa Cruz, CA 95064 USA

ABSTRACT

Broadcasting protocols can improve the efficiency of video on demand services by reducing the bandwidth required to transmit videos that are simultaneously watched by many viewers. It has been recently shown that broadcasting protocols using a very large number of very low bandwidth streams for each video required less total bandwidth than protocols using a few high-bandwidth streams shared by all videos.

We present a hybrid broadcasting protocol that combines the advantages of these two classes of protocols. Our *pagoda broadcasting protocol* uses only a small number of high-bandwidth streams and requires only slightly more bandwidth than the best extant protocols to achieve a given maximum waiting time.

Keywords: video on demand, video broadcasting, harmonic broadcasting

1. INTRODUCTION

Broadcasting protocols for video on demand (VOD) aim at efficiently delivering "hot" videos—that is, videos that are likely to be watched by many viewers. Rather than transmitting one separate data stream to each customer wanting to watch a given video, these protocols repeatedly broadcast the video over several data streams in such a way that no customer will have to wait more than a few minutes before being able to start watching the video.

Two factors make the efficiency of these broadcasting protocols especially critical to the success of VOD services. First, one can conservatively estimate that at least 40 percent of the viewers will be ordering the same 10 to 20 popular videos.[1–3] Second, it is very doubtful that these customers will be ready to pay much more for VOD than they now pay for a video cassette rental or a pay-per-view program. Any reduction in the cost of distributing popular videos through the use of more efficient broadcasting protocols will thus have a direct impact on the overall cost of VOD and, ultimately, on its success on the market place.

The most important performance index for a broadcasting protocol is the total bandwidth it requires to achieve a given maximum waiting time. The last two years have seen the development of several new broadcasting protocols, among which are Viswanathan and Imielinski's *pyramid broadcasting protocol*,[4] Aggarwal, Wolf and Yu's *permutation-based pyramid broadcasting protocol*,[5] Hua and Sheu's *skyscraper broadcasting protocol*,[6] Juhn and Tseng's *harmonic broadcasting protocol*[7] and its variants.[8] These protocols share the common objective of reducing the total bandwidth required to achieve a given maximum waiting time. The results obtained so far have been impressive: some recent broadcasting protocols, such as harmonic broadcasting and its variants, require slightly less than four times the video consumption rate to provide a maximum waiting time of five minutes for a two-hour video. Thus we would only need the equivalent of eighty conventional video streams to service all the customers wanting to watch one of the top twenty videos.

These results come with a price. First, the user set-top box (STB) or set-top computer (STC) must have enough local storage to store up to 40 percent of the video. In the current state of storage technology, this implies that the

Correspondence: paris@cs.uh.edu, carter@cse.ucsc.edu and darrell@cse.ucsc.edu. Most of the work reported here was performed while the first author was on sabbatical leave at the Department of Computer Science, University of California, Santa Cruz. This research was supported by the Office of Naval Research under Grant N00014–92–J–1807.

STB must have a local disk. Second, the most efficient video broadcasting protocols are also the most difficult to implement.

To achieve a maximum waiting time of five minutes for a two-hour video, harmonic broadcasting must divide the video into 24 segments of equal duration and broadcast them over 24 parallel data streams whose bandwidths vary between 1 and 1/24 times the video consumption rate. A server broadcasting the top twenty videos would have to manage 480 independent data streams.

Managing such a large number of independent data streams is likely to be a daunting task. The only existing alternative was to use a non-harmonic broadcasting protocol such as pyramid broadcasting or skyscraper broadcasting. These protocols partition the videos to be broadcast into segments of increasing lengths and use many fewer data streams. Unfortunately, they also require much more bandwidth than harmonic protocols to achieve a given maximum waiting time.

We propose a better solution, namely a hybrid protocol that attempts to combine the best features of harmonic and non-harmonic broadcasting protocols. Like harmonic protocols, our *pagoda broadcasting protocol* partitions each video into fixed-size segments whose duration is equal to the maximum waiting time. It then maps these segments into a small number of data streams of equal bandwidth and uses time-division multiplexing to ensure that successive segments of a given video are broadcast at the proper decreasing frequencies. The result is a protocol that does not require significantly more bandwidth than harmonic protocols and does not use more data streams than non-harmonic protocols.

