Ulmer Informatikberichte Nr. 2000-05, April 2000

Intelligent Prefetching and Buffering for Interactive Streaming of MPEG Videos

Susanne Boll, Christian Heinlein, Wolfgang Klas, Jochen Wandel
Databases and Information Systems (DBIS)
Computer Science Department, University of Ulm, Germany
{boll,heinlein,klas,wandel}@informatik.uni-ulm.de

ABSTRACT

Continuous delivery of media streams like video over IP networks so far is mainly handled by commercial approaches that deliver the stream forward-oriented in their own proprietary format. Though some existing streaming technologies are able to adapt to varying bandwidths, they do not provide smooth reactions to user interactions with the continuous stream.

We have developed the *MPEG-L/MRP* strategy, an adaptive prefetching algorithm for the MPEG-1 video format in combination with an intelligent buffering technique that allows for smooth and quick reactions to user interactions with the stream. With L/MRP [12] an approach already has been presented to deliver and buffer homogeneous continuous data streams like Motion-JPEG with special focus on fast reaction to user interactions. In contrast, the MPEG-1 encoding with its different frame types and the dependencies between frames opens the door to a more fine-grained adaptation of the continous stream. However, the complexity of MPEG-1 calls for comprehensive adaptation and special amendments of the L/MRP algorithm to make it an efficient preloading and buffering technique for MPEG-1 videos.

With the realization of *MPEG-L/MRP* in the context of a multimedia presentation engine on top of a multimedia repository we have an efficient means to deliver continuous streams of interactive multimedia presentations over existing IP infrastructure trying to minimize interaction response time and optimize loading/reloading portions of a video stream.

1. INTRODUCTION

In future, users of multimedia applications will no longer be satisfied with pre-packed presentations on stand-alone systems or proprietary compositions embedded in Web pages and rendered by browser plug-ins. Rather, personalized interactive multimedia presentations are needed, delivered on-demand from a multimedia server over an IP network to a user's flexible presentation environment. In this context, the delivery of continuous multimedia data as well as its presentation must be tailored to the specific requirements of this environment, i. e., the varying bandwidth, response time of the server, and the like.

The motivation of our work in the area of continuous delivery of interactive multimedia presentations over a network stems from our research project "Gallery of Cardiac Surgery" (Cardio-OP[1]) [8] which aims at developing an Internet-based and database-driven multimedia information system in the domain of cardiac surgery. The users of the system request multimedia content from different platforms over different network connections. Video streams are of high importance in this educational environment. During the learning process, it is indispensable for the user to interact on the stream so as to watch a scene again or jump to another interesting part of the video. Therefore the system must support interactions and, to be user-friendly, should react in a very responsive way. Hence, the presentation environment demands for streaming support for continuous media with suitable handling of user interactions.

In the project context, we developed a multimedia presentation engine which includes support for continuous MPEG video streams. For this, we developed the *MPEG-L/MRP* algorithm to continuously deliver MPEG-1 video streams over an IP network which we present in this paper.

Compared with, e. g., Motion-JPEG, the encoding of continuous video streams with MPEG-1 offers a significantly higher compression rate which is very important for a delivery over a network with potentially low bandwidth. We aim at continuously delivering the MPEG-1 stream in small units and at buffering these units in an intelligent way at the client such that the user is provided with a smooth and continous presentation though the user can possibly carry out VCR-like interactions on the stream like fast forward, reverse, or jumping to a bookmark in the video. The buffering technique should hide the request and buffering of units and rather deliver a continuous MPEG-stream of the best quality that can be currently provided to the application.

With *L/MRP* [12] we find a preloading and buffering strategy for continuous streams supporting interactions that has proven to perform better than "traditional" strategies like, e. g., LRU, FIFO, LFU, etc.This approach, however, aims at delivering and buffering *homogeneous* continuous data streams like Motion-JPEG with special focus on fast reaction to user interactions. The complexity of MPEG-1

[1]Partially funded by the German Ministry of Research and Education, grant number 08C58456. Our project partners are the University Hospital of Ulm, Dept. of Cardiac Surgery and Dept. of Cardiology, the University Hospital of Heidelberg, Dept. of Cardiac Surgery, an associated Rehabilitation Hospital, the publisher Hüthig-Verlag, Heidelberg, FAW Ulm, and ENTEC GmbH, St. Augustin. For details see also URL http://www.informatik.uni-ulm.de/dbis/Cardio-OP/

with its heterogeneous frame types of different importance, varying frame sizes, and inter-frame dependencies calls for comprehensive adaptation and special amendments of the original L/MRP algorithm to make it an efficient preloading and buffering technique for MPEG-1 videos. This paper presents our MPEG-1 specific preloading and buffer management strategy *MPEG-L/MRP* for MPEG-1 videos.

The remainder of this paper is organized as follows: Section 2 discusses related work. Section 3 revisits the original L/MRP algorithm and gives a short overview of the parts of MPEG-1 relevant to our approach. In Section 4, our new MPEG-L/MRP approach is presented which consists of a formal model and a corresponding algorithm. Section 5 sketches the implementation of the approach and Section 6 concludes the paper.

