OPTIMIZING VIDEO-ON-DEMAND THROUGH REQUESTCASTING

By

Julie Pochueva

A Thesis Submitted in
Partial Fulfillment of the
Requirements for the degree of

Master of Science

in
Computer Science

at
The University of Wisconsin-Milwaukee
July 1999
OPTIMIZING VIDEO-ON-DEMAND THROUGH REQUESTCASTING

By

Julie Pochueva

A THESIS SUBMITTED IN
PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

COMPUTER SCIENCE

at

The University of Wisconsin-Milwaukee
July 1999

Ethan V. Munson

Graduate School Approval

Date
OPTIMIZING VIDEO-ON-DEMAND 
THROUGH REQUESTCASTING 

By 
Julie Pochueva 

The University of Wisconsin-Milwaukee, 1999
Under the Supervision of Professor Ethan V. Munson

ABSTRACT 

Video-on-demand (VOD) designs typically feature either request or broadcast architectures. Both have limitations. Request architectures experience a limit in the number of clients that can be adequately serviced. Broadcast architectures require large, often unavailable, bandwidth. In addition, it is difficult to limit viewing to
a target audience. In this paper, we present a new architecture for a metropolitan VOD service that we name requestcasting. Our architecture combines the two general approaches of request and broadcast, but not their respective limitations. With the requestcast architecture, we implement an improved pyramid broadcasting protocol. Our synchronized method of employing pyramid broadcasting is key to providing robust VOD service.
I would like to thank my advisor Dr. Ethan Munson for developing my interest in the topic, fruitful discussions and his support throughout the years and the miles. I would also like to thank my husband Denis Pochuev for verifying my work and my love. Finally, I would like to thank my cat Vodka for her ability to cheer me up when working long hours got me down.
Contents

Acknowledgements v

1 Introduction 1
  1.1 Server-Side Data Protocol ........................................ 2
  1.2 Client-Side Requirements ........................................ 4
  1.3 VOD Architectures .............................................. 5
    1.3.1 Request Architecture ...................................... 5
    1.3.2 Broadcast Architecture .................................... 6

2 Requestcast Architecture 7
  2.1 Pyramid Broadcasting ........................................... 8
    2.1.1 Pyramid Broadcasting Protocol ............................ 9
  2.2 Analysis of Pyramid Broadcasting ............................... 11
    2.2.1 Pyramid Broadcasting Example ............................ 13
  2.3 Synchronized Pyramid Broadcasting .............................. 15
    2.3.1 Synchronized Pyramid Broadcasting Example ................. 18
  2.4 Implementation .................................................. 19

3 Further Development and Conclusions 26
  3.1 Further Development ........................................... 26
3.2 Conclusions ................................................. 26
# List of Figures

1. VOD Architecture ............................................. 2
2. (a) Request and (b) Broadcast Architectures .................. 5
3. Requestcast Architecture ....................................... 8
4. Conceptual vs. Actual Physical Channel Division .............. 9
5. Pyramid Broadcasting .......................................... 10
7. Synchronized Pyramid Broadcasting ............................ 16
8. Pyramid Broadcasting without an Integer-Fixed Alpha and/or a Simultaneous Segment Start ............................. 17
9. Requestcast Internal Broadcast Model .......................... 19
10. Broadcast Formations in Terms of Channels/Segments ....... 21
11. Requestcast Server Architecture in Terms of Multiple Clients .... 22
12. Requestcast Server Architecture in Terms of One Client ........ 23
13. Buffer Use for Synchronization ................................ 24
Chapter 1

Introduction

Video-on-demand (VOD) designs need to deliver continuous video with client-real-time, VCR-function support while satisfying the following criteria:

1. minimal channel bandwidth
2. short client waiting times
3. dependable service to a large number of clients, and
4. full support of VCR functions.

The ultimate VOD solution is true video-on-demand (TVOD). In this model, video viewing is immediately available after the client’s request. Unless complex algorithms are employed, one video stream per client is mandatory. If we allow a short waiting time between video requests and viewing capability, then it is possible to achieve a significant drop in the required number of video streams. Typically, this scheme is referred to as near video-on-demand (NVOD) [14]. There are alternative definitions of near video-on-demand which imply that NVOD is VOD that does not feature all VCR functions [9, 19] or that NVOD is simply broadcasting the same program on different channels with varying start times [15]. We do not adhere to these alternative interpretations.

Although a short delay before viewing commencement is tolerated for NVOD, any delay after commencement is not acceptable. Fast forward, pause, rewind, et al. all
have to function as if the client were watching a video with a physical VCR. To do so, several major entities of VOD design need to be considered. They are server-side data protocol, client-side requirements and architecture (the client, the server and the digital delivery system between them\ Figure 1).

1.1 Server-Side Data Protocol

All video data that clients can access is stored on a video server. The manner in which video data is stored to and retrieved from memory is the server-side data protocol. Storage types, data transfer rates and data access and storage methods are all protocol issues.

Server storage type is usually determined by the current technology and financial cost. Typically, a huge amount of memory (several terabytes for a video library [15]) is used to house an entire video archive.

Data transfer rates are dependent upon what memory type is being used and how it is implemented. Speed designators, such as “1x”, “2x”, “4x”, “6x” and “8x”, define multiples of the original playback speed of first generation CD-ROM players. For a CD-ROM player, a 1x speed translates to 153,600 bytes/sec. This is usually rounded down to 150 Kbyte/sec. Therefore, a 1x player reads 150 Kbyte/sec. A 2x player reads at 300 Kbyte/sec. A 4x player reads at 600 Kbyte/sec, and so on. The data transfer rate rises to 3600 Kbyte/sec when a 24x CD-ROM drive is used [1, 17]. Latency, due to non-zero seek time, and other data transfer delays also impact the rate.