The remainder of the paper is organized as follows. Section 2 reviews existing video broadcasting protocols. Section 3 introduces our new protocol and compares its bandwidth requirements to those of the harmonic broadcasting protocols. Section 4 discusses possible optimizations. Section 5 contains our conclusions.

2. BROADCASTING PROTOCOLS

The simplest broadcasting protocol is *staggered broadcasting*.[4] A video broadcast under that protocol is continuously retransmitted over k distinct streams at equal time intervals. The approach does not necessitate any significant modification to the STB but requires a very large number of streams 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.

Many more efficient protocols have been proposed. All of these protocols divide each video into *segments* that are simultaneously broadcast on different data streams. When customers want to watch a video, they must first wait for the next start of the first segment. Once they start watching that segment, their STB starts downloading enough data from the other streams so that it will be able to play each segment of the video in turn.

These protocols can be subdivided into two groups. Protocols in the first group are all based on Viswanathan and Imielenski's *pyramid broadcasting* protocol.[4] These protocols use segments of increasing lengths and data streams of equal bandwidths. Protocols in the second group are all variants of Juhn and Tseng's *harmonic broadcasting protocol*.[7,8] These protocols use fixed-size segments but broadcast them at differing bandwidths.

2.1. Pyramid-Based Protocols

The pyramid-based protocols divide each video v to be broadcast into k segments S_i^v of increasing size. The entire bandwidth dedicated to the M videos to be broadcast is divided into k logical streams of equal bandwidth. Each stream is allocated a set of segments to broadcast so that stream i will broadcast segments $S_i^1,\ S_i^2,\ \ldots, S_i^M$ in turn.

Pyramid-based protocols differ in the way they control the growth of these segments. Pyramid broadcasting and permutation-based broadcasting use a geometric progression: if D^v is the duration of video v, the size of its i^{th} segment is given by

$$D_i^v = \begin{cases} \frac{D^v(\alpha-1)}{\alpha^k-1} & i = 1 \\ D_1^v\alpha^{k-1} & 1 < i \le k \end{cases}$$

where $\alpha > 1$.

Stream	Segments				
1	S_1	S_1	S_1	S_1	...
2	$S_{2,1}$	$S_{2,2}$	$S_{2,1}$	$S_{2,2}$	...
3	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$	$S_{3,1}$	...

(a)

Stream	Segments																
1	S_1				S_1				S_1				S_1				...
2	$S_{2,2}$	$S_{2,4}$	$S_{2,6}$	$S_{2,1}$	$S_{2,3}$	$S_{2,5}$	$S_{2,7}$	$S_{2,1}$	$S_{2,2}$	$S_{2,4}$	$S_{2,6}$	$S_{2,1}$	$S_{2,3}$	$S_{2,5}$	$S_{2,7}$	$S_{2,1}$	...
3	$S_{3,3}$	$S_{3,6}$	$S_{3,9}$	$S_{3,1}$	$S_{3,4}$	$S_{3,7}$	$S_{3,10}$	$S_{3,2}$	$S_{3,5}$	$S_{3,8}$	$S_{3,11}$	$S_{3,1}$	$S_{3,3}$	$S_{3,6}$	$S_{3,9}$	$S_{3,2}$	...

(b)

Figure 1. Examples of the first three streams of a video under (a) harmonic broadcasting and (b) quasi-harmonic broadcasting.

Skyscraper broadcasting uses the series

$$\bar{d} = \{1, 2, 2, 5, 5, 12, 12, 25, 25, 52, 52, \ldots\}$$

to determine segment lengths. The series is constrained to a maximum value (or *width*), and the size of the i^{th} segment is given by

$$D_i^v = \frac{d_i}{\sum_{j=1}^{k} d_j} D^v.$$

2.2. Harmonic Protocols

The harmonic protocols divide each video v to be broadcast into k segments S_i^v of equal size. With the *original harmonic broadcasting protocol*,[7] each segment is broadcast repeatedly on its own stream with a bandwidth of b/i, where b is the consumption rate of the video (see Figure 1). The customer must receive all streams at once, and that means the server and customer must support a bandwidth of

$$B_{HB}(k) = \sum_{i=1}^{k} \frac{b}{i} = b \sum_{i=1}^{k} \frac{1}{i} = bH(k)$$

for each video, where $H(k)$ is the harmonic number of k.