2. RELATED WORK

Related work, concerned with the delivery of multimedia content over the Internet, covers several research approaches dealing with the adaptive streaming of MPEG videos. As a part of the QUASAR project at the Oregon Graduate Institute [19] an MPEG player for adaptive MPEG streaming over the Internet has been developed which addresses resource scarceness in the end-to-end delivery. The focus lies on a quality of service (QoS) model and an adaptation mechanism of the player. To facilitate adaptive streaming, the MPEG video is provided by the server in different qualities. The stream is adapted in the temporal dimension by dropping B frames first, then P frames, and finally I frames. In addition, different spatial resolutions are provided as a second variable quality dimension. Buffering is applied to compensate network jitter but does not support fast reactions to user interactions. Another approach, the Media Streaming Protocol [4] developed at the University of Illinois, provides adaptive streaming of MPEG movies, too. On congestion, the protocol considers the different frame types of MPEG with their frame interdependencies and, similar to our approach, drops less important MPEG frames first. The client side buffer is employed only to smooth the jitter of arriving data but does not allow for minimizing interaction response time and reload of data after possible user interactions.

In the commercial area, many approaches can be found that deal very well with the streaming of videos, e. g., Quicktime [1] or Emblaze [2]. With VDOLive [17] and Real [14] approaches exist, that are furthermore able to adapt the video stream to fluctuations of the available bandwidth. For instance, with the introduction of the SureStream technology [15], Real allows to encode a video clip that serves for up to six different bandwidths. This stream can automatically be adjusted to compensate for network congestions. However, as this technique encodes multiple disjoint streams into one file, it leads to an inflation of the storage size and to redundancy. However, all the commercial approaches mentioned have in common that they operate on proprietary video formats and are neither designed to support minimization of the interaction response time nor to optimize the effort for reloading portions of a video stream.

With Q-L/MRP [3] an interesting application of L/MRP has evolved. Q-L/MRP extends L/MRP with additional interaction sets in order to support the specific QoS requirements of certain users. However, the approach does not deal with MPEG specific preloading and replacement strategies.

3. L/MRP AND MPEG-1 REVISITED

3.1 L/MRP

L/MRP (**L**east/**M**ost **R**elevant for **P**resentation) [12] is a buffer management strategy for interactive continuous data flows in a client/server environment. The client requests and receives a continuous medium in small units and buffers that part of the stream that is relevant for the current and future presentation. The main idea is to request, preload, and buffer those units that are *most relevant* to be presented in the near future. The speciality of the L/MRP strategy here is that the preloading and buffering takes into account the interactions a user possibly carries out on the stream, e. g., switch to fast forward playback or jump to a bookmark. By that means, the interaction response time compared to common buffer management and replacement strategies is reduced (cf. [12]). *Preloading* and *replacement* are the two tasks the buffer management strategy has to master. During preloading the next most relevant units of the continuous stream are determined, whereas the replacement strategy must decide which are the least relevant units as these are removed from the buffer to free space for more relevant units.

The L/MRP buffer management strategy treats the stream as a sequence of so called *Continuous Object Presentation Units (COPUs)* with an ascending numbering of the units. Looking at a sequence of COPUs from a specific presentation point p in time, the single COPUs are differently relevant for the current presentation which is expressed by assigning relevance values to each COPU. Consider Figure 1 for an illustration: The current presentation point is $p = 43$ and the user is watching the stream at double speed in forward direction. Then, every other COPU in forward direction close to the current presentation point is absolutely relevant for the upcoming presentation. These COPUs form the so called *referenced* set, as they are likely to be referenced in the near future. However, there are COPUs that already have been viewed. These belong to the *history* set of COPUs of the stream. As a user could change the direction of the playout at any time, these COPUs are still relevant for the presentation. Finally, the frames in forward direction which are *skipped* due to the double speed playout, are relevant, too, as the user could switch to normal speed playback at any time. These considerations can be continued for further interaction types such as fast backward, jumping to bookmarks, and the like.

The relevance of a COPU with respect to one of these sets is determined by a so called *distance relevance function* which expresses a COPU's relevance as a function of the distance of the COPU to the current presentation point p. For the referenced set, the distance relevance function is monotonously decreasing with value 1 for the next few COPUs to be presented. As the frames of the history and skipped sets are less likely to be presented, their distance relevance functions are decreasing more rapidly. Given one or more relevance functions for each COPU, an overall relevance function can be calculated, e. g., by taking the maximum relevance value for each COPU. This global relevance function is then used by the preloading and replacement of the buffer. The rel-

evance value expresses which COPUs are likely to be presented when taking into account the different interactions a user could perform on the stream. L/MRP tries to keep those most relevant COPUs in the client buffer to achieve a quick and smooth reaction to the user interaction. Depending on the buffer size those COPUs above a certain relevance value are kept in the buffer and those below the threshold value are not loaded/are removed from the buffer to make room for the more/most relevant COPUs. Whenever the presentation point p proceeds, the relevance values are recalculated, the COPUs to be preloaded are determined and the COPU(s) with the least relevance value in the buffer are replaced.