Figure 1: VOD Architecture
Several techniques have been developed and examined for data access and storage. Disk striping, matrix phasing and jukeboxes are among them.

Disk striping [3] involves the distribution of videos across several disks in stripes. This technique produces a simulated borrowing of bandwidth. For example, assume a single popular video and six less-popular videos are divided into sevenths and each seventh is placed on a separate disk. If each disk can support five viewers, then it is now possible to support 35 viewers of the popular video at a time. The less-popular videos will occasionally be accessed, but at a relatively insignificant rate.

Matrix phasing [4, 16] evolved from disk striping. This alternative involves the organization of video data into an \([m \times n]\) matrix of video strips. These strips are consecutively numbered in either row-major or column-major order. The entire matrix is then stored in row-major order if numbered in column-major order or column-major order if numbered in row-major order. These columns or rows are stored and retrieved consecutively from disk. The strip size is selected so that a reasonable amount of video is transferred during each disk access.

The concept behind jukeboxes mimics that of music jukeboxes. When a client requests a video, the tape or disk holding that data is sought and accessed.

Please note that additional server requirements—the handling of extensive client messaging and the implementation of VCR functions—will impact server-side data protocol. The inclusion of these two additional requirements is dependent upon the presence of client-side storage in the VOD design. If client-side storage for an entire video is not present, then the server has to be capable of transferring data while being receptive to client requests for VCR functionality and fulfilling them. This is a challenge as VOD requires the delivery of huge amounts of data, typically 20 to 30 color pictures per second [15].

In practice, all server-VCR-support designs have difficulty functioning within current cable bandwidths. The most problematic function is fast forward. This inability to provide complete VCR functionality plagues many designs. Several researchers have focused their attention on this problem.

Shenoy and Vin [18] propose that for fast forwarding, every nth frame should be transmitted on a base substream and all other frames should be sent on an enhanced
substream. Fast forward accesses the base substream and normal playback accesses both substreams.

Dey-Sircar et al. [8] examine such schemes as borrowing bandwidth and resolution from other streams and delaying bandwidth availability.

Golubchik et al. [10] support adaptive piggybacking when display rates of two relatively close streams are adjusted until they can merge into a single stream as a solution.

Dan et al. [6, 7] discuss how the VCR functions of rewind and pause can be implemented using contingency channels.

The problem common to most server-VCR-support design solutions is that they do not necessarily reduce the amount of required bandwidth. Most of the above proposals need additional, possibly unavailable, bandwidth for multiple streams or channels and extensive bi-directional communication. Furthermore, the ability to handle huge amounts of client messaging and to provide VCR functionality distracts the server from its primary responsibility streaming video data to clients.

1.2 Client-Side Requirements

Clients need to be able to decode and display audio and video data. As MPEG-2 (Moving Pictures Experts Group) was established to provide the standard for video and audio compression for broadcast environments and television applications [15], this format would most likely be employed. MPEG decoders are presently available in hardware and software implementations.

If VCR support is to be conducted at the client end, then in addition to being able to decode and display, enough memory is needed to house the largest video available. The term “largest” does not necessarily mean the largest video itself. Those videos that require more client memory than what is available can be divided into multiple parts like a miniseries.

For instance, acceptable quality can be achieved by using a MPEG-2 video stream at 3 Mbit/sec [2]. We can calculate how much client memory is required for a 2 hour video as

\[
3 \text{ Mbit/sec} \cdot 3600 \text{ sec/hour} \cdot \frac{1}{8} \text{ (byte to bit ratio)} \cdot 2 \text{ hours} = 2.7 \text{ Gbyte}.
\]
1.3 VOD Architectures

VOD architectures describe the connection between client, server and possibly other components of the digital delivery system. Two popular architecture designs are request and broadcast.

1.3.1 Request Architecture

Request architectures connect clients to a server with a bi-directional channel (Figure 2). Whenever a client would like to access a video, a request is sent to the server. The server, conducting its protocol, reads the requested video from storage and transmits it to the client.

The major plus to this architecture is a low bandwidth requirement from the client to the video source. The major shortcoming of this architecture is the relatively low upper bound on the number of supportable clients. Unless a modern communication protocol, such as ATM, is equipped with a multicast facility [12, 13], each video stream has to be devoted to only one user. Therefore, the number of clients is limited by how many can be supported by one stream and by how many streams are available. Assuming that multicasting is viable for VOD purposes, the same video stream could be sent to multiple users without additional server overhead. Batching [6, 7] is based on this assumption.

Batching decreases the necessary number of initial video streams. If requests for the same video arrive within a certain time period of each other, then they can
be grouped together and serviced using a single stream. Clients may experience a wait while they are being grouped together. Ultimately, more clients can be serviced through batching provided that the wait on the client end is acceptable [6, 7]. Batching trades start-time-latency for bandwidth.

There are problems with batching. Although the number of supportable clients may increase, there is still a limit. In order to provide VCR functionality, users must be able to separate themselves from their batch, possibly increasing the number of video streams required. If the allocated bandwidth cannot support additional streams, then service (VCR functionality) has to be denied. Assigning a larger number of clients per batch may assist in this shortcoming, but client waiting times typically increase as the number of required clients per batch increases.

