Video-on-Demand Broadcasting Protocols

Steven W. Carter
University of California, Santa Cruz

Darrell D. E. Long
University of California, Santa Cruz

Jehan-François Pâris
University of Houston

1 Introduction

Video-on-demand (VOD) will one day allow clients to watch the video of their choice at the time of their choice. It will be less complicated to use than a typical web browser, and it will only require that the clients have a **set-top box** (STB) connected to their television set. The STB will allow the clients to navigate a VOD server's video library, and then it will handle the reception and display of the video once the clients make a selection. It is possible that the VOD service will even be **interactive**, allowing clients to use VCR controls such as pause, rewind, and fast forward while viewing their requests. In this case the STB will also be responsible for communicating the desired interactions to the VOD server.

Unfortunately, video is an expensive medium to distribute. Even using a compression scheme such as MPEG-2, a stream of video data will take a bandwidth of at least 4 Megabits per second (Mb/s)—or roughly 70 times the capacity of a 56K modem. This sort of bandwidth will not necessarily be a problem for the client, but the VOD server will be limited by its bandwidth capacity, and so it must be able to service as many clients as possible in the bandwidth it has available.

Simply dedicating a unique stream of video data to each client request is not a good idea because that can quickly use up all of the available bandwidth on the VOD server. Better, more efficient strategies are needed.

One such set of strategies involves broadcasting videos to clients. Video rental patterns suggest that most client requests will be for the most popular 10–20 videos [6], and so broadcasting these videos ensures bounds on the amount of time clients must wait before they can watch their requests and on the total amount of bandwidth each video will require.

In the remainder of this chapter we will first define some common terms and concepts used by broadcasting protocols, and then we will describe the protocols themselves.

2 Common Terms and Concepts

The broadcasting protocols described in this chapter have many concepts in common. For example, many of the protocols divide videos into **segments**. Segments are consecutive chunks of the video. By concatenating the segments together (or by playing them in order), clients will have (see) the entire video.

The **consumption rate** is the rate at which the client STB processes data in order to provide video output. For MPEG-2, this rate can be 4 Mb/s or more. We will represent the consumption rate as b and use it as the unit of measure for VOD server bandwidth. When all video segments are the same size, we define the time it takes the STB to consume a segment as a **slot**.

Each distinct stream of data is considered to be a logical **channel** on the VOD server. These channels do not need to be of bandwidth b, and each video may have its segments distributed over several channels.

When the channel bandwidth is greater than b or when the client STB must listen to multiple channels, the STB must have some form of local storage. Since MPEG-2 requires at least 60 Megabytes per minute, the STB storage is likely to reside on disk. The storage capacity on the STB and the number of channels the STB must listen to are the two client requirements that differ between the broadcasting protocols. The fewer requirements placed on the client STB, the less it will cost to produce, and the more attractive it will be to clients.

3 Staggered Broadcasting Protocols

Staggered broadcasting [2,4,7] is the most straightforward of all broadcasting protocols. It simply dedicates a certain number of channels to each video and then staggers starting times for the video evenly across the channels. That is, given n channels for a video of duration D, staggered broadcasting starts an instance of the video every D/n minutes. The difference in starting times (D/n) is called the **phase offset**. All clients who request a video during the current phase offset will be grouped together to watch the next broadcast of the video, and so the phase offset is the maximum amount of time clients must wait for their request to be serviced.

Staggered broadcasting does not use the VOD server's bandwidth very efficiently. In order to cut the client waiting time in half, for example, the VOD provider would have to *double* the amount of bandwidth dedicated to each video. This is a much higher cost than that required by the other broadcasting protocols described in this chapter.

On the other hand, staggered broadcasting places the fewest requirements on the client STB. The STB only has to listen to a single channel to receive an entire video, and it does not require a significant amount of extra storage.

The most appealing aspect of staggered broadcasting, though, is how it can handle interactive VOD. If it were to allow interactions of arbitrary lengths—that is, allow **continuous** interactions—then each interaction would force a client out of its broadcasting group and require it to use its own unique channel. Such a strategy would quickly use up all of the available bandwidth on the VOD server and cause unacceptable delays.