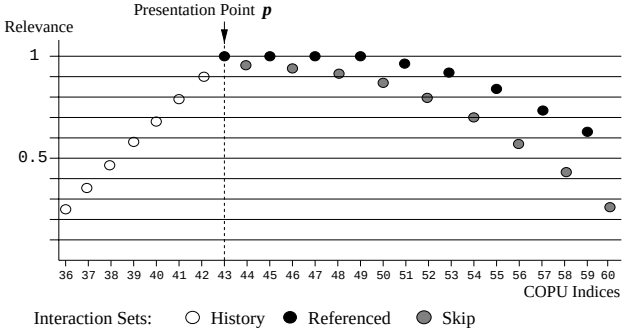

Figure 1: L/MRP: interaction sets and relevance values

3.2 MPEG-1

The MPEG-1 standard [6] is a coding format for audio and video streams. In this paper, we are concerned with video streams only [7]. The main feature of MPEG-1 that is interesting in this paper is that frames are no longer independent of each other as is the case with, e. g., Motion-JPEG which is a series of single JPEG [18] images. Figure 2 shows a sequence of MPEG frames and their interdependencies which are relevant for decoding the stream. An MPEG-1 video sequence in general consists of three different frame types, *I*, *B*, and *P*. Usually, the frames from one I frame up to the frame before the next I frame form a so called *Group of Pictures* (GoP). Since I frames (intra-coded pictures) are encoded similarily to JPEG images, their decoding is independent of other frames. The decoding of P frames (predictive coded pictures) depends on the preceding I or P frame of the same GoP. For B frames (bidirectionally coded pictures) decoding depends on both the preceding and the succeeding I or P frame. P and B frames allow a much higher compression rate than I frames by exploiting temporal prediction using motion vectors. It is important to note that the display order in which the frames are presented is different from the bitstream order in which the frames are decoded due to inter-frame dependencies. Figure 2 illustrates both the display order and the bitstream order of a stream. The order for decoding is very important as a preloading strategy must of course consider the order of decoding and not only of displaying the frames.

A preloading and buffer management strategy for MPEG-1 video must pay attention to the different frame types and

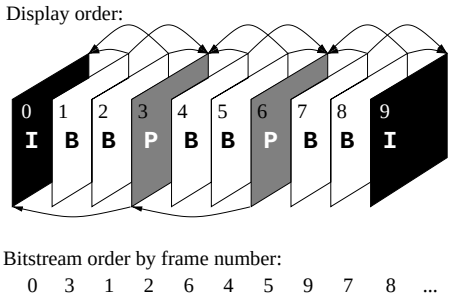

Figure 2: MPEG frame types and their interdependencies

their inter-frame dependencies, the bitstream order for decoding the stream, and the fact that the bitrate/data rate of the video and the size of the frames can heavily vary.

4. MPEG-L/MRP MODEL

4.1 Overview of MPEG-L/MRP

Basic idea

So far the L/MRP approach has proven [12] to be superior to traditional preloading and buffering strategies especially when it comes to fast reaction to user interactions. The basic idea of *MPEG*-L/MRP is to provide the same interaction responsiveness as achieved with L/MRP but in particular to take into account the specific features of the MPEG video stream. The different frame types with their inter-frame dependencies and their different importance for the presentation are the main issue when adapting L/MRP to MPEG. The MPEG-L/MRP strategy exploits the knowledge about the importance and dependencies of the frames such that the video can be optimally presented under the available network bandwidth. Therefore, the interaction sets and the associated relevance functions of the L/MRP strategy are adapted such that they reflect this specific importance of frames for the presentation. When frames do not arrive in time at the client, temporal adaptation is used in order to maintain a continuous presentation.

Choosing the appropriate COPU size

The first issue of adapting L/MRP to MPEG streams is the kind and size of the data that forms a COPU. The COPUs are the basic units for transportation of the stream. Looking at MPEG-1 there are different possibilities to define a COPU:

A COPU corresponds to a GoP. The rather big size of the COPU might be a problem. If such a COPU cannot be delivered to the client, in average half a second of the video is missing. This size is also unsuitable for, e. g., a fast forward presentation of the video, since all frames had to be loaded to the client though only a subset of them would be needed.

A COPU corresponds to a part of a GoP. [5] proposed to use IBB or PBB groups. However, the groups and therefore the COPUs are then dependent on each other. And this restricts the supported coding scheme of the MPEG stream to IBBPBB...PBB patterns.

A COPU corresponds to a frame. Here, still the COPUs are dependent on each other like the frames of the MPEG stream are. However, this granularity allows for fast

and targeted reaction to varying network bandwidth and user interactions.

We decided to use the third alternative as it offers the most appropriate possibility to compensate fluctuations in the available network bandwidth and, at the same time, offers support for fast and smooth reactions to user interactions on the stream. This decision serves as the basis for the formal model to follow.

