MASTER

The development of a video server for video-on-demand

Peeters, P.P.A.A.

Award date:
1999

Link to publication

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
Master’s Thesis:

The Development of a Video Server for Video-on-Demand

P.P.A.A. Peeters

Coach: ir. M.J.M van Weert
Supervisor: prof.ir. M.P.J. Stevens
Period: August 1996–August 1997

The Faculty of Electrical Engineering of Eindhoven University of Technology does not accept any responsibility regarding the contents of Master's Theses.
Abstract

This is a master’s thesis on the description of a video server for Video-on-Demand (VoD) services. A customer using a VoD service, selects a video from a large repository of videos and is able to interactively change the playout sequence by using the VCR functions.

VoD may have a large potential growth market, but there are still some improvements needed before it will become commercially available. A performance objective of developing a VoD service is therefore to support the maximum number of concurrent users with an acceptable quality of service.

The description of the VoD system is done by dividing the VoD into the three parts called the video server, network and client. Each of these parts is worked out separately in this thesis. Because emphasis is laid here on the video server part, the server is divided in a subsystem where the videos are stored and retrieved, a processing part for the retrieved video frames and client interactions, and a networking part to send the video frames over the network.

A very efficient way to support more concurrent accesses is the capability of the server and network to share resources between users in order to lower the total number of concurrent streams. A multicasting scheme is presented to provide multiple users with the same stream.

In case a stream cannot be shared between users, because some users perform VCR actions, bandwidth has to be reserved in order overcome the extra need for bandwidth. Reserving resources like this means that a trade-off is made between the maximum number of concurrent users allowed at the server and the quality of service delivered. Mechanisms are presented which minimize the amount of reserved resources needed and still meeting the requirements.

A scheme is presented for the storage and retrieval of video frames in an efficient way such that the number of users in a system with VCR functions is just a little less than in a system with just PLAY, PAUSE and STOP.
Contents

1 Introduction 1

2 The Elements in a VoD System 3
  2.1 Storage Subsystem 4
  2.2 Memory Subsystem and Processing Unit 4
  2.3 Network Subsystem 4
  2.4 Delivery Network 4
  2.5 Client Side 5
  2.6 Further On 6

3 Storage Subsystem 7
  3.1 Hierarchical Storage 7
  3.2 Striping Schemes 8
    3.2.1 Logical Volume Striping 8
    3.2.2 Application Level Striping 9

4 Memory Subsystem and Processing Unit 10
  4.1 Buffering 10
  4.2 Supporting Different Video Types 11
  4.3 Handling Disk Failures 11
  4.4 Management 12

5 Scheduling 13
  5.1 Client Behaviour 13
  5.2 User Scheduling Policies 14
  5.3 Disk Scheduling 16

6 Multicasting in a VoD System 19
  6.1 The Multicasting Scheme 19
  6.2 Bridging and Piggybacking 20
  6.3 VCR Functions 21

7 Synchronization 22
  7.1 A Scheme for Synchronizing Continuous Media Streams 23
  7.2 Synchronization With VCR Functions 24
List of Figures

2.1 The architecture of a VoD system ........................................ 3
2.2 Protocol stack. ............................................................. 5

3.1 A request accessing the disk array .................................... 8

4.1 Buffering of video streams. ............................................ 10

5.1 Dynamic classification of cold and hot videos. ...................... 15
5.2 Timeline of the disk requests being served. ......................... 16
5.3 The impact of differences in the disk throughput. ................ 17

7.1 Readout of buffered frames. ........................................... 23
7.2 Different possible states of audio and video. ....................... 25

8.1 Model of a user performing a fast playout ........................ 29

9.1 Temporal order in which the frames are displayed. ................ 32
9.2 Storage order of the frames in a block. ............................. 33
9.3 Normal playout of the video blocks. ................................ 34
9.4 Fast playout of the video blocks with frame skipping. .......... 35
9.5 Efficient storage order of the frames in a block ................. 35
9.6 Schematic representation of the frame length distribution in a block. 36
9.7 Schematic representation of the read time distribution in a block. 36

10.1 Schematic representation of a client connected to an access provider. 40
10.2 Position of the set-top box in the system. ......................... 41
10.3 The flow of the Java program. ....................................... 42

A.1 MPEG frame dependencies. ........................................... 45
A.2 MPEG layers. ............................................................ 46
Chapter 1

Introduction

Video-on-Demand (VoD) is a service with a large potential growth market. Developers of multimedia applications have great expectations about the future of VoD in particular and multimedia in general, but there are however some infrastructural improvements to be made before VoD is possible over networks, especially on a wide area scale. Improvements have also to be made on both end systems to make the delivery of these services cost-effective for common households.

VoD is an interactive, distributed, real-time, multimedia service and is interactive in the sense that a consumer can select a video from a large set of videos stored at the video server and perform the same functions as on a traditional VCR.

The term distributed is used here, because there is some networking involved to send the video to the customer in the first place and to make the service interactive. Besides that, video servers are distributed to improve the total number of concurrent accesses.

VoD is inherent a multimedia service, because both video and audio are used. Other types of media can coexist when a video is, for instance, embedded in web page with text and images. Because of the stringent temporal restrictions of video and audio, the VoD service is added the term real-time.

A VoD service allows geographically distributed users to interactively access video files from remote VoD servers at any time and have flexible control over the playout of the video with VCR-like functions.

In this report, a VoD system is presented which has to make VoD possible to a maximum number of concurrent users given a certain quality of the service to be provided. Maximizing the amount of users reduces the costs per user and can make VoD possible commercially.

The VoD system can roughly be broken down into the five parts of which an overview is given in chapter 2. Three of these parts are subsystems of the video server are named the storage subsystem, memory and processing subsystem and network subsystem. The storage subsystem and memory subsystem are described in respectively chapter 3 and 4.

The network subsystem, delivery network and the client side are viewed in chapter 2. Chapters 5–7 are devoted to optimizing the number of concurrent users at the video server, which
is extremely important if VoD has to become commercially available for use in a domestic environment.

In chapter 10 finally, predictions are made about the development and organization of distributed multimedia services and how the customers experience them. And in chapter 11 conclusions are drawn from previous chapters and recommendations are given about the implementation of a video server.
Chapter 2

The Elements in a VoD System

The VoD system presented here is divided into five parts, each having their own requirements and properties. The main objective common for all parts in the development of a VoD system is to provide a service with an acceptable quality of service to as many concurrent users as possible. In this chapter, a short description is given to all parts in the VoD system represented in figure 2.1.

![Figure 2.1: The architecture of a VoD system](image)

The five parts reviewed in figure 2.1 are the STORAGE SUBSYSTEM, MEMORY SUBSYSTEM AND PROCESSING UNIT, NETWORK SUBSYSTEM, DELIVERY NETWORK and the CLIENT SIDE.
2.1 Storage Subsystem

The storage subsystem is the location where the video files of the video server are stored. Different types of media can be used for this purpose. The choice for a storage medium depends on the bandwidth needed, the user base and the price of the storage medium. As will later be explained, the best way is to store videos is to use a hierarchical storage system.

Besides the different storage media, there are also different storage techniques possible. A key issue is to store the videos in such a way that the total number of video streams at the server is optimized. A storage technique which can establish this is striping, which will be explained in chapter 3. Each video is divided in blocks of several consecutive video frames and the blocks are stored on multiple disks. The order in which the frames are stored within a block is worked out in chapter 9.2.

2.2 Memory Subsystem and Processing Unit

The memory subsystem and processing unit is the main part of the video server, where storage and network are connected. Here, the retrieved video frames from the storage are buffered to reduce the jitter in the system. The two different buffering schemes presented are the dedicated buffering scheme and the shared buffer pool scheme.

Within the processing subsystem, the scheduling policies are implemented to define the order for resource accesses to the storage media. A multicasting scheme is used to share streams between users to increase the capacity of the system.

This part of the video server also negotiates with the client about the QoS delivered and performs the incoming VCR controls from the client. Furthermore, there are management functions built in to monitor the access distribution and the available resources.

2.3 Network Subsystem

The network subsystem is the part of the network that resides in the video server. The video frames are packetized and prepared to be pumped into the network and sent to the client side of the system. The network subsystem consists of dedicated hardware to perform this task.

Together with the memory subsystem and processing unit, this part is often called the video pump. The network subsystem is not worked out in this report, because it depends on the architecture and topologies used.

To improve the capacity of the system, servers can be connected to each other through a local area network.

2.4 Delivery Network

This is not part of the video server, but is important to the VoD system. The delivery network connects the servers with the clients and consists of switches and the lower protocol layers at the end systems.
The delivery network can be homogeneous or heterogeneous. In the former case no internetworking is needed between server and client and it is easier to make a deal for a certain QoS. In the latter case some translations have to be made in the lower layers of the protocol stack. The latency of the network is determined by the distance between the client and the server and the protocols used.

For the network protocols there are many choices one can make, depending on the physical transport medium and the transmission rate needed. In any way, it is recommended to take the protocol stack presented in figure 2.2.

In this figure internet protocol (IP) is used as the network protocol, because it is the network protocol that is used the most at the moment. Above IP, the first end-to-end layer is the transport layer. Different transport layer protocols will coexist at the server and the client, because the most widespread used Transmit Control Protocol, has too much processing overhead, like error correction, which is not needed for continuous media like audio and video. TCP will therefore be used to transmit text and images, while User Datagram Protocol or Real-Time Transfer Protocol will be used to transmit audio and video and the control signals from the user.

Asynchronous Transfer Mode and ATM Adaptation Layer 5 are added because ATM is able to integrate different types of traffic, like video, audio and text into one network and has a high throughput in the order of 155Mbps for users and speeds in Gbps for backbones.

2.5 Client Side

The users at the client side of the system have a playback device that is dedicated for retrieving video or it may be an application running on a computer. A process has been started to develop a device that is has the service integrating capabilities of the computer and the complexity and interface of a television.

This device would however be expensive and therefore much research and development is done on the so called Set-Top Boxes (STB), which are boxes placed between the television and the network, and extend the functionality of the television.
When a video arrives at the client side, the video is first buffered in order to keep the variance in network latency low and to synchronize multiple streams if needed. In the next step the stream is decoded and displayed. The user is able to control the stream with VCR functions. The experiences of the client regarding future broadband network services and the organization behind it, is described in chapter 10. Requirements of the client side for in particular VoD are listed below.