By forcing clients to use multiples of the phase offset, staggered broadcasting can service interactions without creating new channels for the clients [2]. These **discontinuous** interactions can simply use the existing channels for the video. For example, each rewind interaction could cause the client to shift to an earlier channel, which would be equivalent to rewinding for an amount of time equal to the phase offset. Similarly, fast forward interactions could be implemented by shifting the client to a later channel, and pause interactions by waiting for an earlier channel to reach the same point in the video.

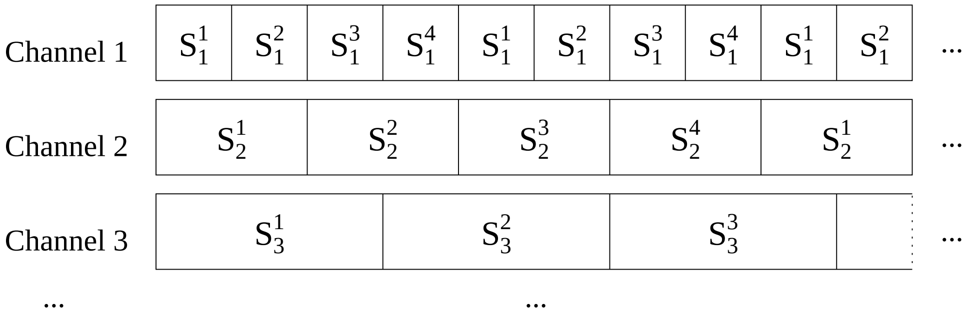

Figure 1: The segment-to-channel mapping for Pyramid Broadcasting with four videos. The superscript represents the video index.

4 Pyramid Broadcasting Protocols

Viswanathan and Imielinski introduced the idea of pyramid broadcasting in 1995. Their **Pyramid Broadcasting** protocol [19, 20] divides up each video into n segments, $S_1, \ldots, S_n$. The protocol then divides up the available bandwidth on the VOD server into n equal-bandwidth channels, $C_1, \ldots, C_n$, and broadcasts serially the i^{th} segment of each video on channel C_i (see Fig. 1).

The video segments are not equally-sized; they grow geometrically using parameter $\alpha \geq 1$. That is, given the duration of the first segment d_1 the duration of the i^{th} segment is

$$d_i = \alpha^{i-1} d_1$$

for $1 < i \leq n$. Pyramid Broadcasting received its name because, if one were to envision the segments stacked one on top of the other, they would form the shape of a pyramid.

Pyramid Broadcasting works as follows. A client must first wait for a showing of segment S_1 on channel C_1. At that point it can begin receiving and consuming the segment. For the remaining segments, the client can begin receiving S_i, for $1 < i \leq n$, at the earliest opportunity after starting the consumption of S_{i-1}.

In order for a client to receive all data on time, it must be able to start receiving S_i before it finishes consuming S_{i-1}. In other words, Pyramid Broadcasting must have

$$\frac{md_i}{b'} \leq d_{i-1} \tag{1}$$

where b' is the bandwidth of each channel, measured in multiples of b, and m is the number of videos to be broadcast. By substituting αd_{i-1} for d_i in Equation 1, we can solve for α and find that as long as

$$\alpha \leq b'/m \tag{2}$$

clients will never experience a break in playback. Viswanathan and Imielinski always used $\alpha = b'/m$ since that resulted in the smallest d_1 and thus the shortest client waiting times. A typical value is $\alpha = 2.5$.

The benefit of using Pyramid Broadcasting is that it uses VOD server bandwidth much more efficiently than staggered broadcasting protocols. This is because client waiting times only decrease linearly with bandwidth in staggered protocols but decrease exponentially with Pyramid Broadcasting. As an example, given $10b$ bandwidth per video, staggered protocols can only guarantee a maximum waiting time of 12 minutes for a two-hour video, but Pyramid Broadcasting can guarantee under two minutes.