4.2 The MPEG-L/MRP Model

Overview

In this subsection, the MPEG-L/MRP model will be developed step by step. Following some preliminary definitions, we introduce *presentation sets* as a means to collect those frames which have to be displayed for a particular kind of presentation of a video, such as normal playback, double speed presentation, and so on. Since P and B frames cannot be decoded independently, additional I or P frames might be necessary to actually decode and display the frames of a specific presentation set. These inter-frame dependencies are captured by *dependency sets*, leading to the notion of *closed presentation sets*.

Afterwards, static and dynamic *relevance functions* are defined as a means to quantify the relevance of frames contained in a particular presentation set. While static relevance functions are used to assign relevance values to frames surrounding a static *reference frame* (representing, e. g., a bookmark), dynamic relevance functions are needed to compute the relevance values of frames surrounding the *current presentation point* which is constantly moving in time during a normal presentation of the video. Both static and dynamic relevance functions are based on *generic relevance functions* which define relevance values independent of a particular reference frame or the current presentation point.

Finally, a *global relevance function* is introduced which combines the relevance values of static and dynamic relevance functions into a single overall relevance value for each frame of the video which will be used by the MPEG-L/MRP algorithm to determine preloading candidates and replacement victims.

Remark: For readers familiar with the details of the original L/MRP model [12], it should be noted that the formal model evolved in several aspects in order to adapt it to the special requirements of the MPEG video format. In particular, the notion of *interaction sets* containing *pairs* of frames (or COPUs) and relevance values (determined by so called *distance relevance functions*) has been split into two orthogonal concepts: *presentation sets* containing frames only on the one hand, and *relevance functions* assigning relevance values to frames on the other hand. By that means, inter-frame dependencies can be captured quite easily by introducing dependency sets which are completely independent of the concept of relevance values. Furthermore, *generic relevance functions*, which are translated to a particular frame and restricted to a particular presentation set in order to obtain static and dynamic relevance functions, are somewhat easier to use than the corresponding *distance relevance functions* of the original model, especially when frames are not equidistantly distributed within a presentation set.

Preliminary Definitions

Let, as usual, $\mathbb{N} = \{1, 2, 3, \dots\}$ be the set of natural numbers and $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$ the set of integer numbers. For $k \in \mathbb{Z}$, let $\mathbb{Z}_k$ denote the set of integers from 0 to k, i. e.,

$$\mathbb{Z}_k = \begin{cases} \{0, 1, \dots, k-1, k\} & \text{for} \quad k \geq 0, \\ \{k, k+1, \dots, -1, 0\} & \text{for} \quad k < 0. \end{cases}$$

For a subset $M \subseteq \mathbb{Z}$ of integers, let $\chi_M : \mathbb{Z} \to \{0, 1\}$ be the *characteristic function* of M assigning a value of 1 to all members of M and 0 to all other numbers:

$$\chi_M(x) = \begin{cases} 1 & \text{for } x \in M, \\ 0 & \text{otherwise.} \end{cases}$$

Presentation Sets

For a particular video comprising $n \in \mathbb{N}$ frames, let

$$F = \{0, 1, \dots, n-1\}$$

be the set of its *frame numbers* in display order. Furthermore, let I, P, and B be pairwise disjoint subsets of F representing the set of all I, P, and B frames of the video, respectively. Assuming that a video does not contain other frame types (in particular, D frames), it holds:

$$F = I \cup P \cup B.$$

A *presentation set* is a subset $S \subseteq F$ of frames which have to be displayed for a particular kind of presentation of the video. For instance, the presentation set

$$F_s = \{f \in F \mid f = i \cdot s, i \in \mathbb{N}_0\} = \{0, s, 2s, \dots\}$$

specifies the set of all frames f which have to be displayed for a forward or backward presentation of the video with a relative speed (or *skip factor*) of $s \in \mathbb{N}$.

Dependency Sets

Due to inter-frame dependencies, in order to be able to decode and display the frames of a particular presentation set S, it might be necessary, however, to decode additional frames. These inter-frame dependencies are captured by the *dependency set* $D(f) \subseteq F$ containing all frames $g \in F$ which are directly or transitively needed to decode and display frame $f \in F$. Using the auxiliary definitions

$$I(f) = \max\{g \in I \mid g \leq f\}$$

and

$$P(f) = \min\{g \in I \cup P \mid g \geq f\}$$

specifying the *closest preceding I frame* of frame f and the *closest succeeding I or P frame* of frame f, respectively, $D(f)$ can be defined as follows:

$$D(f) = \{f\} \cup \{g \in I \cup P \mid I(f) \leq g \leq P(f)\}.$$

Since $I(f) = f = P(f)$ for an I frame $f \in I$, it holds $D(f) = f$ in that case, which means that no additional frame is needed to decode an I frame. For a P frame $f \in P$ it holds $I(f) < f = P(f)$, and thus $D(f)$ contains f and all preceding P frames up to and including the closest preceding I frame. The same holds for a B frame $f \in B$, but since $I(f) < f < P(f)$ in that case, $D(f)$ contains the closest succeeding I or P frame of f, too.