1.3.2 Broadcast Architecture

In broadcast architectures, the server continuously transmits video data independent of client requests (Figure 2). Clients consume information by tuning into different channels. This architecture provides access for a larger number of clients at the cost of additional bandwidth.

Conventional broadcast methodologies feature the transmission of video from start to finish. As employed in pay-per-view, clients tune into channels at certain times. Although simple and reliable, this conventional method needs special protocols to limit viewership to target audiences and does not provide reasonable client waiting times. If the start of the video is missed, then the client has to wait for its broadcast again, whenever that may be.

Wong examined the issue of whether it is better to use request or broadcast architectures [20]. Generally, he concludes that the number of supportable clients and available bandwidth heavily influences whether request architecture or broadcast architecture is more advantageous. Broadcast architectures are desirable when the number of clients exceed a certain threshold that cannot be supported by request architectures [19].

Our ideal would be to create an architecture that would combine the best of both low bandwidth requirements and support for a large number of clients.
Chapter 2

Requestcast Architecture

The term “requestcasting” suggests a combination of broadcast and request architectures. The key concept of this architecture is a switching board. It serves as an interface between the server and its clients. One can also think of the switching board as the dividing line between broadcast and request architectures. The board is connected to the server by a high bandwidth, uni-directional channel and to each of the clients by low bandwidth, uni- or bi-directional channels (Figure 3).

The presence of uni- or bi-directional channels between the switching board and the client is dependent upon how clients are to request videos. If requests are to be conducted through a third mechanism, such as the telephone, then uni-directional channels can be used. If the client is to directly access the server, then bi-directional channels are mandatory.

The server continually broadcasts all available videos to the switching board. Once a client’s request is sent to the switching board, it transmits back to the client only the requested part of the video stream for download into local storage and consumption. As the downloading continues, the switching board is responsible for choosing appropriate data to be sent to each client. This includes proper execution of the data transfer protocol.

Later, we will describe in more detail each part of our architecture as well as their parameters. First, we need to determine what kind of protocol can be used to continuously broadcast all videos from the server to the switching board and how
Figure 3: Requestcast Architecture

this protocol would decrease client waiting time compared to the straightforward broadcasting approach previously mentioned.

2.1 Pyramid Broadcasting

The pyramid broadcasting protocol, the work of S. Viswanathan and T. Imielinski, provides a tradeoff between client waiting time and bandwidth. Unlike most exchanges of this nature, a moderate increase in bandwidth results in a dramatic decrease of client waiting time [19].

A basic assumption of pyramid broadcasting is that the channel between the server and each client has bandwidth larger than that required to transmit a number of videos available for viewing. Another assumption is that the download rate for video data is larger than the fastest rate that a video can be consumed. The pyramid broadcasting protocol consists of server-side and client-side components and requires client-side memory [19].

Before describing this protocol, we will mention a simple but effective way of dividing a physical channel into several logical channels. It is a form of TDM (time-division multiplexing), and should assist in the understanding of pyramid broadcasting. Video data from 1 to n sources is divided into tiny segments. A round robin algorithm is used to place each first segment from all n sources sequentially. All second segments are then placed sequentially and so on. Although the conceptualization of physical
2.1.1 Pyramid Broadcasting Protocol

Pyramid broadcasting describes the organization of videos on logical channels by the server and the access protocol for clients. For simplicity, it is expressed in terms of one video. Scaling for multiple videos is trivial.

1. Server Side:

- The network of bandwidth $B$ is divided into $K$ logical channels. Each channel bandwidth is equal to $\frac{B}{K}$.
- Each video $s$ of size $D$ is divided into $K$ segments $s_1, s_2, \ldots, s_K$. The size of the $i$-th segment $s_i$ is denoted by $D_i$. Concatenation of segments is denoted by $\cdot$, $s = s_1 \cdot s_2 \cdot \ldots \cdot s_K$. Also, $D = D_1 + D_2 + \cdots + D_K$. 
Each segment should be larger than the previous one by some constant factor $\alpha = \frac{D_i}{D_{i+1}}$ for $i = 1, 2, \ldots, K - 1$.

Each of the $K$ logical channels continuously broadcasts one of the $K$ segments (Figure 5). The segments are to be placed in sequential order on the logical channels.

2. Client Side:

- Wait for the beginning of the first segment.
- Begin downloading the first segment of the video into local memory at the start of its broadcast on the first logical channel. Consume it concurrently.
- After a segment is downloaded, switch to the next logical channel.
- Starting at the beginning, download new segment.
- Repeat last two steps until entire video is downloaded.
- Continue consumption of video throughout entire download sequence.

Note that downloading and consumption are concurrent. The client-side memory stores data to be consumed later. To ensure continuity, the rate of download has to be larger than the rate of consumption. Each of the logical channels requires a bandwidth equal to the download rate. Thus, necessary bandwidth is quite large. This
is a trade-off between larger numbers of supportable clients and smaller bandwidth requirements. Broadcast architectures favor larger numbers of supportable clients. Our ultimate goal is to create an architecture that provides the most efficient trade off between the two.

2.2 Analysis of Pyramid Broadcasting

To determine if pyramid broadcasting helps us achieve our goal, let us denote by $d$ the rate of downloading and by $c$ the rate of consumption. Note that $c$ does not have to be the viewing rate of the video. “Consumption” in this case has a somewhat more abstract meaning. For instance, the rate of consumption can be equal to the rate of fast forward. As before, $D_i$ denotes the size of the $i$-th segment of the video, $D$ is the size of the entire video and $\alpha$ denotes the ratio $\frac{D_{i+1}}{D_i}$. $B$ denotes the bandwidth of the channel dedicated to the video of size $D$, which is divided into $K$ logical channels.