The problem with Pyramid Broadcasting is that it makes great demands of the client equipment. Going back to Equation 2 and noting that $\alpha \geq 1$ and $m \gg 1$, the bandwidth per channel will be very high, and clients will have to be able to receive up to two channels at once. So not only will clients have to handle extremely high bandwidths, they will also have to store the large amount of data they receive early. In fact, this storage has to be large enough to hold almost the entire last two segments of the video [20], and those two segments typically account for 75–80% of the video size.

In 1997, Aggarwal, Wolf and Yu greatly reduced client equipment demands with their **Permutation-based Pyramid Broadcasting** protocol [1]. By dividing up each channel into $p \geq 1$ subchannels for each video, and by evenly staggering the starts of segments on those subchannels, they were able to prevent clients from having to receive data from more than one subchannel at one time. This reduction in bandwidth by a factor of $2mp$ also reduced the amount of storage needed by the clients, down to about a third of that required by Pyramid Broadcasting. Unfortunately, Permutation-based Pyramid Broadcasting requires more bandwidth than Pyramid Broadcasting to achieve the same client waiting times.

The next advance in pyramid broadcasting protocols came with Hua and Sheu's **Skyscraper Broadcasting** protocol [10], also in 1997. Instead of using a geometric series to determine the amount of data to place on each channel, their protocol divides up each video into n equally-sized segments, $S_1, \ldots, S_n$, and then uses the series

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

to determine the number of consecutive segments to place on each channel. Thus, channel C_1 would repeatedly broadcast segment S_1, C_2 would repeatedly broadcast S_2 and S_3, C_3 would repeatedly broadcast S_4 and S_5, and so on. This series grows much more slowly than the one employed by the previous pyramid protocols—it is analogous to using $\alpha \approx 1.5$—but each channel only requires bandwidth b, so Skyscraper Broadcasting can use many more channels. Also, the channels are constrained to a maximum number of segments (or **width**), so the protocol avoids the problem of having an extremely long data block on the last channel that needs to be stored on the client STB. Lastly, the client only has to receive at most two channels at once, so transfer rates are kept low. In total, Skyscraper Broadcasting manages to keep the good parts of Permutation-based Pyramid Broadcasting—lower transfer rates and lower storage requirements—while also reducing the client waiting times found in Pyramid Broadcasting.

In 1998, Eager and Vernon improved Skyscraper Broadcasting by allowing it to dynamically schedule channels and to use a more efficient segment-to-channel series [8]. However, in both versions, since the protocols attempt to keep transfer rates low at the client end, they waste bandwidth on the VOD server. Consider, for example, a situation where a video uses five channels and 15 $(1+2+2+5+5)$ segments. Segment S_{15} is broadcast every fifth time slot on channel C_5, and that is three times as often as needed.

An opposite approach to Skyscraper Broadcasting was used by Juhn and Tseng in 1997 with their **Fast Broadcasting** protocol [11, 14]. This protocol uses the series

$$\{\ 1, 2, 4, 8, 16, 32, 64, \ldots\ \}$$

to determine the number of segments to broadcast on each channel, so it uses many more segments than Skyscraper Broadcasting, and its associated client waiting times are much lower. Unfortunately, clients must receive data from all channels at once so their transfer rates are much higher and they must store up to half of the video locally. Fast Broadcasting also allows a mode where clients only receive a single channel at once and don't require local storage, but the maximum waiting time for this mode is half the duration of the video, which is likely to be too long for any video popular enough to be broadcast.