Remark: For $I(f)$ and $P(f)$ to be well-defined for all frames $f \in F$, the first frame of a video must be an I frame and its last frame must be an I or P frame. Without these restrictions, the video would not be standard-conforming, however.

Closed Presentation Sets

The *closure* $\bar{S}$ of a presentation set $S \subseteq F$ can be defined as the set

$$\bar{S} = \bigcup_{f \in S} D(f)$$

containing all frames which are actually needed for a particular kind of presentation, either directly because they have to be displayed or indirectly due to inter-frame dependencies. A presentation set S is called *closed*, if $S = \bar{S}$ holds.

Given the definition of F_s above, the closure $\bar{F}_s$ comprises, for example, all frames which are actually needed for a presentation of the video with a relative speed of s. Intersecting $\bar{F}_s$ with one of the sets I, P, or B, yields the pairwise disjoint sets $I_s = \bar{F}_s \cap I$, $P_s = \bar{F}_s \cap P$, and $B_s = \bar{F}_s \cap B$ containing all I, P, or B frames, respectively, necessary for such a presentation.

If the coding scheme of a video is a constant repetition of the pattern illustrated in Figure 3 (i) (followed by a final I frame), the presentation set F_2 contains all frames depicted as shaded boxes. Since this set comprises all I and P frames of the video, it is already closed, i. e., it holds $\bar{F}_2 = F_2$ in that case. The presentation set F_3 on the other hand, illustrated in Figure 3 (ii) is not closed, since additional P frames identified by black arrows are needed to decode the B frames that are to be presented at a skip factor of 3. That means, that the closure $\bar{F}_3$ contains 6 instead of 4 frames out of each 12-frame pattern IBBBPBBBPBBB resulting in an overhead of roughly 50%.[2]

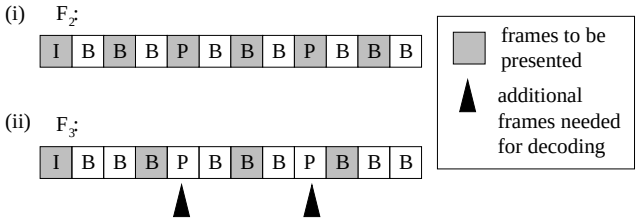

Figure 3: Presentation sets at different skip rates

Relevance Functions

A *generic relevance function* is a function $\rho : \mathbb{Z} \to [0,1]$ assigning a *relevance value* $\rho(x) \in [0,1]$ to each integer number $x \in \mathbb{Z}$. Typically, a generic relevance function is either *monotonously increasing* for $x \leq 0$ and zero-valued for $x > 0$ or zero-valued for $x < 0$ and *monotonously decreasing* for $x \geq 0$. For instance, the linear functions

$$\lambda_a^b(x) = \begin{cases} \max(b \cdot (1 - x/a), 0) & \text{for } x \in \mathbb{Z}_a, \\ 0 & \text{otherwise,} \end{cases}$$

with a peak value of $b \in [0,1]$ for $x = 0$ and positive values for $x \in \mathbb{Z}_a \setminus \{a\}$ $(a \in \mathbb{Z})$ are typical examples of generic relevance functions (cf. Figure 4 (i) for $a \geq 0$ and (ii) for $a \leq 0$).

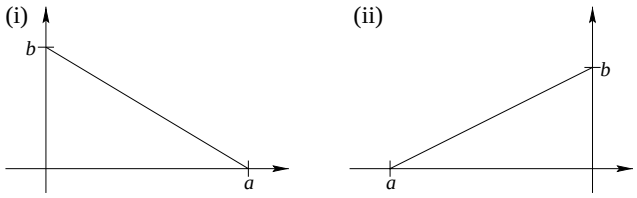

Figure 4: Typical generic relevance functions

A *static relevance function* is a function $\sigma : F \to [0,1]$ assigning a relevance value $\sigma(f) \in [0,1]$ to each frame $f \in F$. Typically, a static relevance function σ is constructed by *translating* the peak of a generic relevance function ρ to a specific *reference frame* $r \in F$ and *restricting* its domain to a particular presentation set $S \subseteq F$:

$$\sigma(f) = \rho(f - r) \cdot \chi_S(f).$$

A *dynamic relevance function* is a function $\delta : F \times F \to [0,1]$ assigning a relevance value $\delta(f,p) \in [0,1]$ to each frame $f \in F$ taking into account the current presentation point $p \in F$. Similar to a static relevance function, a dynamic relevance function is usually constructed by translating and restricting a generic relevance function ρ, where the current presentation point p replaces the static reference frame r:

$$\delta(f,p) = \rho(f - p) \cdot \chi_S(f) \quad \text{for some} S \subseteq F.$$

As noted above, static relevance functions are typically used to describe bookmarks where the reference frame r specifies the position of the bookmark in the video, while dynamic relevance functions are needed to model dynamic presentations of the video like normal playback, reverse playback, fast forward, etc., where the current presentation point p is constantly moving in time.