The main idea of the pyramid broadcasting protocol is that by using a larger bandwidth, the waiting time can be significantly reduced. If conventional broadcasting (the base case of pyramid broadcasting in which $K$ is fixed as 1) is used, then the maximum waiting time $t_{\text{conv}}$ is equal to $\frac{D}{d}$. If pyramid broadcasting is used, then $t_{\text{pyr}} = \frac{D}{d}$ since clients only have to wait for the start of the first segment. Knowing that

$$D = \sum_{i=1}^{K} D_i = D_1 + D_2 + \ldots + D_K$$

and

$$\alpha = \frac{D_{i+1}}{D_i} = \frac{D_2}{D_1} = \frac{D_3}{D_2} = \ldots = \frac{D_K}{D_{K-1}} \Rightarrow D_i = D_{i-1} \cdot \alpha, \text{ for } i = 2, 3, \ldots, K$$

we see that

$$D = D_1 + D_1 \alpha + D_1 \alpha^2 + \ldots + D_1 \alpha^{K-1} = D_1 (1 + \alpha + \alpha^2 + \ldots + \alpha^{K-1}) = D_1 \sum_{i=0}^{K-1} \alpha^i.$$

Using the formula for geometric progression,

$$\sum_{i=0}^{K-1} \alpha^i = \frac{\alpha^K - 1}{\alpha - 1}.$$
we get
\[ D = D_1 \frac{\alpha^K - 1}{\alpha - 1} \Rightarrow D_1 = D \frac{\alpha - 1}{\alpha^K - 1}. \]

Finally, by substituting for \( D_1 \) in \( t_{pyr} = \frac{D}{d} \), we obtain

\[ t_{pyr} = \frac{D}{d} \cdot \frac{\alpha - 1}{\alpha^K - 1}. \]

In other words, \( t_{pyr} \) is \( \frac{\alpha^K - 1}{\alpha - 1} \) times smaller than \( t_{conv} \). Also, when \( K \) (and bandwidth \( B \) as \( B = Kd \)) grows linearly, client waiting time decreases exponentially.

To see exactly how good (or bad) this is, we need to see how \( \alpha \) can be expressed in terms of the physical parameters of the architecture such as bandwidth. Notice that the maximum waiting time to download the \( i + 1 \)-th segment, \( \frac{D_{i+1}}{d} \) (which is equal to \( \alpha \frac{D_i}{d} \)), cannot exceed the time interval between the end of the download of the \( i \)-th segment and the end of the consumption of the \( i \)-th segment.

\[ t_{\text{maxwait}}(s_{i+1}) = t_{\text{downloading}}(s_{i+1}) \leq t_{\text{consuming}}(s_i) - t_{\text{downloading}}(s_i) \]

\[ \frac{D_{i+1}}{d} \leq \frac{D_i}{c} - \frac{D_i}{d} \]

\[ \frac{D_{i+1}}{d} = \alpha \frac{D_i}{d} \Rightarrow \alpha \frac{D_i}{d} \leq \frac{D_i}{c} - \frac{D_i}{d} \]

\[ \frac{\alpha}{d} \leq \frac{1}{c} - \frac{1}{d} \]

\[ \alpha \leq \frac{d}{c} - 1 \]

And thus, the best (maximum) value of \( \alpha \) is \( \frac{d}{c} - 1 \). If \( \alpha \) is greater than \( \frac{d}{c} - 1 \), then seamless video viewing cannot be guaranteed. We also would not want \( \alpha \) to be less than \( \frac{d}{c} - 1 \). As \( \alpha \), increases, the maximum waiting time \( t_{pyr} \) decreases as \( D_1 \) becomes smaller.

An interesting observation can be made here. Pyramid broadcasting protocol is pointless unless the download rate is at least twice as large as the consumption rate. This is due to the following. We know that

\[ \alpha > 1 \]

or the “pyramid” would look like a “ladder” or a “top” (Figure 6). This implies

\[ \frac{d}{c} - 1 > 1 \]
and, therefore,
\[ d > 2c. \]

Please note that \( \alpha \) cannot be arbitrarily large. Although, as previously stated, the larger the \( \alpha \), the smaller the \( t_{pyr} \), feasible and realistic bandwidths to support the download rate \( d \) must be respected.

We now return our attention back to the formula \( t_{pyr} = \frac{D}{d} \cdot \frac{\alpha - 1}{\alpha - 1} \). Substituting \( \frac{d}{c} - 1 \) for \( \alpha \), we get
\[
t_{pyr} = \frac{D}{d} \cdot \frac{\left(\frac{d}{c} - 1\right) - 1}{\left(\frac{d}{c} - 1\right)^K - 1} = \frac{D}{d} \cdot \frac{\frac{d}{c} - 2}{\left(\frac{d}{c} - 1\right)^K - 1}.
\]

We also know that \( t_{pyr} = \frac{D_1}{d} \) and \( D = D_1 \sum_{i=0}^{K-1} \alpha^i \). When we substitute for \( D_1 \) and \( \alpha \), respectively, we see that
\[
t_{pyr} = \frac{D}{d} \sum_{i=0}^{K-1} \alpha^i = \frac{D}{d} \left(\sum_{i=0}^{K-1} \left(\frac{d}{c} - 1\right)^i\right). \tag{1}
\]

We will use this expression later.