Pâris, Carter, and Long improved efficiency even further in 1999 with their **Pagoda Broadcasting** protocol [17, 18]. This protocol attempts to broadcast video segments as infrequently as possible while still maintaining an even transfer rate to the client. Like Skyscraper and Fast Broadcasting, Pagoda Broadcasting divides up each video into n equally-sized segments, $S_1, \ldots, S_n$, but then it uses the series

$$\{\ 1, 3, 5, 15, 25, 75, 125, \ldots\ \}$$

to determine the number of segments for each channel. Unlike the previous protocols, it does not require that the segments appearing on each channel

Channel 1	S_1	S_1	S_1	S_1	S_1	S_1	S_1	S_1	S_1	...
Channel 2	S_2	S_4	S_2	S_5	S_2	S_4	S_2	S_5	S_2	...
Channel 3	S_3	S_6	S_7	S_3	S_8	S_9	S_3	S_6	S_7	...
...					...					
Channel i	S_z	S_{2z}	S_{z+1}	...	S_z	...	$S_{3z/2-1}$	S_{3z-1}	S_z	...
Channel i+1	$S_{3z/2}$	S_{3z}	S_{3z+1}	...	$S_{3z/2}$	...	S_{5z-2}	S_{5z-1}	$S_{3z/2}$	...
...					...					

Figure 2: The segment-to-channel mapping for Pagoda Broadcasting.

be consecutive. In fact, after the first channel, Pagoda Broadcasting uses pairs of channels when assigning segments, and so, for example, segments S_2 and S_4 are assigned to channel C_2 but S_3 is assigned to C_3 (see Fig. 2).

The segment-to-channel mapping works as follows. The first segment S_1 is repeatedly broadcast on channel C_1. Then, given $z-1$, the highest index of a segment appearing on an earlier channel (or channels), Pagoda Broadcasting assigns $4z$ segments to the current pair of channels. The first channel contains:

- Segments S_z to $S_{3z/2-1}$, inclusive, broadcast every other time slot. Since there are $z/2$ segments, each is broadcast once every z time slots.

- Segments S_{2z} to S_{3z-1}, inclusive, also broadcast every other time slot. Each segment is broadcast once every $2z$ time slots.

The second channel contains:

- Segments $S_{3z/2}$ to S_{2z-1}, inclusive, broadcast every third time slot. Since there are $z/2$ segments, each is broadcast once every $3z/2$ segments.

- Segments S_{3z} to S_{5z-1}, inclusive, broadcast two out of every three time slots. Thus, each segment is broadcast once every $3z$ time slots.

Fig. 2 shows a sample of these two channels.

In order for a client to use Pagoda Broadcasting, it must first wait for an instance of S_1 on channel C_1. At that point it can start receiving and consuming S_1, and it can also start receiving data from every other channel dedicated to the video. Since each segment S_i is broadcast at least once every i slots of time, the client will either receive the segment ahead of time and have it in its buffer when it is needed, or it will receive the segment directly from the VOD server when it is needed.

Since Pagoda Broadcasting schedules many more segments per channel than Fast Broadcasting, the required client and VOD server bandwidths are much lower for any given maximum client waiting time. However, Pagoda Broadcasting still requires enough local storage for about half of the video.

None of the pyramid protocols described in this section have been shown

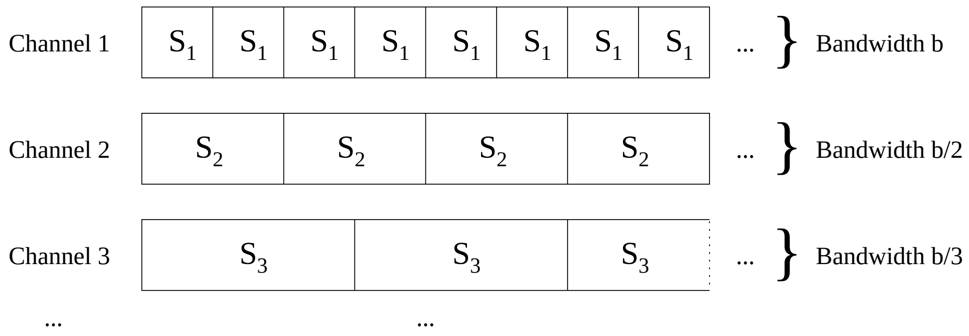

Figure 3: The segment-to-channel mapping for Harmonic Broadcasting.

to work effectively with interactive VOD.