- User friendly interfaces
- Cost-effectiveness
- Quality of Service

Quality of service is indicated by the throughput, delay and jitter and the client side can only affect the jitter. The delay and throughput are mostly determined by the server and the network.

2.6 Further On

In the next chapters, the parts within the video server will be worked out. The storage subsystem is described in chapter 3, where the different media are compared and a the way to store the data. In chapter 9, a finer granularity is presented of how to store the video frames.

Chapter 4 is devoted to the memory subsystem and processing unit and more details are worked out in chapters 5–7.
Chapter 3

Storage Subsystem

The storage subsystem is the part of the video server where the videos are stored. Different storage media can be used for this purpose. RAM has the highest throughput and can therefore provide services to far more customers than an optical disk could. The disadvantage of using RAM for the storage is that is the enormous increase in storage costs. According to Moore’s Law described in appendix B.2, the capacity of RAM doubles every two years and this will lead to a reduction in the costs.

In the next section, a storage scheme is proposed to store the videos as cost effective as possible and thereby optimizing the total number of streams.

3.1 Hierarchical Storage

When storing videos in an archive, the most cost effective way is to use tapes or optical disks. One hour of MPEG-2 encoded video at 4 Mbps occupies an amount of 1.8 GB of storage, so much storage capacity is needed. A disadvantage is however that the throughput is about 10 Mbps and at most 2 clients can watch a video simultaneously. This makes the storage expensive regarding the number of streams. Furthermore, the seek times are typically about 100 times longer than the seek times of a hard disk, so random access within movies have a poor performance.

To make the system both cost effective and capable to service more users, a hierarchical storage scheme is proposed, like in [CKY95], [CKY94], [BR96] and [RF96]. In this scheme, a part of the videos is stored on a hard disk and the rest is stored on the so-called tertiary storage devices like optical disks. The idea behind the hierarchical storage scheme is that videos that are not requested much are stored on tertiary storage devices in order to reduce the storage costs. These videos are called the cold videos and most of the time they are not requested or by one user at a time. If requested more, the video is regarded as a hot video and then the video will be copied from the tertiary storage device to a secondary device like a hard disk.

Using Zipf’s Law from appendix B.1, the set of videos can be separated into hot videos and cold videos in an effective way. Adding more videos to the set of popular videos will increase the need for secondary storage capacity but will release the tertiary devices from a too high load.
The primary storage (RAM) is used to buffer small amounts (in order of a few seconds) of video to overcome the jitter at the server and in the network. Furthermore, it can be used for bridging as will be explained in section 6.2.

To further improve the throughput of the system, the hard disks are connected parallel to each other and videos are placed on multiple parallel disks as explained in the next section.

3.2 Striping Schemes

In the storage subsystem, a video file is distributed on multiple disk drives or disk arrays in special ways to employ the flexibility and aggregated I/O bandwidth of the storage devices. In [HLL+95] the effect of data striping schemes on the performance of a video server is already studied. The two striping scheme introduced there are logical volume striping and application level striping. The following sections give an overview of the working of both striping schemes and will show why application level striping is better.

3.2.1 Logical Volume Striping

A logical volume is a storage entity which behaves like a disk partition, but its storage may span several physical disks. The configuration of a logical volume is defined by two parameters. The striping size specifies the number of physical storage devices used in the logical volume and determines the maximum number of subrequests that can be placed simultaneously. The step size specifies the amount of data that is transferred from one device to the requesting user.

In the logical volume striping scheme, a contiguous stream of data is divided into blocks of data and distributed to the disks in a round-robin style. Each request is translated into a number of subrequests depending on the bandwidth needed. Say that $B_R$ is the bandwidth needed for the main request and each subrequest provides $B_S$, then the following relation is valid for the system in figure 3.1

$$n = \left\lfloor \frac{B_R}{B_S} \right\rfloor$$

Figure 3.1: A request accessing the disk array
where \( n \) is the minimum number of subrequests needed to allocate the bandwidth requested. Assume \( B_R = n \cdot B_S \) and that \( n \leq M \). Since \( M - n \) volumes are idle, the amount of bandwidth not requested for, is then equal to \((M-n) \cdot B_S\). Using the logical volume striping will lead to inefficient bandwidth usage.

It is however possible to store the videos such that \( M-n = 0 \), but then the amount of retrieved data is variable regarding the playout time. There will also be a larger overhead due to the higher seek latencies.

### 3.2.2 Application Level Striping

The logical volume striping provides a service abstraction for users and handles the storing and retrieval of data. On the contrary, the application level striping scheme requires applications to handle the storing and retrieval themselves. This means that the applications must know where to retrieve and store the data.

In contrast to logical volume striping, users access one block of data from one storage device at the time, except when performing a fast playout session (see chapters 8 and 9). Accessing one device at a time makes the scheduling process more efficient, since the scheduler optimizes the order of accesses to one disk, such that seek latencies are reduced.
Chapter 4

Memory Subsystem and Processing Unit

This part of the video server receives the initial and control requests from the clients and allocates the resources needed to provide the QoS needed. VCR actions from the clients are processed and translated into scheduled requests to the storage subsystem. The scheduling process is not worked out here, but in chapter 5. Data from the storage subsystem is buffered first to reduce the jitter of the system.

4.1 Buffering

Buffering within the memory and processing subsystem is done according to the schematic representation in figure 4.1. Video blocks from the storage devices are added the information needed in the network and are then stored in the buffer and read out according to a leaky bucket scheme. The frames from the buffer are pumped into the network through a network interface card (NIC).

![Buffering of video streams.](image)

For the buffering scheme two solutions are possible: a dedicated buffer scheme or a shared buffer scheme. When considering the choice between a dedicated buffer scheme or a shared buffer scheme a trade-off exists between the amount of memory needed and the complexity of the buffering scheme.

In a dedicated buffer scheme every stream is assigned a private buffer space with a size...
depending on the video characteristics and the QoS the system has to provide. This scheme is less complex, but is not very efficient when MPEG frames are used. The size of an MPEG frame differs much, depending on the frame type. However, in a scheme where blocks of frames are used as the elementary unit of data, the dedicated buffer scheme is much efficienter. This higher efficiency is the result of the blocks having a minor variation in size.

To indicate the cost effectiveness of reducing the total amount of buffer space needed, the expected buffer space totally needed is calculated in the following. Given a system which can service 1000 MPEG-2 streams simultaneously, and each MPEG-2 video needs a bandwidth of 4 Mbps. If the video is stored in blocks of \( \frac{1}{2} \) sec and only one whole block is buffered at any time, then a buffer space of totally 250 MB is needed.

In the shared buffer scheme memory is saved by dynamically allocating buffers to a video stream from a pool of shared buffers. The varying buffer requirement of a video stream is adaptively allocated during the entire playout session. The penalty of compensating the variation between different streams is the complexity of the shared buffer system. This complexity regarding the implementation is that a manager is needed to allocate the needed memory and deallocate is afterwards. In [DHL96] the question arises whether the dynamic allocation and deallocation of resources introduces to much overhead.

The data storage organization presented in chapter 9 stores video frames in blocks of a constant length in time. The advantage is that when the buffer is taken long enough, the variation in the buffer space needed is reduced, because the bandwidth needed will be averaged.

### 4.2 Supporting Different Video Types

The problem of supporting concurrent accesses within a video server becomes much more complicated when considered that the block sizes of the video types may differ from each other. The variations of the different video types are characterised by the following set of parameters.

- **Frame size**, which is denoted as the typical size of a frame of a typical video. This value depends on the compression gain and the resolution of the video. In case of MPEG, the frame size depends on the frame type.

- **Display rate** indicates how many frames a user expects to see in a certain time interval.

- **Block size** is the number of frames within a block.

All video types are stored here in blocks of \( \frac{1}{2} \) sec. Buffers storing HDTV will simply be larger than the buffers for PAL.

### 4.3 Handling Disk Failures

Striping the data accross different disks certainly improves the performance of the system, but the penalty for this performance improvement is paid when a disk-failure occurs. A part of the information is gone, and the system will have to operate in a degraded mode. The failed needs to be rebuilt by sending the appropriate video blocks from the archive to the disk. This needs to be done as fast and as soon as possible because when more failures occur, the system would become out of service.
4.4 Management

The management of the video server is also done in this part. Management can be divided into system information management, like resource monitoring and user information used for billing and determining the popularity of a certain video and the access distribution.
Chapter 5

Scheduling

A scheduler defines a partial order for resource accesses according to a certain policy in order to improve the performance of the system. Performance criteria are the optimality and predictability.

Predictability deals with the access distribution of the users and their behaviour during the playout session regarding the use of VCR functions. Using the knowledge of the access distribution and behaviour, the order of the resource accesses is determined in order to improve the performance.

5.1 Client Behaviour

Access Distribution

Clients select a video from the available videos according to an access distribution. This access distribution is non-uniform, so there is a difference between videos in the frequency a particular video is requested. These access frequencies can be characterized by a distribution according to Zipf’s Law with the parameter $\theta$ being equal to 0.271. More about Zipf’s Law can be found in appendix B.1.

The power of Zipf’s Law is that the access pattern can partly be predicted and the server can adapt itself to the predicted situation by performing the actions listed below. The result is that the server can handle more users with a better QoS or the server can do the same thing in a more cost-effective way.

- Providing the popular videos with a better storage medium.
- Replicating the video or caching it on several locations near clients.
- Providing a better QoS for popular videos at the cost of the QoS for the not popular videos.

A minor problem with Zipf’s Law is that it is a formula that needs some interpretation. The use of this law depends on how homogeneous the videos are and the set of customers. The access pattern to a video server may change reasonably within a day depending on the video contents and again the customers.
A Hollywood movie that has just been released would be requested more frequently and is therefore seen as popular. Zipf's Law works rather good for this type of movies, as it does for colleges at a university. Subjects that are lectured at the moment will be requested much more than courses that are lectured in a few months later. Zipf's Law is not applicable to for instance doctors in a hospital. For professional markets there is a more uniform access distribution.

Reneging Probability Versus Time
During a high server or network load and when users are batched, the system may react slow on both initial requests and intermediate requests. Users do not like to wait too long and the result of a too long waiting time is usually that users leave the system.

It is not a surprise that the reneging probability increases with the waiting time and the reneging probability function depends on what is done for the client in exchange. For instance, a clock indicating how long it takes until the video starts will help a lot.