### 2.2.1 Pyramid Broadcasting Example

For the purposes of this example, assume that the number of segments/channels \( K \) is 5, and that the length of a video \( \frac{D}{c} \) is 2 hours. We know that the approximate consumption rate \( c \) required for viewing video with MPEG-2 compression is 3 Mbit/sec.
From this, we can determine the values for $\alpha$, $d$, $B$ and $D$. As just demonstrated,
\[ d > 2c \]
and
\[ \alpha = \frac{d}{c} - 1. \]
So, let us set
\[ d = 3c. \]
Therefore,
\[ d = 9 \text{Mbit/sec}. \]
From this, we see that
\[ \alpha = \frac{9}{3} - 1 = 2. \]
We also know that
\[ B = Kd = 5 \cdot 9 = 45 \text{Mbit/sec}. \]
As $\frac{D}{c} = 2 \text{hours}$ and $c = 3 \text{Mbit/sec}$,
\[ D = 7200 \text{sec} \cdot 3 \text{Mbit/sec} = 21,600 \text{Mbit}. \]

With these values, we can compare the maximum waiting time of pyramid broadcasting $t_{\text{pyr}}$ to conventional broadcasting $t_{\text{conv}}$.

\[ t_{\text{pyr}} = \frac{D}{d} \cdot \frac{\alpha - 1}{\alpha^K - 1} = \frac{21,600}{9} \cdot \frac{2 - 1}{2^5 - 1} \approx 77 \text{sec}. \]
\[ t_{\text{conv}} = \frac{D}{d} = \frac{21,600}{45} = 480 \text{sec} \]
Note that $d = B$ for $t_{\text{conv}}$ as $K = 1$. Therefore, from $B = Kd$,
\[ d = \frac{B}{K} = \frac{45}{1} = 45 \text{Mbit/sec}. \]

We see that $t_{\text{pyr}}$, with $K = 5$, is about 6.2 times smaller than $t_{\text{conv}}$. From this, a question can be raised. Is division of the allocated bandwidth into 5 channels optimal? In this particular case the answer can be obtained by examining a few alternatives. Assume that the duration of the video $D$, the total bandwidth $B$ and the consumption rate $c$ are fixed at their previous values. The following table provides the information about the maximum waiting time for the different number of logical channels.
<table>
<thead>
<tr>
<th>$K$</th>
<th>$d = \frac{B}{K}$</th>
<th>$\alpha = \frac{d}{c} - 1$</th>
<th>$t_{pyr}$</th>
<th>$t_{sec}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>22.50 Mbit/sec</td>
<td>6.50</td>
<td>2 min 08 sec</td>
<td>3.75</td>
</tr>
<tr>
<td>3</td>
<td>15.00 Mbit/sec</td>
<td>4.00</td>
<td>1 min 09 sec</td>
<td>7.00</td>
</tr>
<tr>
<td>4</td>
<td>11.25 Mbit/sec</td>
<td>2.75</td>
<td>1 min 00 sec</td>
<td>8.03</td>
</tr>
<tr>
<td>5</td>
<td>09.00 Mbit/sec</td>
<td>2.00</td>
<td>1 min 17 sec</td>
<td>6.20</td>
</tr>
<tr>
<td>6</td>
<td>07.50 Mbit/sec</td>
<td>1.50</td>
<td>2 min 19 sec</td>
<td>3.46</td>
</tr>
<tr>
<td>7</td>
<td>06.43 Mbit/sec</td>
<td>1.14</td>
<td>5 min 14 sec</td>
<td>1.53</td>
</tr>
</tbody>
</table>

It is clear from the table that the previous choice of 5 logical channels was not optimal as it did not minimize the waiting time. The choice of 4 channels does. Finding the optimal number of channels for all parameters analytically is difficult. The approximation to the solution can be found numerically, but this is a tedious and unnecessary process. We know that the number of channels has to be an integer. A simple method to find the optimal number is to build a similar table. Note that in such a table, the number of rows is equal to $\lfloor \frac{B}{cK} \rfloor$ (there is no need to check when $K = 1$ as this is conventional broadcasting), which as we see below is bounded from above.

\[ 1 < \alpha = \frac{d}{c} - 1. \]

As $d = \frac{B}{K}$,

\[ 1 < \alpha = \frac{B}{cK} - 1 \Rightarrow K < \frac{B}{2c}. \]

Since $B$, the total bandwidth, cannot grow infinitely large, the number of entries in the table is bounded.

### 2.3 Synchronized Pyramid Broadcasting

In this analysis of pyramid broadcasting, we find our first step toward developing a server protocol. It stems from wondering if pyramid broadcasting could be improved upon. Please note that we are not the first to evolve pyramid broadcasting. Both skyscraper broadcasting [11] and permutation-based pyramid broadcasting have origins in pyramid broadcasting.
In the pyramid broadcasting design, the channel between the server and a client is divided into several logical channels. For the majority of implementations of this subdivision, it is not difficult to enforce the synchronization between the channels. It is through this synchronization that we would ultimately be able to further reduce either the waiting time or the total bandwidth.