5 Harmonic Broadcasting Protocols

The first harmonic protocol was Juhn and Tseng's **Harmonic Broadcasting** protocol [12], introduced in 1997. This protocol divides each video into n equally-sized segments, $S_1, \ldots, S_n$, and continuously broadcasts those segments on their own data channels. In particular, segment S_i is broadcast on channel C_i using bandwidth b/i (see Fig. 3). The sum of the channel bandwidths is

$$\sum_{i=1}^{n} \frac{b}{i} = b \sum_{i=1}^{n} \frac{1}{i} = bH(n)$$

where $H(n)$ is the harmonic number of n. It is this series that gives the protocol its name.

For a client to use Harmonic Broadcasting, it must first wait for the start of an instance of S_1 on C_1. Then it can begin receiving and consuming S_1 while simultaneously receiving and storing data from every other channel. Once the client has a complete segment S_i, it can stop listening to channel C_i, but it still must support a bandwidth of $bH(n)$ since that is how much it will receive during the first slot of time.

The appeal of Harmonic Broadcasting is that the harmonic series grows very slowly. As with other protocols, the maximum client waiting time is the duration of a segment, but Harmonic Broadcasting can use hundreds of segments and still not require much bandwidth. For example, with a bandwidth of only $5b$, it can support $n = 82$ segments and have a maximum client waiting time of under a minute and a half for a two-hour video.

On the down side of Harmonic Broadcasting is the fact that it sends data to clients (well) before it is needed, and so the clients must have some sort of local storage. In fact, Harmonic Broadcasting requires storage for about 37% of the video as long as $n \geq 20$ [12].

One other problem with Harmonic Broadcasting is that it does not quite work as originally presented. Consider again Fig. 3 and suppose a client arrives in time to start receiving the second instance of S_1. During that time it will receive and store the second half of S_2, but when it comes time to consume S_2 it will only be receiving the first half at half the consumption rate, and so much of the data will arrive after it is needed.

A straightforward way to fix Harmonic Broadcasting is simply to force the client to delay for a slot of time after receiving the first segment of the video [15]. Then the client will have each segment completely in its local storage before starting to consume it, and no data will arrive late. The only drawback to this **Delayed Harmonic Broadcasting** protocol is that its maximum client waiting time is twice that of the original harmonic protocol.

In 1998, Pâris, Carter, and Long presented three harmonic protocols that fixed the correctness problem of Harmonic Broadcasting but used bandwidth more efficiently than Delayed Harmonic Broadcasting. With the **Cautious Harmonic Broadcasting** protocol [15], C_1 stays the same but C_2 alternately broadcasts S_2 and S_3, and C_i, for $3 \leq i < n$, broadcasts S_{i+1} at bandwidth b/i. This scheme guarantees that a client will either receive a segment at full bandwidth when it is needed or have the segment in its local storage before it is needed. Since C_2 uses bandwidth b instead of $b/2$, Cautious Harmonic Broadcasting requires about $b/2$ more bandwidth than Delayed Harmonic Broadcasting, but its maximum client waiting time is only one slot instead of two.

The second protocol, **Quasi-Harmonic Broadcasting** [15], divides each of the n segments into a number of subsegments (or **fragments**), and, while each segment is broadcast on its own distinct channel, the subsegments are not broadcast in order. By rearranging the subsegments and by broadcasting the first subsegment twice as often as the other subsegments, clients can start consuming a segment while still receiving parts of it and still receive all of the data on time. As with Cautious Harmonic Broadcasting, the maximum client waiting time is one slot, but the bandwidth converges to $bH(n)$ as the number of subsegments increases.