Global Relevance Function

Given a set of static relevance functions $\sigma_1, \ldots, \sigma_k$ and a set of dynamic relevance functions $\delta_1, \ldots, \delta_m$, the *global relevance function* $\gamma : F \times F \to [0,1]$ is defined as an appropriate combination of these functions, e. g., by computing a *weighted maximum value* for each frame $f \in F$:

$$\gamma(f,p) = \max\left(\max_{i=1,\ldots,k} \omega_i \cdot \sigma_i(f), \max_{j=1,\ldots,m} \pi_j \cdot \delta_j(f,p) \right).$$

Here, the weighting factors $\omega_1, \ldots, \omega_k \in [0,1]$ and $\pi_1, \ldots, \pi_m \in [0,1]$ can be used as global regulators similar to the slide controls of a sound mixer.

[2] Since the sizes of I, P, and B frames are usually quite different, this is indeed only a rough estimation.

Examples

Example 1 – Forward Presentations

For a skip factor $s \in \mathbb{N}$, let $\vec{\rho}_s^I$, $\vec{\rho}_s^P$, and $\vec{\rho}_s^B$ be generic relevance functions[3] which are zero-valued for $x < 0$ and monotonously decreasing for $x \geq 0$. Since I frames are generally more important than P frames which are in turn more important than B frames, the relationship $\vec{\rho}_s^I(x) \geq \vec{\rho}_s^P(x) \geq \vec{\rho}_s^B(x)$ shall hold for all $x \in \mathbb{Z}$ (cf. Figure 5 for a typical example using a skip factor $s = 1$).

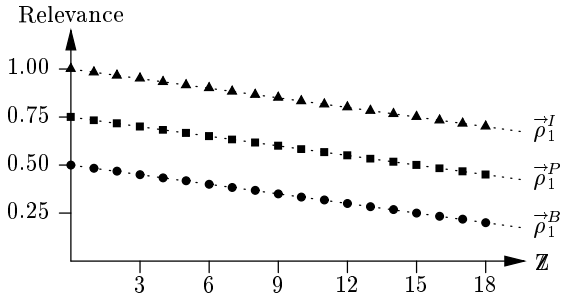

Figure 5: Generic relevance functions $\vec{\rho}_1^I$, $\vec{\rho}_1^P$, $\vec{\rho}_1^B$

Based on these generic relevance functions, dynamic relevance functions $\vec{\delta}_s^I$, $\vec{\delta}_s^P$, and $\vec{\delta}_s^B$ can be defined using the presentation sets I_s, P_s, and B_s, respectively, defined earlier (cf. Figure 6, again using $s = 1$):

$$\begin{aligned} \vec{\delta}_s^I(f,p) &= \vec{\rho}_s^I(f-p) \cdot \chi_{I_s}(f), \\ \vec{\delta}_s^P(f,p) &= \vec{\rho}_s^P(f-p) \cdot \chi_{P_s}(f), \\ \vec{\delta}_s^B(f,p) &= \vec{\rho}_s^B(f-p) \cdot \chi_{B_s}(f). \end{aligned}$$

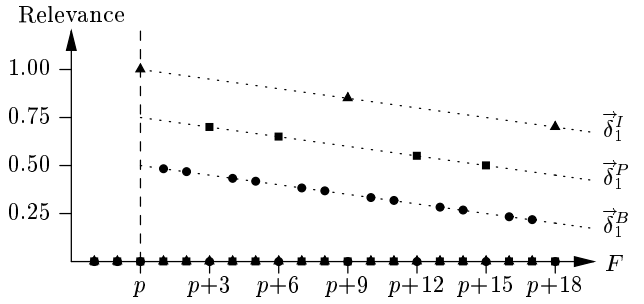

Figure 6: Dynamic relevance functions $\vec{\delta}_1^I$, $\vec{\delta}_1^P$, $\vec{\delta}_1^B$

Depending on the current presentation point p, these functions express the relevance values of those I, P, and B frames, respectively, which will be needed in the near future for a forward presentation of the video with a skip factor of s. Since the presentation sets I_s, P_s, and B_s are pairwise disjoint, at most one of the values $\vec{\delta}_s^I(f,p)$, $\vec{\delta}_s^P(f,p)$, and $\vec{\delta}_s^B(f,p)$ will be positive for each frame $f \in F$, while the others are zero.