In our synchronized version (Figure 7), as in pyramid broadcasting protocol, the channel is divided into $K$ logical channels. The video $s$, of length $D$, is divided into segments $s_1 \ldots s_K$, which are continuously transmitted through different logical channels. Unlike general pyramid broadcasting, the ratio of the lengths of segments in this protocol $\frac{D_{i+1}}{D_i} = \alpha$ has to be an integer. It is through this fixing of $\alpha$ as an integer and by a mandatory simultaneous start of segment transmission, that synchronization is possible. If this is not enforced, then it is possible to just miss the start of a segment (Figure 8). This results in the maximum segment waiting time, for all practical purposes, being the span of that entire segment. Due to the fact that the lengths ratio of segments is an integer, the beginning of the $(1 + \alpha j)$-th repetition of segment $s_i$ and the beginning of the $(1+j)$-th repetition of $s_{i+1}$ for ($j = 0, 1, \ldots$) will be synchronized.

As before, after the downloading the $i$-th segment, the client switches to a different channel and downloads the next segment $s_{i+1}$. The synchronization ensures that the waiting time before the client can start downloading the $i+1$-th segment is no greater than $\frac{2-1}{\alpha} \cdot \frac{D_{i+1}}{d}$, where $d$ is the rate of the downloading. This implies that the relation
between $d$ and the rate of consumption $c$ is the following.

$$t_{\text{maxwait}}(s_{i+1}) = \frac{\alpha - 1}{\alpha} t_{\text{downloading}}(s_{i+1}) \leq t_{\text{consuming}}(s_{i}) - t_{\text{downloading}}(s_{i})$$

$$\frac{\alpha - 1}{\alpha} \cdot \frac{D_{i+1}}{d} = (\alpha - 1) \frac{D_i}{d} \leq \frac{D_i}{c} - \frac{D_i}{d}$$

$$\frac{\alpha - 1}{d} \leq \frac{1}{c} - \frac{1}{d}$$

It follows that, in this case,

$$\alpha \leq \frac{d}{c}$$

and

$$t_{\text{sync}} = t_{\text{maxwait}}(s_1) = \frac{D_1}{d} = \frac{D}{d} \cdot \frac{\alpha - 1}{\alpha^{K-1}} = \frac{D}{d} \cdot \frac{(\frac{d}{c} - 1)}{((\frac{d}{c})^{K-1})}.$$ 

Alternatively, as $t_{\text{sync}} = \frac{D_1}{d}$ and $D = D_1 \sum_{i=0}^{K-1} \alpha^i$,

$$t_{\text{sync}} = \frac{D}{d} \sum_{i=0}^{K-1} \alpha^i = \frac{D}{d} \left(\frac{\alpha^{K-1} - 1}{\alpha - 1}\right).$$

$$\frac{D}{d \cdot t_{\text{sync}}} = \sum_{i=0}^{K-1} \left(\frac{d}{c}\right)^i = 1 + \frac{d}{c} + \left(\frac{d}{c}\right)^2 + \ldots + \left(\frac{d}{c}\right)^{K-1}$$

We now return to the formula for $t_{\text{pyr}} (1)$.

$$t_{\text{pyr}} = \frac{D}{d \left(\sum_{i=0}^{K-1} \left(\frac{d}{c} - 1\right)^i\right)}$$

$$\frac{D}{d \cdot t_{\text{pyr}}} = \sum_{i=0}^{K-1} \left(\frac{d}{c} - 1\right)^i = 1 + \left(\frac{d}{c} - 1\right) + \left(\frac{d}{c} - 1\right)^2 + \ldots + \left(\frac{d}{c} - 1\right)^{K-1}$$
Note that each member is larger for the equation involving $t_{sync}$.

$$\sum_{i=0}^{K-1} (\frac{d}{c})^i \geq \sum_{i=0}^{K-1} (\frac{d}{c} - 1)^i \Rightarrow \frac{D}{d \cdot t_{sync}} \geq \frac{D}{d \cdot t_{pyr}}$$

$$t_{pyr} \geq t_{sync}$$

Therefore, when the parameters of the network are the same, the synchronized pyramid broadcasting protocol will always provide the shorter waiting time.

The above considerations also demonstrate that introducing synchronization increases the value of $\alpha$ by 1 ($\frac{d}{c}$ for synchronized pyramid broadcasting versus $\frac{d}{c} - 1$ for general pyramid broadcasting) if all other parameters of the architecture remain unchanged. This reduction can be used to decrease waiting time or total bandwidth.

### 2.3.1 Synchronized Pyramid Broadcasting Example

We provide two examples for comparison purposes between general and synchronized pyramid broadcasting when the bandwidth $B$ and the maximum waiting time $t$ are held the same between them. In both examples, as in the pyramid broadcasting examples, $c = 3$ Mbit/sec and $D = 21,600$ Mbit. Also, $d = 3c = 9$ Mbit/sec.

To hold the bandwidth $B$ constant at 45 Mbit/sec between the $t_{sync}$ and $t_{pyr}$ examples, $K = 5$ as $B = Kd$. This results in an $\alpha$ of 3 ($\alpha = \frac{d}{c}$). The maximum waiting time in this case is

$$t_{sync} = \frac{D}{d} \cdot \frac{\alpha - 1}{\alpha^K - 1} = \frac{21,600}{9} \cdot \frac{3 - 1}{3^5 - 1} = 19.83 \text{ sec}.$$  

We already calculated that the shortest waiting time we could achieve in regular pyramid broadcasting $t_{pyr}$ is 1 min. In this particular case, synchronization helped cut maximum waiting time by more than a factor of 3.

Let us see what happens if the number of logical channels $K$ is reduced to 4, while the downloading rate $d$ remains the same at 9 Mbit/sec. The maximum waiting time is

$$t_{sync} = \frac{21,600}{9} \cdot \frac{3 - 1}{3^4 - 1} = 60 \text{ sec}.$$
When this is the case, the waiting time is equal to one of the general pyramid broadcasting examples in which $K = 4$. In the synchronized case,

$$B = Kd = 4 \cdot 9 = 36 \text{Mbit/sec}.$$  

The necessary bandwidth is reduced by 20 percent (9 Mbit/sec) from the 45 Mbit/sec required for general pyramid broadcasting. Again, this is due to synchronization.