Polyharmonic Broadcasting [16] uses an integral parameter $m \geq 1$ and forces each client to wait m slots of time before consuming data. While waiting, the client can receive data, and so each segment can be broadcast with a lower bandwidth than that used in the other harmonic protocols. In particular, each channel C_i, for $1 \leq i \leq n$, can use bandwidth $b/(m+i-1)$ to continuously broadcast segment S_i. Since a great deal of the total value of $H(n)$ comes from its first few terms, and since Polyharmonic Broadcasting bypasses some of those terms when $m > 1$, even though clients must wait m slots of time, Polyharmonic Broadcasting can use m times as many segments, and so the real waiting time is still the same as Cautious and Quasi-Harmonic Broadcasting. Moreover, the bandwidth used by Polyharmonic Broadcasting converges to $bH(n)$ as m increases, and, for realistic

Table 1: Client STB requirements for the broadcasting protocols.

Broadcasting Protocol	Storage Requirement (percent of video)	Bandwidth Requirement (multiples of b)
Staggered	0	0
Pyramid	75	4–5[a]
Permutation-based Pyramid	20	2–3
Skyscraper	10	1–2
Fast	50	5–8
Pagoda	45	5–7
Harmonic	40	4–6

[a]per video broadcast

parameters, uses less bandwidth for a given maximum client waiting time than Quasi-Harmonic Broadcasting.

Juhn and Tseng also presented a new harmonic protocol in 1998: the **Enhanced Harmonic Broadcasting** protocol [13]. This protocol uses a strategy similar to Polyharmonic Broadcasting, but it suffers from the same correctness problem as their original harmonic protocol.

None of the harmonic protocols described in this section have been shown to work effectively with interactive VOD.

6 Summary

The work of recent years has led to the the development of many efficient broadcasting protocols for video-on-demand (VOD). These protocols use a fixed amount of VOD server bandwidth to guarantee a maximum waiting time before clients can begin viewing their requests.

Although the protocols appear to be different, most share a fundamental strategy: that if clients frequently request the same video, and if the clients have some form of local storage, then data occurring later in the video does not need to be broadcast as often as data occurring earlier in the video. By taking advantage of this strategy, the broadcasting protocols can save bandwidth on the VOD server and either allow more videos to be broadcast or allow the VOD server to be manufactured with less expense.

Client STB requirements for the broadcasting protocols are summarized in Table 1, and the VOD server bandwidth requirements for the protocols are shown in Fig. 4. We assumed that Skyscraper Broadcasting had a maximum width of 52 and that Polyharmonic Broadcasting used $m = 4$. These are representative values, and they should not greatly affect the results. Also, for Fig. 4, the waiting times are for a two-hour video.

Between Table 1 and Fig. 4, there is no clear "best" protocol. Polyharmonic Broadcasting poses the lowest bandwidth requirements on the VOD server, but its numerous data channels per video may be too much for a client STB to stage into its disk drive. Pagoda Broadcasting has a low VOD server bandwidth as well, but the amount of bandwidth it requires the client

STB to receive may be too much for the client's network connection or disk drive to handle. Staggered broadcasting has by far the worst VOD server bandwidth requirements, but it puts no extra load on the client STB, and it is the only broadcasting protocol that as yet allows for interactive VOD.

There are several open issues relating to broadcasting protocols. First of all, none of the most efficient protocols have been shown to work with interactive VOD. It is not clear whether such extensions are not possible, are not efficient, or just have not yet been explored. Secondly, the broadcasting protocols assume (implicitly if not explicitly) that the video signal has a fixed bit rate. This is likely not to be the case. The MPEG compression schemes all use variable bit rates, for example. One solution is to reserve enough bandwidth in a channel for the maximum bit rate, but that will waste a lot of bandwidth. Are there better solutions? Finally, broadcasting protocols do not handle well fluctuations in video popularity. If horror videos, for example, are more popular at night than in the day, is there a graceful way for the broadcasting protocol to change the amount of bandwidth the videos use? Staggered broadcasting has the easiest solution to this problem—it can simply change the phase offset (although even this might be a problem with interactive VOD)—but what of the other protocols?