Unfortunately, harmonic broadcasting does not always deliver all data on time,[8] but two variants have been developed which solve that problem without imposing much additional waiting time on the customer.[8] With *cautious harmonic broadcasting*, the second stream is changed to alternate between broadcasting segments S_2 and S_3, and later segments S_i are broadcast with bandwidth $b/(i-1)$. These changes mean the customer will either receive a segment at full bandwidth when it is needed, or have the entire segment already in its buffer before it is needed. *Quasi-harmonic broadcasting* further divides each segment into m subsegments and uses a complex scheme for mapping these subsegments to streams (see Figure 1 for an example). As m grows in size, the protocol approaches the same waiting time as the original harmonic protocol.

The major advantage of these three harmonic protocols is their low bandwidth requirements. Figure 2 shows the bandwidth versus customer waiting times for harmonic and cautious harmonic broadcasting and compares them with those of pyramid broadcasting,[4] the "unconstrained" version of permutation-based pyramid broadcasting[5] and skyscraper broadcasting with a maximum width of 52.[6] Quasi-harmonic broadcasting was omitted for the sake of clarity: each value of m would have produced a different set of data and all the resulting curves would have fallen between those of harmonic broadcasting and cautious harmonic broadcasting.

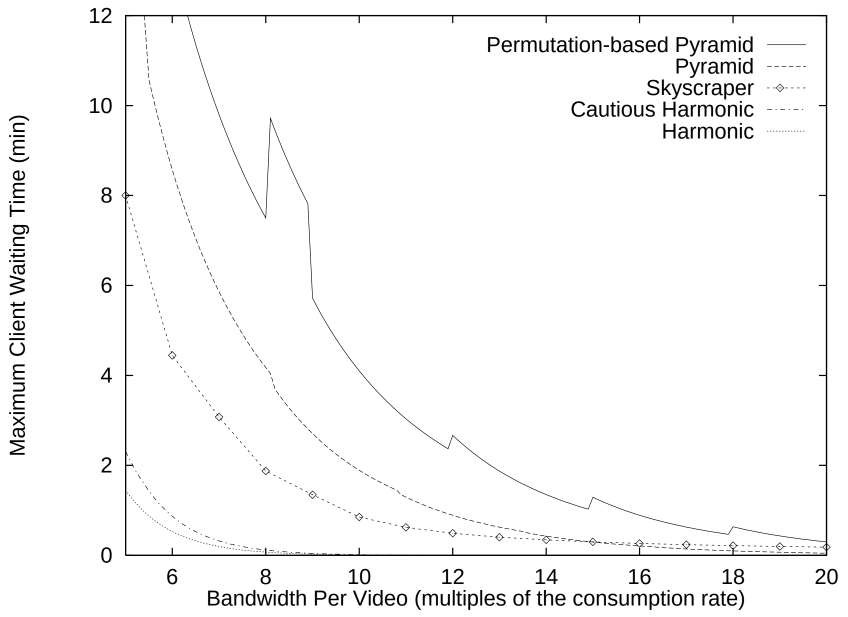

Figure 2. How harmonic broadcasting compares to other broadcasting protocols (from[8])

3. THE PAGODA BROADCASTING PROTOCOL

The lower bandwidth requirements of harmonic protocols result from the fact these protocols use fixed-size segments and transmit each individual segment at the minimum bandwidth required to guarantee on-time delivery of the data. Using segments of increasing size will necessarily be less efficient because each entire segment must be broadcast at the same bandwidth and periodicity while the data at the end of a segment could have been broadcast either at lower bandwidth or at lower frequency.