For the users, there is also a big difference between initial waiting times or waiting for response after a VCR action. Generally, users are willing to wait longer before starting a service than in case of a resume or a VCR function.

Improving the responsiveness of the server will limit its performance, because more bandwidth has to be reserved for serving users requesting to perform VCR action. In case of a policy where clients are batched, the limitation is that the number of clients using the same stream decreases.

Behaviour During Playout
The behaviour during playout will not affect the choice for a scheduling policy, only the way a user resumes from PAUSE or STOP, because these channels may be in use by an active stream. How many times a user performs a VCR actions and how long will definitely affect the amount of bandwidth to reserve.

5.2 User Scheduling Policies
In this section some policies are proposed for scheduling initial requests from users. These policies are derived from [Chu96]. Even if there is a high load on the system, users do not want to wait long to access a video. In the following different policies are given that wish to optimize the responsiveness of the system.

FCFS Policy
The First Come First Served policy queues all requests for all videos into a single request queue. Once a channel becomes available, the scheduling policy selects the video requested by the client at the front of the request queue.

Any requests for the same video are also served by the same stream using the multicast facility. This policy seams to be a fair one, since the clients are served in order of their requests and the users requesting the same video as the first arrived client are just lucky.
CHAPTER 5. SCHEDULING

FCFS-n Policy

This policy assigns dedicated streams for the $n$ most popular videos, called hot videos. After every predetermined time interval a new playback stream is started. This time interval is called batching window. A further improvement is made by varying the size of the batching depending on the popularity of the video and the expected arrival rate of a client for this video.

A request for a popular video has therefore a maximum waiting time equal to the length of the batching window. The remaining cold videos are served according to the FCFS policy over the remaining bandwidth. The FCFS-n policy is a more unfair policy to the users who request a cold video, but users will mostly take this into account.

Another policy based on the batching window principle is the forced wait policy. In the forced wait policy, the first arriving requests have to wait a minimum time when a channel becomes free. This is done in order to maximize the number of requests served at the same time. The maximum waiting time of the forced wait policy depends on the popularity of the video and on the reneging probability.

MQL Policy

The Maximum Queue Length policy maintains a separate queue for each requested video, and queues each request into its corresponding queue. The advantage is that the most popular video at the moment is served first and therefore more users are served within a shorter time. A drawback is that requests for a cold video are served after a longer time and in some cases these requests are even not served at all!

A Combination

The previous policies were taken from [Chu96] and take a rather strict approach to what cold videos are and which videos are hot. Users viewing a cold video may starve and lack any service if there is a high load at the server. In figure 5.1 a schematic representation is given of a cold video entering the hot video set. The explanation follows next.

\[
\begin{align*}
\text{COLD} & \xrightarrow{\tau_{\text{threshold}}} \text{ELIGIBLE} \\
& \xrightarrow{\tau_{\text{threshold}}} \text{HOT}
\end{align*}
\]

Figure 5.1: Dynamic classification of cold and hot videos.

The batching scheme, called wait tolerance, makes effective use of the viewer wait tolerance. Videos are classified in hot videos and cold videos. A video is considered hot when one the following criteria are met.
CHAPTER 5. SCHEDULING

- It is a hot video.
- Its queue has more than one request.
- Its queue has a single request that exceeded the batch threshold.

The last criterium listed is depicted in figure 5.1. After a time $\tau_{\text{thresh}}$ the cold video enters the eligible subset of the hot videos only if this subset is empty. The eligible subset consists of the hot videos which will exceed the batch threshold if not serviced as soon as possible.

5.3 Disk Scheduling

The principle of disk scheduling is to put the requests entering the disk in a certain order such that the performance is maximized. The requests dealt with are requests for the retrieval of certain blocks. Maximizing $n$ in figure 5.2 (taken from [CKLV95]) and making sure that the blocks are retrieved within $T_{\text{period}}$ maximizes the performance.

![Figure 5.2: Timeline of the disk requests being served.](image)

The numbers 1...$n$ are the requests for the retrieval of video blocks with corresponding retrieval time $t_i$ and seek latency $\tau_{s_i}$. The slot time of the system is denoted with $T_{\text{period}}$. Before the end of $T_{\text{period}}$ the requests arrive for the next period and are put in order. The maximum number of requests is such that

$$T_{\text{period}} = \sum_{i=1}^{n} (t_i + \tau_{s_i}) + \tau_{\ell}$$

holds and there is some $\tau_{\ell}$ time left over, $\tau_{\ell} \geq 0$.

The conventional ways to perform disk scheduling for continuous media streams are round-robin, SCAN and EDF of which the combination of SCAN-EDF the highest throughput yields. In the following, the disk scheduling policies are described shortly.

**Round-Robin**

Using the round-robin scheduling scheme, requests are served in order of their arrival time. The round-robin scheduling scheme has the lowest throughput, because each client is serviced in a fixed order and henceforth the disk head sweeps through the disk in search for the requested blocks. This introduces unnecessary seek latencies. The throughput would improve greatly if the head did not to go back and forth between requests.
SCAN

In the SCAN scheme the head also moves back and forth but only from end to end. Requested blocks are read as the head passes the block to read. By placing the requests in order of their distance from the head, the total distance that the disk head must travel is reduced and thereby overall seek latencies.

Earliest Deadline First

With the EDF scheme, each request is given a deadline for retrieval, and the request with the earliest deadline for retrieval. The request with the earliest deadline is serviced first to minimize the waiting times. Users are allowed to enter the system until the fraction of the deadlines that cannot be met becomes too high.

SCAN-EDF

SCAN-EDF is performed in case requests shared the same deadlines. This scheme behaves almost similar as the SCAN scheme when for all videos the same storage scheme is used.

Each MPEG-2 encoded block is stored in blocks of about $\frac{1}{3}$ sec. The time to read a block is done in a far shorter time, but the deadline for the next block is $\frac{1}{3}$ sec later as visualized in figure 5.3, wherein also the impact of the I/O throughput on the number of users allowed in the system is showed.

![Diagram](image)

Figure 5.3: The impact of differences in the disk throughput.

The area within a drawn block denotes the length of a block in unity of bits. In figure 5.3 is obvious that FAST RETRIEVAL RATE can service more requests before $T_{period}$ is reached than SLOW RETRIEVAL RATE. In case the time to read a block equals $T_{period}$, then only one request can be served during that period and is CONTINUOUSLY RETRIEVED.
From figure 5.3 can also be derived what happens if another video type is used. Take for instance the situation that HDTV with a frame rate of 60 fps is used. Then two scenario's are possible.

In the first scenario the blocks remain to be of length equal to $\frac{1}{2}$ sec. The number of frames then stored in a block is twice as large and the block will therefore be larger. The retrieval rate $r$ from figure 5.3 won't change, so less requests are possible within $T_{\text{period}}$.

In the second case the blocks will remain about the same size measured in bits. The number of frames stored in a block then decreases and, counted in time, less video is stored. When less video is played in $T_{\text{period}}$ then $T_{\text{period}}$ has to decrease also to maintain the frame rate of 60 fps.

From the previous can be concluded that videos of different types will affect the system either in the deadline or the number of requests possible or the size of the blocks.
Chapter 6

Multicasting in a VoD System

Video-on-Demand systems which provide a guaranteed fast response on interactions from the users typically allocate resources in the system dedicated for each user. The resource usage in the system is henceforth proportional to the number of clients served. To make the system more scalable regarding the number of concurrent user possible, resources could be shared by multiple users. Multicasting means that multiple requests for the same video are grouped and share the same video stream. The difference between multicasting and broadcasting is that when the broadcasting scheme is used, all users receive all channels.

When users are served simultaneously and thereby allocating the same resources, the responsiveness of the system could be affected in a negative manner. The illusion can, however, be kept up that a user is allocated a dedicated stream. This is done by making use of the multicasting scheme presented in the next section.

6.1 The Multicasting Scheme

The advantage of using multicasting is the increase in the number of simultaneous users the system can serve simultaneously. Increasing the number of users in this ways means that a trade-off is made against an increased system complexity or a degradation of the delivered service.

The multicast scheme will only be used for the so called hot videos because for these type of videos there is some gain expected \(^1\). The first action to be made is to change the continuous time into a discrete one. Continuous time is divided into equal length time slots in which customer requests will be grouped and then scheduled as described in section 5.2. Requests that arrive in the same time slot are serviced together at the end of the time slot.

Using this discretization, the initial delay of the system becomes higher. However, users do typically not mind an extra initial delay as long as this delay is bounded and the upper bound is known by the users. This delay is in section 5.2 denoted as forced wait policy.

\(^1\)From chapter 5 we learned that a cold video having multiple requests simultaneously pending would be regarded as a hot video.
6.2 Bridging and Piggybacking

The multicasting scheme allocates a single stream for requests for the same resources arriving in the same time slot. A user in a stream used by multiple users will leave that stream if a VCR action is performed.

After resuming the normal playout session again, there are different policies to allocate resources. Since the storage is regarded as the main bottleneck of the system, the most likely situation of a resume policy is that no extra streams are needed to retrieve the data from the storage medium.

A method called bridging is used to send these requests to the RAM instead to a secondary or tertiary storage device. The principle of bridging is as follows.

**Bridging**

If a video is resumed, first a check is done whether there is another stream at a position within the video very near to the position of the resumed stream. If so, the resumed stream does not need a new stream. This trick can only be used if the other stream is close enough in time such that the resuming process will not take too long.

It is however most probable that a new stream is needed. If another stream is close, but too far away to make the users share streams, the difference between the streams is copied into RAM. This principle is called bridging. The stream leading in time is allowed to read from the storage devices and the stream lagging behind will have to read from memory.

Bridging is especially useful for hot videos whose batching window is small. A smaller batching window results in a more economic memory usage, because the time differences are shorter. Bridging will therefore only be used for hot videos only.

**Piggybacking**

The disadvantage of bridging is that it is a memory intensive operation and will only be used if the load on the storage devices is too high. A method to bring down the amount of memory usage is called adaptive rate piggybacking.

The principle of adaptive rate piggybacking is to bring down the distance between streams in time and henceforth the memory usage is brought down. Bringing down the distance is done by altering the frame rate of the video a little bit. A user watching a video does not notice differences in the frame rate as long as the relative rate difference is below the 5% of the normal rate.