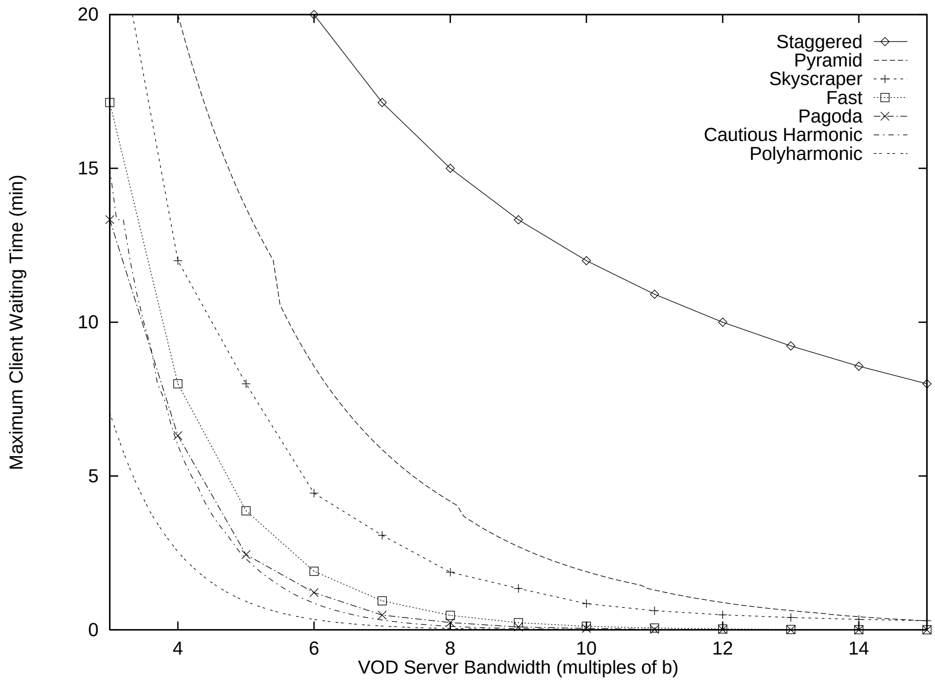

Figure 4: VOD server bandwidth requirements for the broadcasting protocols.

7 Defining Terms

Channel: A stream of data on the VOD server.

Consumption rate: The rate at which the client STB processes data in order to provide video output.

Interactive video-on-demand: A VOD service which also allows clients to use VCR controls such as pause, rewind, and fast forward.

Phase offset: In staggered broadcasting, the amount of time between starts of a particular video.

Segment: A consecutive chunk of a video.

Set-top box (STB): The piece of equipment linking a client to a VOD server.

Slot: The amount of time it takes for an STB to consume a segment of data.

Video-on-demand (VOD): A service allowing clients to watch the video of their choice at the time of their choice.

Width: In Skyscraper Broadcasting, the maximum number of segments per channel.

References

[1] Charu C. Aggarwal, Joel L. Wolf, and Philip S. Yu. A permutation-based pyramid broadcasting scheme for video-on-demand systems. In *Proceedings of the International Conference on Multimedia Computing and Systems*, pages 118–26, Hiroshima, Japan, June 1996. IEEE Computer Society Press.

[2] Kevin C. Almeroth and Mostafa 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(5):1110–22, August 1996.

[3] Steven W. Carter and Darrell D. E. Long. Improving bandwidth efficiency on video-on-demand servers. *Computer Networks and ISDN Systems*, 30(1–2):99–111, January 1999.

[4] Asit Dan, Perwez Shahabuddin, Dinkar Sitaram, and Don Towsley. Channel allocation under batching and VCR control in video-on-demand systems. *Journal of Parallel and Distributed Computing*, 30(2):168–79, November 1995.

[5] Asit Dan and Dinkar Sitaram. Buffer management policy for an on-demand video server. Technical Report RC 19347, IBM Research Division, T.J. Watson Research Center, January 1993.

[6] Asit Dan, Dinkar Sitaram, and Perwez Shahabuddin. Scheduling policies for an on-demand video server with batching. In *ACM Multimedia*, pages 15–23, San Francisco, CA, USA, October 1994. ACM.