Our *pagoda broadcasting protocol* can be viewed as a hybrid of pyramid-based and harmonic protocols. Like harmonic protocols, pagoda broadcasting partitions each video into n fixed-size segments of duration d, where d is also defined as a *time slot*. Unlike harmonic protocols, it broadcasts these segments at the same bandwidth b but at different periodicities. The effect of having one dedicated stream for each segment is achieved through time-domain multiplexing among many segments sharing a few streams.

A major advantage of the approach is that we need not be concerned with all the problems resulting from streams delivering their data at a lower rate than the video consumption rate. Its only difficulty lies in selecting the proper segment-to-stream mapping and the proper broadcasting periodicity for each segment.

It would be very tempting to associate with each segment S_i a periodicity equal to $i \times d$ so that the first segment of each video would be constantly repeated, its second segment repeated every two slots and so forth. This naïve approach would not work because it would result in unacceptable fluctuations of the total bandwidth required to broadcast the video. The solution we propose avoids this problem without significantly increasing the bandwidth requirements by mapping segments to pairs of streams rather than individual streams.

The best way to introduce the segment-to-stream mapping is to look at it for the first three streams and then generalize it from there. The first stream transmits segment S_1 at frequency $1/d$. That is, it repeats the segment continuously:

S_1	S_1	S_1	S_1	S_1	S_1

Stream 2 is allocated in the following fashion: the odd slots contain segment S_2, so it is transmitted at frequency $1/(2d)$; the even slots alternate between segments S_4 and S_5, so those segments are transmitted at frequency $1/(4d)$:

S_2	S_4	S_2	S_5	S_2	S_4

The lowest-numbered segment to be transmitted by stream 3 is segment S_3, which needs to be transmitted at frequency $1/(3d)$. So each third slot will transmit S_3; the remaining slots will transmit segments S_6, S_7, S_8 and S_9 so that these segments will be transmitted at frequency $1/(6d)$:

S_3	S_6	S_8	S_3	S_7	S_9

Let us now consider the other streams. Assume that we start with an even stream l with $l = 2k$. Let z denote the index of the lowest-numbered segment S_z transmitted by stream l. As we will show later, z will always be even.

Segment S_z needs to be transmitted at minimum frequency $1/(zd)$. Consider now z contiguous slots within stream $2k$. All odd slots will be allocated to segments z to $3z/2 - 1$. They will be broadcast at frequency $1/(zd)$. The remaining slots will contain segments $2z$ to $3z - 1$, which will be broadcast at frequency $1/(2zd)$. One way to achieve that is to have the slot immediately following that allocated to segment $z + i$ alternate between transmitting segments $2z + 2i$ and $2z + 2i + 1$ with $0 \leq i < z/2$:

S_z	S_{2z}	$\cdots$	$\cdots$	$S_{3z/2-1}$	S_{3z-2}	S_z	S_{2z+1}

That is,

S_z	S_{2z} or S_{2z+1}	S_{z+1}	S_{2z+2} or S_{2z+3}	$\cdots$	$\cdots$	$S_{3z/2-1}$	S_{3z-2} or S_{3z-1}

where "S_i or S_j" denotes a slot that is allocated to either segment S_i or segment S_j using strict alternation.

Stream $2k$ is now full and we need to allocate slots in stream $2k + 1$. The lowest numbered segment we have not yet mapped is segment $S_{3z/2}$. It needs to be transmitted at a frequency $2/(3zd)$. Consider now groups of $3z/2$ contiguous slots. Each third slot will transmit one of segments $S_{3z/2}$ to S_{2z-1} at a frequency $2/(3zd)$. The remaining z slots will be allocated to the segments S_{3z} to S_{5z-1} in such a way that each pair of consecutive sets of $3z/2$ slots will contain exactly one instance of each of these $2z$ segments. They will thus be broadcast at frequency $1/(3zd)$. One way to achieve this goal is to allocate the two free slots following that occupied by segment $S_{3z/2+i}$ with $0 \leq i < z/2$ to the four segments $S_{3z+2i}, S_{4z+2i}, S_{3z+2i+1}, S_{4z+2i+1}$:

$S_{3z/2}$	S_{3z} or S_{3z+1}	S_{4z} or S_{4z+1}	$\cdots$	$\cdots$	$\cdots$	S_{2z-1}	S_{4z-2} or S_{4z-1}	S_{5z-2} or S_{5z-1}

The pair of consecutive streams $(2k, 2k+1)$ will thus contain $4z$ segments allocated in the following fashion:

Segments	Stream	Broadcasting Frequency
S_z to $S_{3z/2-1}$	$2k$	$1/(zd)$
$S_{3z/2}$ to S_{2z-1}	$2k+1$	$2/(3zd)$
S_{2z} to S_{3z-1}	$2k$	$1/(2zd)$
S_{3z} to S_{5z-1}	$2k+1$	$1/(3zd)$

Let us now derive a closed form for the number of segments $N(2k + 1)$ that can be transmitted using $2k + 1$ streams. Since the pair of streams $(2k, 2k+1)$ contains $4z$ segments starting with segment S_z, the highest-numbered segment transmitted by the pair will be segment S_{5z-1} and we have

$$N(2k + 1) = 5z - 1$$

Since S_z is the lowest-ranked segment being broadcast by stream $2k$, it is the immediate successor of the highest ranked segment broadcast by stream $2k - 1$ and we have

$$N(2k - 1) = z - 1$$

from which we can derive the recurrence

$$N(2k + 1) = 5N(2k - 1) + 4.$$

Table 1. Number of segments and maximum waiting times achieved by pagoda broadcasting with one to eight data streams.

Number of Streams	Number of Segments	Maximum Waiting Time (percent of video)
1	1	100
2	3	33.3
3	9	11.1
4	19	5.26
5	49	2.04
6	99	1.11
7	249	0.40
8	499	0.20

Since $N(1) = 1$, we can solve the recurrence and obtain

$$N(2k+1) = 5^k + \sum_{j=0}^{k-1} 4(5^j) = 2(5^k) - 1$$

Observing that $N(2k+1)$ will be odd for all values of k, it follows that the lowest-ranked segment to be broadcast in an even stream will always have an even index z.

We have dealt so far with odd numbers of streams. Suppose now we have an even number of streams, say $2k$ streams, and that once again the lowest-ranked segment to be broadcast on the stream has index $z = N(2k-1)+1$. Segment S_z must be broadcast with minimum frequency $1/(zd)$, and that means we can allocate z segments to the stream. In particular, those segments will be S_z to S_{2z-1}.

The total number of segments $N(2k)$ that can be broadcast using $2k$ streams is given by

$$N(2k) = 2z - 1 = 2N(2k-1) + 1 = 2(2(5^{k-1}) - 1) + 1 = 4(5^{k-1}) - 1$$

and thus

$$N(n) = \begin{cases} 4(5^{k-1}) - 1 & n = 2k \\ 2(5^k) - 1 & n = 2k+1 \end{cases}$$

Table 1 displays the number of segments that can be transmitted using up to eight streams and the resulting maximum customer waiting times expressed as percentages of the total duration of the video. To give a concrete example, four data streams are sufficient to guarantee that no customer will ever have to wait more than 2 minutes and 27 seconds before starting to watch a two-hour video.

Figure 3 shows the bandwidth versus customer waiting times for pagoda broadcasting, harmonic broadcasting, cautious harmonic broadcasting and skyscraper broadcasting with a maximum width of 52.[6] We assumed a video duration of two hours and used the video consumption rate as the unit of bandwidth for the four protocols under consideration. Data for pyramid broadcasting and permutation-based broadcasting were not included as we have already seen that these protocols require even more bandwidth than skyscraper broadcasting.

As the figure indicates, the performance of our pagoda protocol is almost the same as that of the cautious harmonic protocol. The performance penalty for replacing a large number of low-bandwidth streams by a few full-bandwidth streams is thus minimal. The only true disadvantage of pagoda broadcasting seems to be indeed a lesser flexibility as the bandwidth it requires is necessarily an integer multiple of the video consumption rate while quasi-harmonic broadcasting allows much more flexibility. We address this limitation in the next section.