Users are more perceptive to disturbances in the audio than to video disturbances and therefore the audio rate will not adapt to the 5% difference in the video frame rate. Because this will lead to synchronization problems between audio and video, after a minimum time some audio frames are dropped.

Adaptive piggybacking is used until the videos are near enough to perform only bridging or when there is no difference between the streams anymore.
6.3 VCR Functions

For the implementation of VCR functions, there are two possibilities. The choice of a certain possibility determines whether the VoD service is a TVoD service or a NVoD service. More about TVoD and NVoD is explained in appendix D. The two implementations are denoted here as resource reservation and stream switching and are explained in the following.

Resource Reservation

Using resource reservation, a stream in a multicast group allocates a new stream that is set aside by the server. The channels that are set aside are called contingency channels. More about the use of contingency channels can be found in [DSST95]. In [DSST95] a client who pauses returns the bandwidth to the system, like the RD scheme in chapter 8. Only the scheme presented in [DSST95] does not provide the fast playout functionalities.

If all contingency channels are allocated in [DSST95] then the system starts allocating the free channels which are normally used for normal playout. The clients requesting to enter the normal playout mode coming from the contingency channels (and free channels) have priority over new client requests.

In the scheme presented in chapter 8, it is not taken into account that users may share the same stream and therefore no contingency channels are reserved, but only free channels.

Stream Switching

The use of stream switching is a typical NVoD implementation. Video $i$ is started every interval of time $t_i$. Requests for video $i$ are assembled in one interval and share the same stream in the next interval.

The use of VCR actions can only be done in timelengths of $n \cdot t_i$ where $n \in \mathbb{N}$ is done by switching to a different group.
Chapter 7

Synchronization

One of the major problems associated with multimedia is the synchronisation of continuous media streams. The far most famous example for continuous media streams that need to be synchronized is the combination of audio and video. This type of synchronization is also known as lip-synch. The lip-synch problem is an inter-stream synchronization problem.

The continuous stream synchronization problem is a combination of two problems, intra-stream synchronization and inter-stream synchronization, already mentioned. Both these continuous stream synchronization problems are correlated and cannot be solved separately, partly because an extra requirement is that for continuous media services, the end-to-end delay must be kept low. Especially for the continuous media with user interactivity.

Intra-Stream Synchronization

During the encoding process of the videos, each frame is added a time-stamp to reconstruct the original sequence of video frames. Since videos have a constant frame rate, this time-stamp is used as the time base for the video.

As video frames are displayed in a linear time, the time-stamps are also linear with the time. Between adding time-stamp and detecting them, there will always be a difference in time, as in the case of sending a video frame and receiving it. This delay is not a problem if the mean delay is not to high. The main problem is the variance in the delay. For intra-stream synchronization this variance in delay can cause the derived time to be non-linear and henceforth the frames are not played at a constant speed.

Inter-Stream Synchronisation

Inter-stream synchronization is the establishment of the timing relationship between multiple media stream. Assumed here is that this relationship between the streams is known. In VoD services, inter-stream synchronisation is almost always between the video and the associated sound, called lip-synch.

Synchronisation of Continuous Media Streams

Synchronisation of continuous media streams deals with both intra- and inter-stream synchronization and with mean system latency control in addition. In the next section, a scheme
is developed which takes all relevant issues into account.

7.1 A Scheme for Synchronizing Continuous Media Streams

To smooth out the variances in the latency, the video and audio frames are buffered in both the client and the server. A simple rule is that the longer the buffer at the client, the better the synchronization works. There are however two reason not to provide the client side with a too large buffer.

The first reason is a financial one. The more video is stored in the memory at the side of the client, the more expensive the equipment costs. Second, the large buffers at the client side introduce a higher latency from source to display, because the buffer has to be filled first. Furthermore, frames arriving at the buffer will have to wait for the frames in the buffer to be processed. In an interactive multimedia system, a higher latency results in a system that reacts slow on user interactions. The system has become less reactive.

In the scheme introduced here, smaller buffers are used and these smaller buffers always have to be partly filled with frames. At startup, the buffers are filled before reading them. Users do not mind this extra startup latency, especially when this latency is not longer a few seconds.

Figure 7.1 explains the working of the buffering mechanism. Frames are read according to the leaky bucket principle. The buffer is filled with frames at a speed with some variations and emptied at a rather constant rate.

![STREAM IN](STREAM IN) STREAM OUT

Figure 7.1: Readout of buffered frames.

To make sure that the frames are read at a constant rate and to keep audio and video synchronized, the filled level of the buffers have to be above a certain minimum. When reaching the point denoted with MIN, the client sends a message to the server to increase the rate in which the frames are transmitted or skip a few frames. Below the minimum point it is still possible to overcome the jitter and the synchronization, but it will be more difficult. A message is send early enough, because a high load in the system is mostly the cause of a buffer being emptied.

The MAX point in figure 7.1 is added to notify the server early enough that the buffers will be overloaded.

If the load on the system is high or simply not enough resources are left in the system, the buffers would soon be emptied by the reading process of the client. This unwanted situation can be prevented by slowing down the reading process of the client. In practice this is done
by lowering the frame rate of the video. This has to be done such that the client would not notice this. Therefore, the frame rate is allowed to be changed within a 5% band around the normal rate.

The proposed mechanism of adapting the frame rate will not prevent clients from being blocked if the load at the system is high, but with a proper access control on both the network and the server it should work in most cases.

So far the intra-frame synchronization is solved, as long as the buffers are not completely emptied. Inter-frame synchronization is done by checking the time-stamps. The difference between the time-stamps of the different streams is upper bounded. In case of lip-synch, the difference between audio and video is not noticed by the users if the difference is below 80ms.

Because users are more perceptive for audio disturbances than for video disturbances, the audio and video streams are kept in synchronization by adapting frame speed for video. The worst case occurs when audio leads before video. The video then has to catch up with the audio by increasing the frame speed. This can be a problem, because the reason that the video lags behind the audio is possibly due to a high load in the system. A high load will mainly attack video frames, because video needs a significantly higher bandwidth than audio frames. If it is not possible for the video to catch up with the audio, then frames are simply skipped at the side of the server.

When using an MPEG format then the skipping process is preferably performed on B-frames since the decoding of the MPEG frames is still possible then. Throwing a frame away is only done when the bottleneck is the processing power at the client side.

If the problem is the load in the network or at the server, then frame skipping is not possible, because a block of frames is regarded as the elementary unit of a video stream. However, it is possible to skip B-frames, but the frame rate will not be affected by this. As will be cleared in chapter 9 only the bandwidth usage is lowered. To increase the frame rate, either entire blocks will have to be skipped or multiple subrequests have to be made \(^1\). Therefore, the skipping process must affect an entire block, which will be detected by a user. The viewer of the video will see a discontinuity in the playout, but this is allowed only if it does not happen too often. Audio is synchronized by skipping consecutive audio frames. This is percepted by the users as a click.

### 7.2 Synchronization With VCR Functions

The probability that video and audio get out of synchronization is very high during a fast playout session. Partly because frames are skipped and inherently time-stamps, too. There is however a scenario possible where this time-stamp information could be recovered, but there are some strong arguments in favor of not sending the audio at all.

- Audio played out at a higher speed is not very sensible, only audible.
- Conventional VCRs do not process sensible sound either.

\(^1\)Multiple subrequests are used for performing fast playout. If this means an increase in bandwidth usage then this will not be a solution. In case of a high load, the bandwidth has to be reduced.
• The bandwidth requirements are lower.
• Less complexity at the side of the client.

Using the arguments above, a scheme could be developed for the synchronization between audio and video like presented in figure 7.2. In this figure the transition from PLAY to FAST PLAY and back must always pass through STOP. The audio does not have a FAST PLAY state, but stops as the video enters FAST PLAY. After the video returns to the STOP state the audio enters RAN to get to the point in the audio sequence equivalent to the position taken in the video sequence.

![Figure 7.2: Different possible states of audio and video.](image)

A very short time in the STOP state, both the video and audio automatically enter PLAY. The STOP state is added, because the location within the audio sequence is known after the FAST PLAY session has ended.
Chapter 8

Providing Fast Playout Functionalities

Although much research is done on VCR capabilities of video servers, very little is done about the fast playout of a video stream. The main problem of fast playout is the increase in bandwidth needed by a single user. How much bandwidth is needed depends on the way the storing and retrieval process is done, but that is explained in chapter 9.

An extra problem occurs when users are batched. The case that multiple users share a single stream is worked out in chapter 6. In this chapter every user will have a dedicated stream.

The main goal of this chapter is to find a bandwidth allocation policy which provides an effective fast playout service. This policy has to maximize the number of users given the number of users in the system and their behaviour. The major constraint here is the guaranteed quality of service provided to the users in terms of bandwidth and responsiveness.

8.1 Problems with VCR Functions

In addition to the STOP, PAUSE and PLAY functionalities, a desirable feature in all VoD applications is the capability of viewing a video stream at a higher rate, like the conventional VCR’s fast-forward (FF) and rewind (REW) mechanisms. The FF and REW mechanisms are jointly denoted here as fast playout (FP) and the rate is given in an integer multiple of the normal playout rate. There is no distinction is made between FF and REW, because the only difference is the order in which the frames are sent.

Another alternative to fast playout is to jump to a certain location within a video. This can be done by using a random acces function (RAN) or a special video browser functionality. These alternatives are preferred over the fast playout functions, because less resources are needed. When the user performs the video browse function to the server, pictures of different locations in the video are sent and the users jumps to a location by selecting a picture. This function does not make use of continuous media.

The main difficulty in handling FP is that it introduces a wide variability in the bandwidth requirements of a single video stream. A video played at $n$ times the normal speed requires several components of the video server to operate at $n$ times the bandwidth than in case of
a normal playout. Video data of one video stream may require the storage media to retrieve data \( n \) times faster, from \( n \) disks parallel or a combination between these two.

8.2 Fast Playout Mechanisms

In order to overcome the increase in bandwidth usage due to fast-playout some mechanisms are proposed in this section.

8.2.1 Guaranteed Fast Playout Mechanism

To be able to provide each user a guaranteed playback rate \( n \) times the regular playback rate, the capacity of the system regarding the number of concurrent users is reduced by a factor \( n \). This is the result in case bandwidth is allocated based on the peak bandwidth method. The peak bandwidth method allocates resources based on the expected maximum need for normal playout. This method is extended for fast playout capabilities and denoted here as guaranteed fast playout.