Combining the functions $\vec{\delta}_1^I$, $\vec{\delta}_1^P$, and $\vec{\delta}_1^B$ into a global relevance function using weight factors $\vec{\pi}_1^I = \vec{\pi}_1^P = \vec{\pi}_1^B = 1$, yields the function $\vec{\gamma}_1$ shown in Figure 7:

$$\vec{\gamma}_1(f,p) = \max\left(\vec{\delta}_1^I(f,p), \vec{\delta}_1^P(f,p), \vec{\delta}_1^B(f,p)\right)$$

[3] the $\vec{\ }$ denotes the direction of the playout

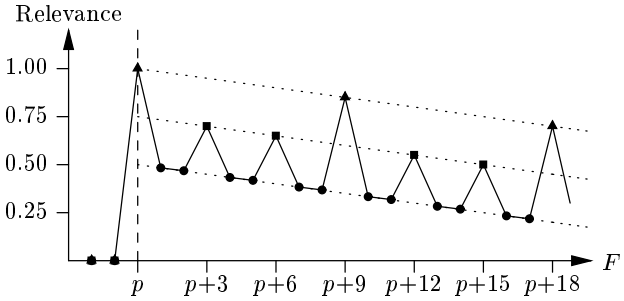

Figure 7: Global relevance function $\vec{\gamma}_1$

Example 2 – Backward Presentations

Replacing the generic relevance functions $\vec{\rho}_s^I$, $\vec{\rho}_s^P$, and $\vec{\rho}_s^B$ with their reflected counterparts $\overleftarrow{\rho}_s^I$, $\overleftarrow{\rho}_s^P$, and $\overleftarrow{\rho}_s^B$, respectively, which are monotonously increasing for $x \leq 0$ and zero-valued for $x > 0$, yields analogous dynamic relevance functions

$$\overleftarrow{\delta}_s^T(f,p) = \overleftarrow{\rho}_s^T(f-p) \cdot \chi_{T_s}(f) \quad \text{for } T = I, P, B$$

expressing the relevance values of those T frames which will be needed for a backward presentation of the video in the near future (cf. Figure 8 for $s = 1$).

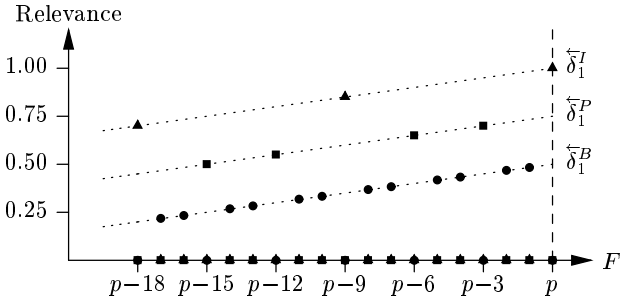

Figure 8: Dynamic relevance functions $\overleftarrow{\delta}_1^I$, $\overleftarrow{\delta}_1^P$, $\overleftarrow{\delta}_1^B$

Remark: Since the frames $g \in D(f) \setminus \{f\}$ are necessary to decode frame f, their relevance values should be greater than that of f. For the "forward functions" $\vec{\delta}_s^T$ described above, this requirement is normally fulfilled due to the monotony of the generic functions $\vec{\rho}_s^T$ and the global relationship $\vec{\rho}_s^I \geq \vec{\rho}_s^P \geq \vec{\rho}_s^B$. The only critical case here is a P frame $g \in D(f)$ *succeeding* a B frame f. By choosing functions $\vec{\rho}_s^P$ and $\vec{\rho}_s^B$ whose difference is large enough, it can be guaranteed, however, that the desired condition $\vec{\delta}_s^P(g) > \vec{\delta}_s^B(f)$ is always satisfied.

For the "backward function" $\overleftarrow{\delta}_s^P$, however, the monotony of the generic function $\overleftarrow{\rho}_s^P$ is partially counter-productive, since it yields the undesired relationship $\overleftarrow{\delta}_s^P(g) \leq \overleftarrow{\delta}_s^P(f)$ for a P frame $f \in P$ and all frames $g \in D(f) \subseteq P$. To remedy this flaw, a correction term can be included in the definition of $\overleftarrow{\delta}_s^P$ which reduces the relevance value of a frame $f \in P$ by a small amount ε for every frame $g \in (D(f) \cap P_s) \setminus \{f\}$ that is necessary to decode f (cf. Figure 9):

$$\begin{aligned} \overleftarrow{\delta}_s^P(f,p) = &\left(\overleftarrow{\rho}_s^P(f-p) - k \cdot \varepsilon\right) \cdot \chi_{P_s}(f) \\ &\text{where } k = |D(f) \cap P_s| - 1. \end{aligned}$$

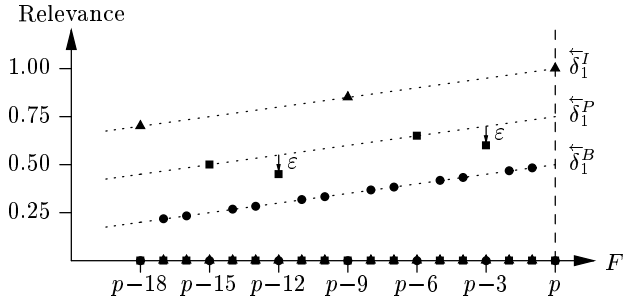

Figure 9: Dynamic relevance functions $\overleftarrow{\delta}_1^I$, $\overleftarrow{\delta}_1^P$, $\overleftarrow{\delta}_1^B$ with $\overleftarrow{\delta}_1^P$ modified