One last aspect of the performance of pagoda broadcasting we need to analyze is its maximum disk storage cost. To derive this, we will assume that the STB never attempts to store on its disk drive segments that will be repeated

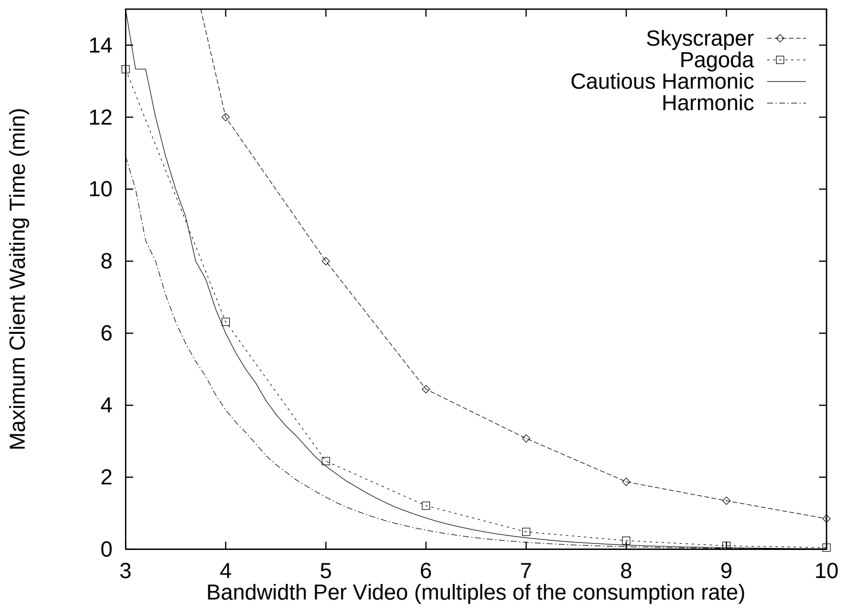

Figure 3. How pagoda broadcasting compares to other broadcasting protocols

again before being actually viewed by the customer. This assumption is reasonable because doing otherwise would require more disk storage to achieve the same maximum response time for the same total bandwidth.

To compute the maximum storage, we observe that the STB will receive data from the video server at a variable rate while the customer will consume the data at a fixed rate. Moreover the rate at which the STB will receive data from the server is a decreasing function of time as the STB will download data from less and less data streams as the video progresses. The maximum storage requirement will occur when the STB box starts downloading less data than it is consuming.

Consider first the case of a video being broadcast over an even number $n = 2k$ of streams. Since the last stream will contain over half of the segments of the video, and since the customer STB will require each segment of that stream, the STB will download at least as much data as it consumes the entire time it is downloading data. Since it will take the STB $2(5^{k-1})$ slots of time to download the video, it will have to store

$$N(2k) - 2(5^{k-1}) = 4(5^{k-1}) - 1 - 2(5^{k-1}) = 2(5^{k-1}) - 1 = N(n-1)$$

segments in its buffer. This is roughly 50 percent of the video.

The case for an odd number $n = 2k + 1$ of streams is slightly more complicated. Once again the last stream will have more segments than any other stream, but this time some of those segments will be redundant to the STB. That means the maximum storage cost will not be achieved when the STB finishes downloading the last stream of the video but when it finishes downloading stream $n-1$. At that point it will have $2(5^{k-1}) - 1$ segments from stream $n-1$ and $\lceil 8(5^{k-1})/3 \rceil$ segments from stream n in its buffer. In total, that will be

$$2(5^{k-1}) - 1 + \lceil 8(5^{k-1})/3 \rceil = \lceil 14(5^{k-1})/3 \rceil = \lceil (14/12)N(n-1) + 1/6 \rceil$$

segments, or about 46 percent of the video.

4. POSSIBLE EXTENSIONS

We present in this section two extensions of the pagoda protocol aimed at improving some particular aspects of its performance. The first extension improves the protocol performance for even numbers of data streams while the second allows several videos to share the same data stream.