Guaranteed fast playout provides enough resources to the user to perform fast playout sessions, analogous to the peak bandwidth method for playback only services.

For each user in the system, a certain amount of bandwidth is guaranteed to perform an FP session. In practice this guaranteed bandwidth approach is inefficient and not needed because of two reasons.

The first reason deals with the allocation method. When the peak bandwidth is reserved, the bandwidth allocation is very inefficient since \( n \times NP \) bandwidth is reserved for a single user. In case of a planned bandwidth allocation, where each user is also guaranteed enough bandwidth, a conflict occurs between the principle of this method and the nondeterministic nature of the users to perform a FP action. The principle of planned bandwidth allocation is that the needed bandwidth is known in advance. However, since users are allowed to deviate from the normal playout sequence, the needed bandwidth cannot be calculated proper enough in advance.

Second, in normal VCR's there is a small delay between performing an action and the actual execution of this action. This delay is caused by mechanical elements in the VCR. Besides this (mechanical) delay, a reduction of the video and audio quality takes place during a FP session. This suggests that it may not be a problem to introduce some latency and loss in resolution to the system when a FP session is requested.

The alternative to the two described solutions above is a more lossy one, which takes into account that a bandwidth reduction will not always be percepted by the users. The alternative is called effective fast playout service mechanism and is more effective since it does not provide hard guarantees anymore and the capacity of the system will therefore be higher. The principle of effective fast playout mechanisms is that a shared pool of bandwidth is used to overcome the increased use of bandwidth.
8.2.2 Effective Fast Playout Mechanisms

There are two motivations that make effective FP possible. The first is already mentioned in section 8.2.1 and denotes that a minor degradation in the quality of the video is allowed. The second motivation comes from the observation that users spent most of their time in NP mode and the duration of time spent in the FP mode is relatively small.

Because users spent just a little part of the time in FP mode, the extra bandwidth needed to perform FP functions does not have to be reserved all the time. Therefore, the new mechanism works with a guaranteed bandwidth only for the NP function. In addition, a small fraction of the server bandwidth is reserved for FP and is used to serve all of these requests. The principle of reserving channels or streams is also used in chapter 6 where clients are batched and some channels are used as contingency channels. The amount of bandwidth needed to reserve depends on the QoS constraints described in the next section and the ratio between the time spent in FP mode and in NP mode. If this ratio is known then the expected value of the number of users simultaneously performing FP is known from the total number of users that are currently in the system.

In [DSSKT94] two schemes are proposed for sharing reserved FP bandwidth among the FP requests. The first is called Delay Scheme (DS) and the second is called Loss Scheme (LS).

8.2.3 Delay Scheme

This scheme is called Delay Scheme (DS), because a user may be subject to an extra delay in case there is not enough bandwidth available to perform a FP session. The extra delay is caused by a waiting queue where clients wait for enough bandwidth to be freed. While waiting there are two possibilities.

Hold Bandwidth Delay Scheme

The first possibility is that the client continues to watch the video in the normal rate. This scheme is referred to as Hold Bandwidth Delay Scheme (HD). The advantage is that the user continues to watch the video, until FP becomes possible. When FP is possible, an extra amount of bandwidth is allocated.

Release Bandwidth Delay Scheme

In the second case, the server stops sending video frames and the bandwidth is released. Henceforth, this scheme is given is referred to as Release Bandwidth Delay Scheme (RD). For the user it seems like if the video has stopped or paused, but will only notice this after the buffer at the client side is emptied.

The released bandwidth could be used to provide other users to perform FP functions in case of a high load. The bandwidth that is released this way will however not be allocated by users entering the video server, because a user re-entering the NP state may not be subject to any delay.
8.2.4 Loss Scheme

Another way to serve requests for FP is to lower the bandwidth of each FP stream in order to accommodate the new request. This is only done when the amount of users performing a FP session exceeds the maximum number of concurrent FP users allowed by the system.

Under this scheme, requests are not queued and therefore the latency is smaller than in the other two schemes. A disadvantage of this method is the degradation in picture quality. A minor degradation in quality is no problem, but in order to guarantee a minimum quality, there is some maximum fractional loss defined. The loss experienced by a stream in FP mode depends on the number of concurrent client in FP mode.

As explained in chapter 9, a playout rate of \( n \) times the normal rate will not result in an increased bandwidth need by a factor \( n \). Skipping frames is allowed at the server side as long as the clients performing a FP session still receive video at about 30 fps.

From the previous it can be concluded that during a FP session a kind of loss scheme is applied inherently in the system. However, the loss scheme presented in [DSSKT94] performs an extra loss depending on the server load.

Skipping frames in an MPEG encoded stream is preferably done with B-frames, since no other frames depend on a B-frame. In section 9.3 a policy presented where B-frames are skipped. The impact of reducing the block length this way, is given in section 9.4.

8.3 A Model of the System

The model of a user watching a video is represented in figure 8.1. The user is only able to watch a video in the normal playout mode or in the fast playout mode. The other functions are left outside this model, because these use less bandwidth than the normal playout.

![Figure 8.1: Model of a user performing a fast playout](image)

In this two state Markov process, the arrival rate of a user performing FP is denoted with \( \lambda \). The average time the user stays in FP mode is negative exponential with mean \( \frac{1}{\mu} \). From the model the next expression can be derived:

\[
\lambda \cdot p_{NP} = \mu \cdot p_{FP}
\]

Using \( p_{NP} + p_{FP} = 1 \) and applying \( \rho = \frac{\lambda}{\mu} \) results in the following expressions for \( p_{NP} \) and \( p_{FP} \).

\[
p_{NP} = \frac{1}{1 + \rho}
\]
\[ p_{FP} = \frac{\rho}{1 + \rho} \]

Let \( B \) be the total amount of bandwidth available. The value of \( B \) is equal to bandwidth of the system bottleneck. \( \Phi_{NP} \) is denoted as the bandwidth needed by a user to perform a normal playout and \( \Phi_{FP} \) is the bandwidth needed to perform a fast playout. The ratio between these bandwidth is denoted with \( n = \frac{\Phi_{FP}}{\Phi_{NP}} \) and is only valid when no fractional loss applied. The value of \( n \) depends on the playout speed chosen by the user and may therefore vary from FP session to FP session. Within the value of \( n \) the use of frame skipping is already accounted for.

The expected number of users to perform FP is given in the following expression

\[ \mathbb{E} N_{FP} = N \cdot \frac{p}{1 + \rho} \]

where \( N \) is the number of users currently in the system. The maximum number of users possible in the system, \( \hat{N}_{\text{max}} \), is defined as the number of users possible in case of normal playout only service is used and where each user has a dedicated stream with bandwidth \( \Phi_{NP} \). \( \hat{N}_{\text{max}} \) is calculated using the following.

\[ \hat{N}_{\text{max}} = \left\lfloor \frac{B}{\Phi_{NP}} \right\rfloor \]

The number of \( \hat{N}_{\text{max}} \) users will not be possible unless streams are shared among users, because a part of the available bandwidth will be used as a shared pool for FP streams.

The amount of bandwidth needed for this purpose calculated from the ratio \( \rho \) and the average fast playout rate \( \bar{n} \). Because \( n \) is variable and is determined by the user each FP session, the average over all sessions is taken and the average \( \bar{n} \) is used instead. The expected average bandwidth needed for FP functions is equal to

\[ \mathbb{E} B_{FP} = \frac{\rho \cdot \bar{n}}{1 + \rho} \cdot N \cdot \Phi_{NP} \] (8.1)

where \( \Phi_{FP} = \bar{n} \Phi_{NP} \) is used. The amount of bandwidth needed for NP service can be calculated the same way and results in

\[ \mathbb{E} B_{NP} = \frac{1}{1 + \rho} \cdot N \cdot \Phi_{NP} \] (8.2)

which has to be, of course, equal to \( B - B_{FP} \).

From the equations 8.1 and 8.2, the following expression is found for the amount of bandwidth to reserve for FP functions.

\[ \mathbb{E} B_{FP} = \frac{\bar{n} \cdot \rho}{1 + \bar{n} \cdot \rho} \cdot B \]

Normalized to the system bandwidth \( B \), the relative bandwidth used for FP is

\[ \frac{\mathbb{E} B_{FP}}{B} = \frac{\bar{n} \cdot \rho}{1 + \bar{n} \cdot \rho} \]
and is equivalent to a maximum of
\[
\frac{\rho}{1+\bar{n} \cdot \rho} \cdot \hat{N}_{\text{max}}
\]
users. The system will be fully utilized if there are
\[
N_{\text{max}} = \frac{1 + \rho}{1 + \bar{n} \cdot \rho} \cdot \hat{N}_{\text{max}}
\]
users in the system. Fully utilized means that all bandwidth is allocated! However, in the model presented here, a user re-entering the NP state may not be subject to a delay. Therefore, the maximum number of users allowed in the system is
\[
N_{\text{max}}^{\text{NP}} = \frac{1}{1 + \bar{n} \cdot \rho} \cdot \hat{N}_{\text{max}}
\]
and denotes the maximum number of users concurrently to perform a NP session. If more users are allowed to access the server then the waiting times increase.

The release bandwidth delay scheme allows almost the same number of users that a playback only service allows under the circumstances that the fast playout is not used much compared to the normal playout function and that clients are prepared to wait up to a certain time.

Applying the fractional loss to the system will increase the maximum number of users even more. Adding this to the presented model, the fractional loss ratio \(\alpha\) is introduced and \(\bar{n}\) is replaced with \(\alpha \cdot \bar{n}\) \((\bar{n} \rightarrow \alpha \cdot \bar{n}\), with \(0 < \alpha_{\text{min}} < \alpha \leq 1\)\) and \(\alpha\) is calculated using the following equation
\[
\alpha = \frac{N_{\text{max}}^{\text{FP}}}{N_t^{\text{FP}}}
\]
where \(N_{\text{max}}^{\text{FP}}\) is defined as the maximum number of users to perform FP functions and \(N_t^{\text{FP}}\) is the number of users allocating FP bandwidth at a moment \(t\). The loss scheme is used when \(N_t^{\text{FP}} \geq N_{\text{max}}^{\text{FP}}\) and has a lower bound \(\alpha_{\text{min}}\).