Now that we know our synchronized pyramid broadcasting scheme provides a significant reduction in either maximum waiting time $t$ or bandwidth $B$, we need to determine if it can function under the constraints of current technology with our proposed architecture. To do so, we segue into a discussion about requestcasting implementation.

### 2.4 Implementation

Our requestcast architecture features a server that continuously broadcasts video data to a switching board through a high bandwidth channel. It is possible to simplify the architecture by including the switching board as part of the server and implementing our synchronized pyramid broadcasting (SPB) protocol as the server data access protocol. This allows us to eliminate the high bandwidth channel used in the “broadcasting” part of our architecture. Informally, we can say that broadcasting occurs inside the server (Figure 9).
To do so, we start with 24x CD-ROM drives as primary server storage. Each of the K video segments are stored on one or more CD-ROMs. If the segment is small like the first one, then multiple copies of it may be sequentially stored. If the segment is large like the last one, then two or more CD-ROMs may be required to house one copy of the segment. Whether or not this is necessary is dependent upon CD-ROM capacities and the amount of video data $D_i$ in each segment.

As it has been shown that $D_1 = D \frac{\alpha - 1}{\alpha^K - 1}$ and $D_i = D_{i-1} \cdot \alpha$, we can determine how much data is required for each segment. As

$$D_2 = D_1 \alpha = D \frac{\alpha - 1}{\alpha^K - 1} \cdot \alpha,$$

$$D_3 = D_2 \alpha = D \frac{\alpha - 1}{\alpha^K - 1} \cdot \alpha^2,$$

$$D_4 = D_3 \alpha = D \frac{\alpha - 1}{\alpha^K - 1} \cdot \alpha^3,$$

and so on,

$$D_i = D \frac{\alpha^i - 1}{\alpha^K - 1} \cdot \alpha^i.$$

Note that demand distributions could effect the amount of data in each segment $K$. In the case of commercial movies, 80 percent of the demand is for new releases or top ten movies of the year [5]. This accounts for a few videos out of an archive of thousands. Requestcasting can integrate this tendency into VOD design to reduce clients’ average waiting time. Unlike the situation when the demand distribution has to be known in advance [3, 4, 6, 7], requestcasting is more adaptive. More-popular videos would get higher bandwidth, reducing their waiting time. For our specific requestcast implementation, this implies that more-popular videos would have larger numbers of CD-ROM drives devoted to them. Each additional CD-ROM drive reduces client waiting time for that video by approximately a factor of $\alpha$. Despite the increase in the waiting time for less-popular videos, the average waiting time for all videos will decrease.

For example, assume there are two videos functioning with a download rate $d$ of 9 Mbit/sec, a consumption rate $c$ of 3 Mbit/sec and $K = 5$. The size of each video $D$ is 21,600 Mbit. When this is the case

$$t_{average} = t_{sync} = \frac{D}{d} \cdot \frac{\alpha - 1}{\alpha^K - 1} = \frac{21,600}{9} \cdot \frac{3 - 1}{3^5 - 1} = 19.83 \text{ sec.}$$
If we know that for every one request for the first video there are nine requests for
the second video, then we can “take a channel” from the less-popular video and give
it to the more-popular video. The less-popular video would function with 4 channels
K and the more-popular video with 6. This has the following effect on $t_{\text{average}}$ (Figure
10). For the popular video:

$$t_{\text{sync}} = \frac{D}{d} \cdot \frac{\alpha - 1}{\alpha^K - 1} = \frac{21,600}{9} \cdot \frac{3 - 1}{3^6 - 1} = 6.59 \text{ sec.}$$

For the less-popular video:

$$t_{\text{sync}} = \frac{D}{d} \cdot \frac{\alpha - 1}{\alpha^K - 1} = \frac{21,600}{9} \cdot \frac{3 - 1}{3^4 - 1} = 60.0 \text{ sec.}$$

$$t_{\text{average}} = \frac{(9 \cdot 6.59) + (1 \cdot 60.0)}{10} = 11.9 \text{ sec}$$

We have decreased the waiting time for the popular video by a factor of 3 (our $\alpha$
value) and $t_{\text{average}}$ for the two videos by 40 percent.

The next step is to determine if CD-ROMs can transmit at the fast forward rate.
Assume that the fast forward which we provide is three times faster than the constant
bandwidth 3 Mbit/sec. Consumption time $c$ equals 3 Mbit/sec $\cdot$ 3 = 9 Mbit/sec. As
in the last example for synchronized pyramid broadcasting, we choose $\alpha = 3$. This
implies that the download rate $d = c \cdot \alpha = 9 \text{ Mbit/sec} \cdot 3 = 27 \text{ Mbit/sec}$. Modern
CD-ROMs can provide a sustained transfer rate of 28.8 Mbit/sec. This is faster than
our needed download rate.

To ensure seamless delivery, we need to account for the “restart” time of the
CD-ROMs. This is the time when reading restarts from the beginning of the CD
or transfers to another CD. This time differs from one device to another. Each disk
needs one secondary storage for the amount of video data that should be transferred
Figure 11: Requestcast Server Architecture in Terms of Multiple Clients

during the restart time. We can calculate this amount of video data $V$. If $r$ denotes the CD-ROM transfer rate and $l$ is the CD-ROM latency, then $V = r \cdot l$.