[7] Asit Dan, Dinkar Sitaram, and Perwez Shahabuddin. Dynamic batching policies for an on-demand video server. *Multimedia Systems*, 4(3):112–21, June 1996.

[8] Derek L. Eager and Mary K. Vernon. Dynamic skyscraper broadcasts for video-on-demand. In *Proceedings of the 4th Internantional Workshop on Advances in Multimedia Information Systems*, pages 18–32, Berlin, Germany, September 1998. Springer-Verlag.

[9] Leana Golubchik, John C. S. Lui, and Richard R. Muntz. Adaptive piggybacking: a novel technique for data sharing in video-on-demand storage servers. *Multimedia Systems*, 4(30):140–55, June 1996.

[10] Kien A. Hua and Simon Sheu. Skyscraper Broadcasting: a new broadcasting scheme for metropolitan video-on-demand systems. In *Proceedings of SIGCOMM 97*, pages 89–100, Cannes, France, September 1997. ACM.

[11] Li-Shen Juhn and Li-Meng Tseng. Fast broadcasting for hot video access. In *Proceedings of the Fourth International Workshop on Real-Time Computing Systems and Applications*, pages 237–43, Taipei, Taiwan, October 1997. IEEE Computer Society Press.

[12] Li-Shen Juhn and Li-Ming Tseng. Harmonic broadcasting for video-on-demand service. *IEEE Transactions on Broadcasting*, 43(3):268–71, September 1997.

[13] Li-Shen Juhn and Li-Ming Tseng. Enhanced harmonic data broadcasting and receiving scheme for popular video service. *IEEE Transactions on Consumer Electronics*, 44(2):343–6, May 1998.

[14] Li-Shen Juhn and Li-Ming Tseng. Fast data broadcasting and receiving scheme for popular video service. *IEEE Transactions on Consumer Electronics*, 44(1):100–5, March 1998.

[15] Jehan-François Pâris, Steven W. Carter, and Darrell D. E. Long. Efficient broadcasting protocols for video-on-demand. In *Proceedings of the International Symposium on Modelling, Analysis, and Simulation of Computing and Telecom Systems*, pages 127–32, Montreal, Canada, July 1998. IEEE Computer Society Press.

[16] Jehan-François Pâris, Steven W. Carter, and Darrell D. E. Long. A low bandwidth broadcasting protocol for video on demand. In *Proceedings of the 7th International Conference on Computer Communication and Networks*, pages 609–7, Lafayette, LA, USA, October 1998. IEEE Computer Society Press.

[17] Jehan-François Pâris, Steven W. Carter, and Darrell D. E. Long. A hybrid broadcasting protocol for video on demand. In *Proceedings of the 1999 Multimedia Computing and Networking Conference*, pages 317–26, San Jose, CA, USA, January 1999.

[18] Jehan-François Pâris, Steven W. Carter, and Darrell D. E. Long. A simple low-bandwidth broadcasting protocol for video-on-demand. In *Proceedings of the 8th International Conference on Computer Communications and Networks*, Boston, MA, USA, October 1999. To appear.

[19] S. Viswanathan and T. Imielinski. Pyramid broadcasting for video-on-demand service. *Proceedings of SPIE—the International Society for Optical Engineering*, 2417:66–77, 1995.

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

Further Information

Broadcasting protocols are proactive in that bandwidth is reserved in advance. There are also several reactive protocols that reserve bandwidth on the fly. Some of these protocols include **adaptive piggybacking** [9], **interval caching** [5], and **stream tapping** [3]. Since their bandwidth requirements depend on the access patterns of the videos, reactive protocols tend to be less efficient than broadcasting protocols for popular videos, but can be very efficient for "lukewarm" or cold videos.

Current research on broadcasting protocols can be found in a variety of places. Relevant journals include *IEEE Transactions on Broadcasting*, *Computer Networks and ISDN Systems*, and *Multimedia Systems.* Conferences include *ACM Multimedia*, *Infocom*, the *Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, the *International Conference on Computer Communication and Networks (ICCCN)*, and the *Multimedia Computing and Networking Conference (MMCN).*