### 8.4 Performance of the Schemes Compared

The performance of the different schemes is worked out in [DSSKT94] and the result is that the RD scheme allows more users and the reaction time is reasonably smaller than the HD scheme. An intuitive approach will learn us the following about the difference.

Users performing FP when there is a high load at the server, are put in a waiting queue. Using the HD scheme, the queue can grow until all the users are in it and the rest performs FP and as the latency of the system grows with the queue.

Using the RD scheme the queue is bounded to a value depending on the fast playout rate of the user first in the queue. The maximum queue length will henceforth be equal to \(\bar{n} - 1\) users. The main difference between both schemes is that in the RD scheme the reserved FP bandwidth is not fixed, but can grow and shrink with the arrival of the users.

The best performance of the system is achieved when the release bandwidth delay scheme is used in combination with the loss scheme. Using the release bandwidth delay scheme, the bandwidth for FP users can grow first and if the FP is already fully utilized then the loss scheme can be applied.
Chapter 9

Video Data Organization and Playout

The structure and definition of the compressed video stream imposes several constraints on the video data storage and playout. The challenge is to devise a scheme for fast playout operations which would satisfy the constraints of the decoder and require a minimum of additional system resources. The solution policies devised below for video data storage and playout meet these criteria.

The policies presented in this chapter are specific for MPEG-2 encoded streams, but could handle other encoding types as well. Some modifications are however needed.

9.1 Storage Policy for Video Blocks

To comply with the dependencies between the different frames of an MPEG-2 encoded stream, a video stream is divided into blocks, where each block consists of consecutive frames starting with an I-frame and ending before another I-frame. Typically the length of these blocks are about half a second long. The blocks in which the videos are stored are the primary unit of storage and retrieval. Allocation and storage of the video stream is done in blocks. Consecutive video blocks are placed on different disks in order to optimize the retrieval throughput in the disk array. It is not really the throughput that is optimized, as will be explained later, but the load balancing of the disks in the disk array and the disk utilization.

At first, the temporal order is given in which the video frames are displayed at the clients terminal. The MPEG-2 stream shown in figure 9.1 is just an example of some MPEG-2 encoded stream and shows the dependencies of the B-frames.

Figure 9.1: Temporal order in which the frames are displayed.
From figure 9.1 it can be derived that a part of the B-frames are displayed before it associated P- or I-frames are. In these cases a problem occurs at the decoder, which does not know how to decode these B-frames. First the decoder needs to decode the I-frames, then the P-frames and at last the B-frames.

Therefore, there is a difference between the way the frames are stored and the way frames are displayed. Frames have to be stored in a particular order such that a frame presented at the decoder does not depend on its successors. The storage order of the frames within a block is depicted in figure 9.2.

![Figure 9.2: Storage order of the frames in a block.](image)

The storage order of the frames within a block is chosen in the order in which they are presented to the decoder at the clients side. This choice minimizes the processing overheads during the normal playout of the video sequence and the memory needed to store the video sequence. In figure 9.2 it is shown that B-frames immediately preceding an I-frame are stored after this I-frame. It may even occur that a B-frame, that is displayed immediately before an I-frame, will be fetched one block of frames later.

Typically, a video block is large and comprises one or more disk tracks that are allocated entirely, because of the resulting substantial increase in access efficiency for storage and retrieval. In [KKMK96] the length of a video block is taken of size 64 kB–256 kB and the time to playback the video in a block is fixed, like in [ACM95] where a predetermined playback time of $\frac{1}{2}$ sec is chosen. In case of an MPEG-2 video stored in a block, $\frac{1}{2}$ sec video is equal to 256 kB, assumed that MPEG-2 needs 4 Mbps of bandwidth.

The advantage of a fixed playback time is that the video sequence needs a bandwidth with a much lower variation. The inherently variable bitrate video and audio streams are made constant.

### 9.2 Playout Policy for Normal Playout

Normal playout of the video streams is done in a *round robin* fashion. In each round, for every video stream, media blocks are read from each disk in the disk array. Consecutive blocks of one stream are placed on different disks in the disk array in order to get a higher throughput.

This higher (average) throughput is achieved because of the *load balancing* effect that will occur. When the load in the system is balanced, every disk will get the same amount of requests independent of the video contents on the disks.

In case of normal playout all frames within a block are read from the disk, as figure 9.3 already suggests. In this figure the gray part of a block is actually read.
The blocks are related like a linked list where the information about the location of the next block must be known at the end of the previous block. For a normal playout session this can either be done by adding this information within a block of video frames, or in a pointer list.

For a system where a user can perform fast playout it is more practically to keep a pointer list where the next blocks to access can be calculated from that list.

9.3 Playout Policy for Fast Playout

In the fast playout mode, the disk array based video server retrieves segments based on either the segment sampling method or the segment placement method [Chu96]. These methods ensure that for any given stream the segments to be retrieved all reside on different disks. Hence, the retrieval, buffering and transmission characteristics for the video server during the fast playout mode are similar to those during the normal playout mode.

In fast playout mode, information about the location of the blocks after the next block must also be known. There are different ways to establish this, but this depends on the way the fast playout service is actually implemented. For example, if fast playout with speed $n$ is done by retrieving $n$ frames serially then the mechanism of section 9.2 can be used.

When the $n$ frames are retrieved parallel or simultaneously, then the location of the $n-1$ frames following the first frame are not known at the time they are needed. A method to overcome this problem is by making use of a pointer list where all the locations of the blocks are stored.

Another way to perform fast playout is by selectively skipping frames at the server side. This can be done by skipping entire blocks or by skipping certain frames within a block. From now on the first method will be denoted as block skipping and the other method is named frame skipping.

Block skipping is easier to implement, because blocks are read similar to the normal playout blocks, except that in the fast playout mode blocks are skipped. The information of the next block is not known and is therefore calculated from a pointer list and the scheduler knows in advance which block to access.

For the user, it looks like a jerky normal playout, especially at higher playout rates. The main reason, therefore, not to choose for this alternative is because the users may not like the fast playout service.

The other solution plays all the blocks, but not entirely as figure 9.4 suggests. The reason to choose for frame skipping is that it is possible to playout the video at speed $n$ times the normal rate, without the need of an increase in bandwidth by the same factor. The skipping of blocks and frames is allowed, because the user receives video at 30 fps.
When playout is done as in figure 9.4 care must be taken that the frames sent to the client do not depend on frames that are skipped. Therefore, a storage scheme is chosen as depicted in figure 9.5. The only disadvantage using this storage order is that the client needs a longer buffer to store the P-frame now stored before the B-frames. But since the buffers at the receiving party have to be long enough to overcome the jitter, this would eventually not lead to an increased buffer need.

Using the storage scheme as in figure 9.5 the video server is able to read the blocks like depicted in figure 9.4. In [KNMN96] a similar fast playout scheme is given, but only I-frames are used for this purpose. The scheme presented here will use both I-frames and P-frames unless the speed requested is too high for this.

Compared to block skipping, the frame skipping scheme needs a higher bandwidth, because the smaller B-frames are skipped. The average frame length is therefore larger for frame skipping than for the block skipping scheme. Besides that, in case of frame skipping, multiple requests are placed within one time slot, unlike in the block skipping method.

The retrieval of the video blocks can be done serially as figure 9.4 suggest. In this case, a method called skip search is used. Skip search is taken from [KNMN96]. The other method is denoted here as subrequest method.

In the skip search method the data is exactly retrieved as visualized in figure 9.4. The I-frames and P-frames are read from a block and after that the next blocks are read in the same time that a normal playout retrieval would read one block.

Since the scheduling scheme presented in chapter 5 works with discrete time slots and requests are only accepted before the start of a new time slot, the requests are not free enough to skip through disks during a time slot. Because of this reason the subrequest method is used.

The request for a fast playout session is translated into requests for multiple blocks, which are retrieved parallel. The number of subrequests generated depends on the playout speed. The time to read a block in fast playout mode is shorter than the reading for normal playout. Therefore, there is some time left to read more fast playout streams before the cycle time is expired.
The disadvantage of using the subrequest method is that the principle of using a linked list is not usable for the fast playout functionality.

As mentioned before, the retrieval of a block in fast playout mode takes less time compared to the retrieval during normal playout. To give an indication about which part of the blocks in figure 9.4 is actually gray, the illustration in figure 9.6 is used.

In this figure, the letters I, P and B naturally stand for the frame types and the length of a frame is denoted by the value on the $\ell$ axis. Data is retrieved from the disk at a constant rate $r$. Since I-frames are the largest frames, they would take the longest time to be read. Based on the frame length distribution from figure 9.6, the distribution of the read times would then look like in figure 9.7. In this figure, $\tau_{NP}$ is equal to the time to read an entire block and $\tau_{FP}$ the time to read a block in fast playout mode where both I-frames and P-frames are transmitted to the client.

Let $\ell_x$ be the average length of an $x$-frame and $n_x$ the amount of $x$-frames stored in a block, where $x \in \{I, P, B\}$. Then, the following relation between $\tau_{FP}$ and $\tau_{NP}$ is derived:

$$\tau_{FP} = \frac{n_I \ell_I + n_P \ell_P}{n_I \ell_I + n_P \ell_P + n_B \ell_B} \cdot \tau_{NP}$$

In figures 9.1, 9.2 and 9.5 the next, arbitrary ratio $n_I:n_P:n_B=1:2:9$ is taken. However, the ratio of the frame type within a video depends on the encoding process of the video and therefore on the characteristics of the video itself. The ratio between the average length of the different frame types is usually taken as $\ell_I:\ell_P:\ell_B=4:2:1$.

Using the information from the previous paragraph the expression $\tau_{FP} = \frac{8}{17} \tau_{NP}$ is derived. Note that the resulting retrieval ratio strongly depends on video characteristics and is only as an illustration. In this illustration, a playout speed of 2x the normal speed would result in a lower bandwidth usage!
If a playout rate of about $10 \times$ the normal speed is used, then the choice could be made only to send I-frames. The time to read a block in this scenario is $\tau_{FP} = \frac{4}{17} \tau_{NP}$. Within a time slot 10 blocks are read this way resulting in an increase in bandwidth usage with a factor $10 \cdot \frac{4}{17} \approx 2.4$.

In the following section it will be shown that the bandwidth usage during 10 times the normal speed is not that efficient as suggested.