Table 2. Number of segments and maximum customer waiting times achieved by the improved pagoda protocol with one to six data streams.

Number of Streams	Number of Segments	Maximum Waiting Time (percent of video)
1	1	100
2	3	33.3
3	9	11.1
4	21	4.76
5	49	2.04
6	123	0.87

4.1. Improving Performance for Even Numbers of Streams

Looking back at Figure 3, one can notice that pagoda broadcasting always performs slightly worse for even numbers of streams than for odd numbers of streams. This is because the segment-to-slot mapping for the last stream of a video broadcast is always less efficient when the video is broadcast over an even number of streams. Let stream $n = 2k$ be that last stream. Since $N(n) = 4(5^{k-1}) - 1$ and $N(n-1) = 2(5^{k-1})$, stream n will contain segments $S_{2(5^{k-1})}$ to $S_{4(5^{k-1})-1}$ or $2(5^{k-1})$ segments in total repeated at a frequency $1/(2d(5^{k-1}))$.

We propose to modify the segment-to-slot mapping of stream n in a way that would allow the high-numbered segments of the stream to be broadcast at a somewhat lower frequency. This would increase the total number of segments that could be broadcast by stream n and reduce proportionally the maximum customer waiting time.

To illustrate our new segment-to-slot mapping, let us consider a video being broadcast using four streams. Since $N(4) = 19$ and $N(3) = 9$, stream 4 will contain segments S_{10} to S_{19} repeated at a frequency $1/(10d)$. These 10 segments will be allocated 10 consecutive segment slots that will be repeated endlessly. Let us now consider sets of five consecutive slots and assume they constitute rows of large matrix representing the segment-to-slot mapping. The current segment-to-slot mapping could be represented by:

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

Rather than allocating the slots to the segments using the row major order, we could allocate them using the column major order:

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

Under this new mapping, segments S_{10} to S_{15} would continue to be broadcast at a the same $1/(10d)$ frequency as before. Segments S_{15} to S_{19} would now be broadcast at frequency $1/(15d)$, which would allow us to add segments S_{20} and S_{21}. Increasing the number of segments that can be broadcast from 19 to 21 would reduce the maximum customer waiting time from 1/19 to 1/21 of the duration of the video—that is, from 5.26 percent to 4.76 percent. This would allow us to guarantee a maximum waiting time of less than six minutes for a two-hour video.

The same procedure can be repeated for all even values of n by adjusting the number of columns in the mapping. The process is somewhat tedious as the best segment-to-slot mapping will be the result of an exhaustive search process. The results are summarized in Table 2. Since the procedure only affects video broadcasts using an even numbers of data streams, data for broadcasts using an odd number of streams are not affected. The improvement is particularly significant for $n = 6$ because six streams suffice now to guarantee a maximum waiting time of less than two minutes for a two-hour video.

4.2. Sharing Data Streams Among Videos

Since the pagoda protocol allocates full-bandwidth data streams to videos, we might encounter situations where, say, four streams do not suffice to guarantee a satisfactory waiting time for a given video but five streams would be excessive. Consider for instance the case of a three-hour video and let us assume that our objective is to guarantee a maximum waiting time of six minutes. Four streams would only guarantee a maximum waiting time of over nine minutes while five streams would bring that maximum waiting time well below four minutes. A better solution would be to allocate to the video just enough extra bandwidth to bring the maximum waiting time below six minutes. We propose to achieve this goal by allowing videos to share data streams.

Since we want to guarantee the same maximum waiting time d for all the videos we broadcast, all videos will also have the same segment size d. We will therefore partition the data streams that we want to share among several videos into fixed-size slots of the same duration d and number these slots consecutively starting at zero. Rather than allocating individually each of these slots to a specific segment of a specific video, we could partition each shared data stream into some number of *substreams* such that if the stream has p substreams, substream i would contain all slots j such that $j \bmod p = i$. Videos that require slots for extra segments could then be allocated one or more substreams.