For this secondary storage, we choose to use ROM chips as there is no need to use an expensive or complicated alternative for the simple task of housing a relatively small amount of data. Any programmable ROM type PROM (Programmable Read-Only Memory), EPROM (Erasable Programmable Read-Only Memory), EEPROM (Electronically Erasable Programmable Read-Only Memory) can be used. ROM starts transmitting data once the end of a CD is reached. The key to this method is that the beginning of each CD is missing the data stored in ROM. Otherwise, there would be a repeat of that data.

Each CD-ROM with its corresponding ROM continuously broadcasts one particular segment of a video in the manner explained by the pyramid broadcast model. As a client’s request reaches the server, the server control switches the client to the video’s channel (Figure 11). The server control is also responsible for switching the client to the subchannel that broadcasts the first segment of the video. At the end of that segment, the user is switched to the subchannel containing the second video segment and so on until all video data is stored at the client side (Figure 12).

Now we need to determine if the required bandwidth of the client channel is realistic. The sustained transfer rate of 27 Mbit/sec is the required bandwidth. This is
well within the scope of the bandwidth provided by cable video transfer protocols [14]. For instance, 8-VSB and 16-QAM protocols provide bandwidth of 32.25 Mbit/sec. An even faster rate of 43 Mbit/sec is provided by 16-VSB. This ensures the feasibility of our server composition and our ability to provide fast forward functionality.

Each CD-ROM with its corresponding ROM continuously broadcasts one particular segment of a video in the manner explained by the pyramid broadcast model. As a client’s request reaches the server, the server control switches the client to the video’s channel (Figure 11). The server control is also responsible for switching the client to the subchannel that broadcasts the first segment of the video. At the end of that segment, the user is switched to the subchannel containing the second video segment and so on until all video data reaches the client (Figure 12).

At this point, we are ready to explain synchronized pyramid broadcasting integration.

In the case of a high bandwidth channel that is being divided into several logical channels, synchronization usually does not require additional arrangements. However, our implementation of requestcasting uses separate CD-ROM drives that broadcast into separate channels. Therefore, synchronization does not come for free. We will concede that synchronizing reading from several CD-ROM drives with the help of
electro-mechanical means (synchronizing start, rotation, et.al) may be difficult or even infeasible. Instead, we introduce the idea of using an intermediate buffer to which streams from different CD-ROM drives are directed. Within a reasonably small period of time after start, streams from all CD-ROM drives will reach the buffer. At this point, data can be sent through.

This is done in an inverse manner to the division of the bandwidth among logical channels. Suppose that data from CD-ROM drives is read in small portions one portion per processor clock cycle. Channel 1 has portions $d_{11}, d_{12}, d_{13}, \ldots$. Channel 2 has portions $d_{21}, d_{22}, d_{23}, \ldots$. Channel $n$ has portions $d_{n1}, d_{n2}, d_{n3}, \ldots$. A wait occurs until the first portions from all channels are in the buffer (Figure 13). Some channels may have data available before others if their respective CD-ROMs are being read from moments sooner. These channels could have multiple portions in the buffer before others have any. Once all channels have at least the first portion in the buffer, we output them in the following order: $d_{11}, d_{21}, d_{31}, \ldots, d_{n1}, d_{12}, d_{22}$, and so forth. Once a portion is output, a new one can be written in its place. This way, the buffer size does not grow infinitely. In addition, we will never run out of data in the buffer since

Figure 13: Buffer Use for Synchronization
the rate of reading data is the same as the rate of outputting data.

We can estimate the size of such a buffer $b$. If $W$ represents the wait between when the first and the last of the channels stores into the buffer and $r$ is the data transfer rate of the CD-ROMs, then $b = W \cdot K \cdot r$. 
Chapter 3

Further Development and
Conclusions

3.1 Further Development

We have explained the relationship between requestcasting components and the integration of synchronized pyramid broadcasting. Analytically, we have demonstrated that requestcasting is feasible and very promising. The next logical step is implementation. Our architecture needs to be further tested once it is fitted with appropriate hardware, firmware and software. We may learn that hard drives work better than CD-ROMs. How requestcasting functions with such real-time issues as clock cycles and understanding how to adjust the model accordingly is vital to requestcasting development.

3.2 Conclusions

So, does requestcasting optimize VOD? Let us judge by the four VOD criteria that we started with:

1. minimal channel bandwidth
2. short client waiting times
3. dependable service to a large number of clients, and

4. full support of VCR functions.

We address how we have been able to satisfy each metric in the same order.

1. Requestcasting, unlike many designs, is capable of functioning with current cable transfer protocols.

2. Waiting times at one minutes or less (depending upon whether the VOD designer selects to optimize bandwidth over waiting time or vice versa) can be achieved.

3. Requestcasting is scalable. The number of clients is unlimited as long as hardware requirements are met. And, unlike conventional broadcast methodologies, viewership can be limited to a target audience.

4. All VCR functions of pause, rewind and fast forward are supportable at client end.

Besides satisfying these criteria, requestcasting also simplifies VOD. In previous designs, completely different methodologies are suggested for popular versus unpopular videos. The VOD designer was left to determine how to manage the less-popular videos with low, no or even negative bandwidth. Although we suggest elongated pyramids of shorter segments and squat pyramids of larger segments for popular and unpopular videos, respectively, both can be managed through requestcasting. It is a uniform solution.
Bibliography