### 9.4 Overhead in Reading Blocks

The estimation of the bandwidth usage is best shown by calculating the maximum number of users the video server supports. For this purpose, figure 5.2 is used. This figure shows the retrieval of the information on one disk in one time slot.

In this system, the deadline to serve all requests is $T_{period}$. This time is spend by retrieving information ($t_i$) or searching this information ($\tau_{s_i}$). The derived expression for $T_{period}$, taken from section 5.3, is

$$T_{period} = \sum_{i=1}^{n} (t_i + \tau_{s_i}) + \tau_{\ell}$$

with $\tau_{\ell} \geq 0$. The time to read a block is the same as the length of the block ($L - i$) divided by the retrieval rate ($r$) of the storage medium. After applying $t_i = \frac{L_i}{r}$ we get the following expression.

$$T_{period} = \sum_{i=1}^{n} \left( \frac{L_i}{r} + \tau_{s_i} \right) + \tau_{\ell}$$

This expression is simplified by taking the average of the blocklength, $\bar{L} = \frac{1}{N} \cdot \sum_{i=1}^{N} t_i$ and the average of the seek latencies, $\bar{\tau}_s = \frac{1}{N} \cdot \sum_{i=1}^{N} \tau_{s_i}$. For $T_{period}$ we get

$$T_{period} = N \cdot \left( \frac{\bar{L}}{r} + \bar{\tau}_s \right) + \tau_{\ell}$$

To be able to calculate the effect of fast playout, the fast playout characteristics are added to the expression above. Let $\xi$ be the fraction of users that performs a normal playout and $1-\xi$ the fraction for fast playout averaged on a continuous basis ($0 \leq \xi < 1$). The average fast playout rate is denoted by $\bar{n}$ and $\alpha$ is the the average fraction of the blocklength used for fast playout ($0 < \alpha \leq 1$).

The use of fast playout will increase the maximum number of requests per disk, because the blocklength is smaller. However, the total number of requests supported by the system decreases due to the multiple seek latencies needed for a fast playout session.

The overhead of reading a block in fast playout is higher, because the total seek latency is increased by a factor $\bar{n}$ and the amount of information retrieved by a factor $\alpha \bar{n}$. To illustrate this, the model of the retrieval process is altered without losing the correctness of the already
presented retrieval process. The parallel retrieval from the disks is serialized in a way that subrequests to multiple disks are considered as subrequests to a single disk. This is allowed when averages are considered.

Adding fast playout to the system results in the following.

\[
T_{\text{period}} = \frac{\xi \cdot N \cdot \left( L \cdot \frac{1}{r} + \bar{\tau}_s \right)}{r} + (1 - \xi) \cdot \frac{N \cdot \left( \frac{\bar{n} \cdot \alpha \cdot \bar{L}}{r} + \bar{n} \cdot \bar{\tau}_s \right)}{r} + \tau_{\xi}
\]

\[
\geq \frac{\xi \cdot N \cdot \left( L \cdot \frac{1}{r} + \bar{\tau}_s \right)}{r} + (1 - \xi) \cdot \frac{\bar{n} \cdot \alpha \cdot \bar{L}}{r} + \bar{\tau}_s
\]

\[
= N \cdot \left\{ \frac{\xi \cdot \left( L \cdot \frac{1}{r} + \bar{\tau}_s \right)}{r} + (1 - \xi) \cdot \bar{n} \cdot \left( \frac{\alpha \cdot \bar{L}}{r} + \bar{\tau}_s \right) \right\} \]

The expression for the number of users allowed to access a disk is

\[
N \leq \frac{r \cdot T_{\text{period}}}{(1 - \xi) \cdot (\alpha \cdot \bar{L} + r \cdot \bar{\tau}_s)} \cdot \frac{1}{n + \frac{\xi \cdot L + r \cdot \bar{\tau}_s}{\alpha \cdot L + r \cdot \bar{\tau}_s}} \tag{9.1}
\]

**Example**

The maximum number of users allowed on a single disk is calculated from equation 9.1, assumed that \( r = 40 \) Mbps, \( T_{\text{period}} = \frac{1}{2} \) s, \( \xi = \frac{9}{10} \), \( \alpha = \frac{1}{2} \), \( \bar{L} = 2 \) Mb and \( \bar{\tau}_s = 0.5 \) ms. Given these values, there are \( N \leq \frac{190}{n + 17.6} \) simultaneous users possible. For \( \bar{n} = 5 \), this is equal to \( N < 9 \).

By taking \( \bar{L} = 2 \) Mb and \( T_{\text{period}} = \frac{1}{2} \) s, a stream has a bandwidth of about 4 Mbps. With a disk transfer rate \( r \) of 40 Mbps, the disk can handle a maximum of 10 streams.
Chapter 10

The Future of Broadband Internet Services

In this chapter, a prediction is made about the development of new multimedia services delivered in the future and the (dis)organization behind these services. Assumed is an infrastructure capable to deliver broadband services.

A brief look is presented about the development of services and how the customers actually experience them. The customer is a very important parameter in the design of a system and service, because they make or break a service.

From the point of view of the users, the system is divided into two parts, which are denoted here as the front-end and the back-end. The front-end part of the system is the interface to the customers and is described in section 10.3.

In section 10.1 the back-end development is viewed regarding the organizational sense. Who will deliver services and how the customers connected to these services? Section 10.2 is devoted to the technological innovative techniques in distributed computing.

10.1 The Back-End Development – The Organization

10.1.1 Integration

An integration process is started to deliver different type of services over a network. Traditional and typical networking systems which are commercially available in a domestic environment are the bidirectional low bandwidth circuit-switched telephone networks and the unidirectional analog coaxial cable used by cable companies to broadcast television channels. Telephone systems allow bidirectional traffic, but lack a proper system to handle bursty data traffic, like for instance the control signals to interact with a remote video server.

Sharing the different types of traffic over a single medium is not the only integration process in the development of multimedia information services. The equipment at the client side is also subject to an integration process as explained in section 10.3. The organization behind the delivered services will be distributed one and providers will deliver integrated packages of services.
For instance, cable companies intend to deliver internet services next to broadcast television and integrate different services to deliver them as a single package. But what will the organization look like if internet television is introduced and there are more than thousand channels available on the internet, not to mention different other services like remote data processing and telephony.

10.1.2 Distribution of Media Sources

From a historical point of view, media has developed itself from a more central, governmental approach to a distributed, commercial one where the cable companies determine the broadcast channels one can choose from. The choice is made at the client side of the system, because customers receive all channels even when only one channel is actually used. Selection is made only at the side of the client and the interactivity is therefore limited to the capabilities of the equipment at the client side and the information this equipment receives.

On the internet, there is much more freedom to choose, partly because internet is an extremely distributed medium and traffic is bidirectionally. The advantage of bidirectional traffic is that interactions reach further into the network and have influence on what the equipment at the client side receives.

The freedom in accessing a service is limited only by the knowledge of the users, their authority and their equipment. Authority is determined by the governmental law, the service providers and the access provider. The terms service provider and access provider used here, are best explained using the schematic representation in figure 10.1.

```
Figure 10.1: Schematic representation of a client connected to an access provider.
```

Customers access the network through access providers, which can best be compared with today’s cable companies. Access providers deliver the service off the service providers, which are servers elsewhere on the internet or can reside in the access provider itself.

Access providers can help customers finding the services wanted by preselecting a subset of the offered sites and presenting this to the user. This is further explained in section 10.3.

The organization of multimedia services will become on user-demand and the provision of these services is done by access providers, which have a distributed character regarding the internet, but a more centralized character regarding the subscribed customer. Traditional broadcast channels would also become available this way and will then coexist with newer generation of multimedia services and distributed services. The principle of broadcasting will
however be replaced by a multicasting scheme, where users receive a channel only if requested for it.

10.2 The Back-End Development – A Technical Overview

The overview of the technical development is viewed using the schematic representation of the system in figure 10.2. The arrow from STB to SERVER is the uplink used to interactively choose a service and to navigate and control within a service. This uplink will eventually share its medium with the downlink, but before the cable becomes bidirectional, the twisted pair lines of the telephone will be used for this purpose.

![Figure 10.2: Position of the set-top box in the system.](image)

Most familiar with current services is the download link, where data is sent from the provider to the customer. When the downlink is made digital and new interactive services are introduced, the devices at the client side (DEV) have to be extended with a Set-Top Box, denoted in figure 10.2 as STB. Philips is currently working on a STB where an Electronic Program Guide (EPG) is integrated.

The STB is a small box with its own platform and is able to run software. Signals from the network into the STB have to be translated such that devices connected to the STB are able to process them. The STB is furthermore able to extend its functionality by downloading software from the server if needed. The whole idea behind this is that customers use the offered services transparently from the software involved. A new service is added to the terminal by sending the required programs.

Distributed applications which load programs into the client side equipment require that STBs have the same platform, or that the application is platform independent. The only real platform independent language known is Java. Java is developed by Sun and the principles of Java are depicted in figure 10.3.

Microsoft has developed a counterpart of Java and is called ActiveX. ActiveX makes use of standard objects within the operating system. ActiveX is not really platform independent yet, but that is currently under development.

10.3 The Front-End

The integration process will eventually reach the equipment at the side of the customers. The STB is just a temporary solution for the STB which is integrated into the television. And this newer generation television will be replaced or coexist with a device which integrates multiple devices like telephone, radio, television, computer and an internet terminal.
This integrated device will have different levels of complexity, depending on what the customer wishes to do with that device. For instance, if a user seldom uses the television as a data processor, then the television will not be integrated with an entire computer, but with just enough equipment to perform the data processing part at a remote server. The functionality in a television and its complexity will be specified by the user.

As stated in the previous sections, the customer can make or break a service. Therefore, much effort is done in the development of friendly user interfaces. Many potential customers are not familiar with internet and may want to use internet services provided that it is easy to access these services. With the development of the new generation televisions and set-top boxes, Philips targets these consumers that have never owned a computer.

The importance of making a preselection and categorisation is more important than ever, because an over-abundance of information can scare customers away, because the services look and feel very complex and difficult. The capabilities of the internet are reduced to something that is much richer than Teletext and even easier to use.

A feature that can be added is that the preferences of a user are stored and are used when the user accesses the system again. When subscribed to an access provider, a client could get a small account to keep the mailbox and its preferences in. This account is naturally transparent to the user. The user only sees its interface with mail icons and bookmark icons and performs interaction by using the remote control. The remote control will however be extended with a few buttons or a trackball but still resembles far more like a remote control than like the optional keyboard.