Let us return, for instance, to the case of our three-hour video. Achieving a maximum waiting time of six minutes would require partitioning the video into 30 segments—that is, nine segments more than what can be achieved using four streams managed by the improved version of the protocol. These nine additional segments, namely segments S_{22} to S_{30}, would need to be broadcast at a minimum frequency of $1/(22d)$. We could achieve that goal by allocating, say, nine substreams of a shared stream having between 12 and 22 substreams or five substreams of a shared stream having between 8 and 11 substreams. In the first case, each of the nine substreams allocated to the video would be assigned one segment to broadcast at a total cost varying between 9/22 and 9/12 the video consumption rate. In the second case each of the five substreams selected would be assigned two segments to broadcast and the total cost would be between 5/11 and 5/8 times the video consumption rate.

5. CONCLUSIONS

Video broadcasting protocols can improve the efficiency of video on demand services by reducing the bandwidth required to transmit videos that are simultaneously watched by many viewers. It has been recently shown that broadcasting protocols using a very large number of very low bandwidth streams for each video performed better than protocols using a few high-bandwidth streams shared by all videos.

We have presented a broadcasting protocol that does not require significantly more bandwidth than the best extant protocols and will never need to use more than seven separate data streams per video. Our *pagoda broadcasting protocol* partitions each video into fixed-size segments whose duration is equal to the maximum waiting time. It maps these segments into data streams of equal bandwidth and uses time-division multiplexing to ensure that successive segments of a given video are broadcast at the proper decreasing frequencies.

We found that the bandwidth required by pagoda broadcasting to guarantee a given maximum customer waiting time is almost the same as that required by *cautious harmonic broadcasting*.[8] Moreover, the protocol never needs to store more than 50 percent of the video on the local drive of the customer set-top box.

More work is still needed to extend the method to other harmonic broadcasting protocols, such as *polyharmonic broadcasting*,[9] and to develop segment-to-stream mappings that would minimize the impact of the bandwidth fluctuations inherent to MPEG video transmission.[10,11]

REFERENCES

1. D. Clark, "Oracle predicts interactive gear by early 1994." *The Wall Street Journal*, November 10, 1993.
2. A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling policies for an on-demand video server with batching," in *ACM Multimedia*, pp. 15–23, (San Francisco, California), Oct. 1994.
3. A. Dan, D. Sitaram, and P. Shahabuddin, "Dynamic batching policies for an on-demand video server," *Multimedia Systems* **4**, pp. 112–121, June 1996.
4. S. Viswanathan and T. Imielinski, "Metropolitan area video-on-demand service using pyramid broadcasting," *Multimedia Systems* **4**, pp. 197–208, Aug. 1996.
5. C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "A permutation-based pyramid broadcasting scheme for video-on-demand systems," in *Proceedings of the International Conference on Multimedia Computing and Systems*, pp. 118–26, IEEE Computer Society Press, (Hiroshima, Japan), June 1996.
6. K. A. Hua and S. Sheu, "Skyscraper Broadcasting: a new broadcasting scheme for metropolitan video-on-demand systems," in *SIGCOMM 97*, pp. 89–100, ACM, (Cannes, France), Sept. 1997.
7. L. Juhn and L. Tseng, "Harmonic broadcasting for video-on-demand service," *IEEE Transactions on Broadcasting* **43**, pp. 268–271, Sept. 1997.
8. J.-F. Pâris, S. W. Carter, and D. D. E. Long, "Efficient broadcasting protocols for video on demand," in *Proceedings of the 6^{th} International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '98)*, pp. 127–132, July 1998.
9. J.-F. Pâris, S. W. Carter, and D. D. E. Long, "A low bandwidth broadcasting protocol for video on demand," in *Proceedings of the 7^{th} International Conference on Computer Communications and Networks (IC3N'98)*, pp. 690–697, October 1998.
10. J. Beran, R. Sherman, M. Taqqu, and W. Willinger, "Long-range dependence in variable bit-rate video traffic," *IEEE Transactions on Communications* **43**, pp. 1566–1579, 1995.
11. M. Garrett and W. Willinger, "Analysis, modeling and generation of self-similar VBR video traffic," in *Proc. ACM SIGCOMM '94*, pp. 269–280, Aug. 1994.