Within the clients equipment there is a small platform where software can run. This software is mainly something that works like an internet browser with extensions to decode and run applications. In case the software is too old or not present at all, it will automatically be downloaded from the server.
To make the equipment at the client side cost effective, the complexity is kept low in the beginning enough to deliver the basic services. Extra functionality can be loaded dynamically if some extended functionality is wanted. Users can extend the functionality of their equipment by upgrading their system common like the computer world.
Chapter 11

Conclusions and Recommendations

The main conclusion is of course that much work still has to be done to make VoD services possible on a wide area scale and that it will take a long time before true VoD is commercially available in a domestic environment. The amount of time needed is difficult to predict, because the development of the different disciplines in a VoD system is as distributed as the service itself.

VoD will in any sense be a very cost intensive service to implement and therefore a major design issue is to maximize the number of concurrent streams possible traded off a certain, given a certain quality of service to deliver. The disadvantage of true VoD is that resources will have to be set aside to perform VCR functions and the amount of streams are thereby reduced. Because true means that a user must have the idea it uses its own, private VCR, a fast reaction of the system to the user is required. This reduces the total number of concurrent streams even more.

The mechanisms proposed in this thesis for the storage and playout of video streams increase the number of concurrent streams to almost the amount possible in a VoD service with only PLAY, PAUSE and STOP functionalities.

Much effort is done to make it possible to run applications distributed over a network and over multiple platforms. Examples are ActiveX and Java and it is recommended that the implementation of a video server and client is done in such a language. A further recommendation for the implementation of a video server is that is must be made scalable from a server for a local network and a few videos to a wide area server with extended functionalities like interactive television.

VoD is complexer than interactive television, but using the current infrastructure, even interactive television is only possible when the channels are transmitted through the coaxial cables of the cable companies, and the control signals from the client are transmitted through the twisted pair telephone lines, because the cable is unidirectional.

When the cable becomes bidirectional and the bandwidth for all users increases then the development of services like VoD will speed up and will eventually coexist besides a broadcast television that is also on user-demand.
Appendix A

MPEG

MPEG is a video and audio encoding standard and is the acronym for Motion Pictures Expert Group. Different standards for MPEG have been proposed. The first one was MPEG-1 and was targeted and optimized for CD-ROM or applications at about 1.5 Mbps.

After MPEG-1, another MPEG standard called MPEG-2 was developed that was targeted for cable television industry. Later it was added the HDTV standard and MPEG-3 stopped. Typical bitrates for MPEG-2 encoded videos lie between 2–20 Mbps.

A.1 MPEG-2

MPEG-2 video consists of three different frame types. These are the intra frame encoded frames, the predictive encoded frames and the bidirectional encoded frames and they relate to each other like in figure A.1.

Figure A.1: MPEG frame dependencies.

B-frames are typically the smallest frames, because only the difference information between the consecutive I-frames and P-frames is encoded. The compression rate is about 1:30 and figure A.2 shows the different layers from where information is retrieved.
Figure A.2: MPEG layers.
Appendix B

Laws

B.1 Zipf’s Law

Zipf distributions are frequently used to express the probability of selection of a particular object from a fixed number of objects where there is a skew toward some of the objects. A pure Zipf distribution has a single parameter, the number of discrete objects to be selected from. In a Zipf like distribution a second parameter, \( \theta \), is added to specify the skew. If \( \theta \) is zero, the distribution is Zipf; if it is one it is uniform. Relative probabilities are converted to probabilities by normalization. The use of a Zipf like distribution with \( \theta \) of 0.271 closely matches a published video rental distribution of 92 movies.

\[
F_i = \frac{1}{i^{1-\theta}}
\]

\[
P_i = \frac{i^{1-\theta}}{\sum_{i=1}^{N} i^{1-\theta}} = \frac{F_i}{\sum_{k=1}^{N} F_k}
\]

B.2 Moore’s Law

The observation that the logic density of silicon integrated circuits has closely followed the following curve

\[
N = N_0 \cdot 2^{t-1962}
\]

where \( N \) is the number of bits per square inch and \( t \) is the time in years. The amount of information storable on a given amount of silicon has roughly doubled every year since the technology was invented. This relation first uttered in 1964 by semiconductor engineer Gordon Moore.

B.3 Parkinson’s Law of Data

“Data expands to fill the space available for storage”. Buying more memory encourages the use of more memory-intensive techniques. It has been observed over the last ten years that the memory usage of evolving systems tends to double roughly once every eighteen months. Fortunately, memory density available for constant dollars also tends to double about once every twelve months (see Moore’s Law).
Unfortunately, the laws of physics guarantee that the latter cannot continue indefinitely.
Appendix C

Scalability in a VoD System and Quality of Service

C.1 Importance of Scalable Systems

A system is called scalable if it is able to continue to work, even when some parameters in this system vary significantly. In a VoD system, the best example is the increasing number of users. If the number of users grows, the system must be able to handle this without resource problems or significant changes in the system structure.

Scalability of a system is important in a way that, when some parameters in the system vary, the system is able to overcome this unexpected or unwanted situation and keeps on working without significant performance degradation. A user experiences the scalability of a system in how fast and accurately the system responds to actions [Ka95]. The response should be received in an acceptable time without any errors. The quality of the system can be measured via system response time, availability of the system and the reliability.

C.2 Response time

The response time is defined here as the time between moment that the user performs an action and the moment the user can detect the reaction of the system. It is of great importance to keep the response time low, because long response times can have devastating effects on the popularity of a service. In a VoD service, the user expects to have a response time that is about the same as conventional video players up to a maximum of about 2–3 seconds.

There is however a difference between initial response time and response during the service. At the start of a service, users are willing to wait a longer time as long as the maximum waiting time is known.

C.3 Availability

The availability is defined as the percentage of the time the system has been available. The availability depends on all components in the system and therefore a system has to built such that the failure of one of the components cannot result in the service to become unavailable.
A method to avoid this is by duplicating the service to several servers. If some of the servers become unavailable then only some capacity is lost.

Capacity can also be the cause of unavailability. If there is a high network or server load and the system is running on its capacity, then the user will not be serviced. The probability that a user cannot be serviced due to a high load is called blocking probability. Blocking probability depends heavily on the users base. For wide-area internet services, this user base is not fixed and therefore unknown in advance.

A system with a user base known in advance can be designed such that a given blocking probability is met. This is not a new design issue. Telephone systems in the past had to design their systems such that a certain blocking probability is met [Wee93]. The probability for a user to access a server that is currently busy is preferably kept low. Since there is a trade-off between availability and cost of the system, a percentage of below 5% is fine and 1% is good.

C.4 Reliability

Reliability in VoD system means that after a request is submitted, it accurately responds without errors. The server hardware and software must be designed with such carefulness that unexpected events or system overload cannot cause a reduction of the reliability.

C.5 Quality of Service

The scalability of a system is related to the Quality of Service (QoS). When scaling the system, the QoS still has to be met. QoS is mostly measured in the throughput of the system, the delay and the variations in the delay, called jitter.
Appendix D

Video-on-Demand Services

The current television broadcasting will meet a fundamental change by interactive video delivery services. In contrast to broadcasting, where users select one of the available channels to view at the particular time of broadcasting, interactive services provide a much wider selection of programs and the programs become available at user demand.

Based on the level of interactivity, interactive services can be classified into several categories. The following categories are collected from [LV94] and listed in order of increasing interactivity.

D.1 Service Categories

Broadcast Television

Broadcast services similar to broadcast television, in which the user is a passive participant and has no control over the session, except the choice of a particular channel.

Currently, a terminal at the side of the user receives all the broadcast channels. Broadcast television will change into a multicast television, where the user can log in onto a multicast stream.

Pay-Per-View

PPV services in which the user signs up and pays for specific programming, similar to existing CATV PPV services, already available. PPV represents an incremental change from broadcast television supporting prescheduled programs selected by a user, and is easily supported by simple VCR’s and network communication. Pay-per-view is highly multicast, like broadcast television and therefore less complex to implement than the following categories.

Interactive Television

This is the same as the previous two services, but this service is added user interaction that goes beyond channel switching. An example of interactive television is multiple perspective interactive television, where a user is able to choose from which camera instance to watch by selecting one of the available streams. Another example is that a user can interactively attend to a quiz.
APPENDIX D. VIDEO-ON-DEMAND SERVICES

Near Video-on-Demand

NVoD services provide additional interaction by grouping requests for individual programs and scheduling them at regular intervals. This provides the ability to pause a program and resume by restarting the program within a separate group at a constant time offset. Fast playout is done in integer multiples of the time intervals by switching to a different group. Near Video-on-Demand is a multicasting scheme with user interactions and provides VoD with a lower QoS.

True Video-on-Demand

(TVoD) services, where the user has complete control over the session presentation. The user has full-function VCR (virtual VCR) capabilities, including forward and reverse play, freeze, and random positioning. TVoD needs a single channel per user, while multiple channels become redundant.

Of all categories mentioned, TVoD is the most complex and expensive. In the rest of this report VoD is always TVoD unless otherwise stated. Within the category of VoD there are different service levels, based on the Quality of Service (QoS) delivered. Examples of QoS parameters are the video quality and responsiveness of the system.

D.2 Interactive Services

Interactive services cover a wide range of services from Movies-on-Demand to distance learning. Some of the basic interactive multimedia services are listed in the following.

Movies-on-Demand

Customers can select and play movies with full VCR-capabilities. In the rest of this report this is denoted as Video-on-Demand.

Interactive video games

Customers can play downloadable computer games without having to buy a physical copy of the game. An example of interactive video games is SEGA channel, a trial of SEGA in North America.

Interactive news television

Newscasts tailored to customer preferences with the ability to see more detail on the preferred highlights. Interactive selection and retrieval of the items is possible.

Catalogue browsing

Customers examine and purchase commercial products.

Distance learning

Customers subscribe to courses being taught at remote sites. Students tailor courses to individual preferences and time constraints.
Interactive advertising
Customers respond to advertiser surveys and are rewarded with free services and samples.

Multimedia mailing systems
Electronic messages can contain audio, video, text and graphics. This service will be implemented in WebTV™, which is a television with internet capabilities.

Video conferencing
Clients can communicate with each other using the integrated audio, video, text and graphics.
Bibliography