Combining the forward functions $\vec{\delta}_1^I$, $\vec{\delta}_1^P$, and $\vec{\delta}_1^B$ and the backward functions $\overleftarrow{\delta}_1^I$, $\overleftarrow{\delta}_1^P$, and $\overleftarrow{\delta}_1^B$ into a globale relevance function using weight factors $\vec{\pi}_1^I = \vec{\pi}_1^P = \vec{\pi}_1^B = 1$ and $\overleftarrow{\pi}_1^I = \overleftarrow{\pi}_1^P = \overleftarrow{\pi}_1^B = 0.75$ expressing that the overall relevance of backward playing should be weighted 0.75 compared with the overall relevance of forward playing, yields the function $\overleftrightarrow{\gamma}_1$ shown in Figure 10:

$$\overleftrightarrow{\gamma}_1(f,p) = \max(\vec{\gamma}_1(f,p), \\ 0.75 \cdot \overleftarrow{\delta}_1^I(f,p), 0.75 \cdot \overleftarrow{\delta}_1^P(f,p), 0.75 \cdot \overleftarrow{\delta}_1^B(f,p))$$

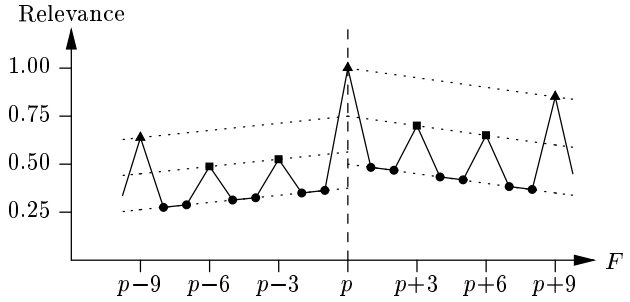

Figure 10: Global relevance function $\overleftrightarrow{\gamma}_1$

Example 3 – Bookmarks

Setting a bookmark at a specific frame $b \in F$ should allow a user to jump to that frame at any time and continue the presentation of the video at that point. Therefore, frames surrounding a bookmark frame b should have the same relevance values as those surrounding the current presentation point p. Thus, depending on the presentation directions (forward/backward) and the skip factors $s \in \mathbb{N}$ which should be allowed after jumping to a bookmark, static relevance functions $\vec{\sigma}_s^T$ and possibly $\overleftarrow{\sigma}_s^T$ should be defined for each bookmark frame b based on the generic relevance functions $\vec{\rho}_s^T$ and $\overleftarrow{\rho}_s^T$, respectively, introduced above:

$$\vec{\sigma}_s^T(f) = \vec{\rho}_s^T(f-b) \cdot \chi_{T_s}(f) \quad \text{for } T = I, P, B,$$
$$\overleftarrow{\sigma}_s^T(f) = \overleftarrow{\rho}_s^T(f-b) \cdot \chi_{T_s}(f) \quad \text{for } T = I, P, B.$$

In order to include these functions into a global relevance function like, e. g., $\overleftrightarrow{\gamma}_1$, their weighting factors $\vec{\omega}_s^T$ and $\overleftarrow{\omega}_s^T$ should depend on factors like the frequency bookmarks are jumped to, the number of bookmarks which have been set, and the available buffer size. If, for instance, the number of bookmarks is small with respect to the buffer size and bookmarks are referenced frequently, factors $\vec{\omega}_s^T = \overleftarrow{\omega}_s^T = 1$ might be sensible in order to give bookmarks the same overall relevance as the current presentation point. If, on the other hand, the number of bookmarks is very large, smaller weighting factors have to be chosen in order to avoid that bookmark frames completely push out frames which are relevant for the current presentation.

4.3 MPEG-L/MRP Algorithm

Preloading and Replacement of Frames

The MPEG-L/MRP algorithm, which is based on the buffer management algorithm presented in [12], integrates preloading and replacement of frames. Using the global relevance function γ, those frames of an MPEG video are determined that are to be loaded next as they are most relevant for the presentation. If the buffer is full, the global relevance function is also used to determine those frames that are to be removed to make room for more relevant ones. Listing 4.1 shows the essential parts of the algorithm which are explained in the following.

The buffer containing the currently loaded frames is represented by an object b of type `Buffer` supplying methods `full()`, `load()`, `toss()`, and `undo()` explained below. To simplify notations, b is also used as a set comprising all currently loaded frames.

The function `LoadMostRelevantFrames()`, which is called whenever the current presentation point p changes, maintains two frames, l and t, with corresponding relevance values, L and T (cf. Figure 11). The *load threshold* L represents the maximum relevance value $\max_{f \in F \setminus b} \gamma(f,p)$ of all frames $f \in F \setminus b$ of the movie which are currently *not* loaded, while the *toss threshold* T represents the minimum relevance value $\min_{f \in b} \gamma(f,p)$ of all frames $f \in b$ which *are* currently loaded. Thus it is known, that all frames with a global relevance value greater than L are already loaded, while those having a global relevance value lower than T are not loaded. The *load candidate* $l \in F \setminus b$ and the *toss victim* $t \in b$ are frames with global relevance values $\gamma(l,p) = L$ and $\gamma(t,p) = T$, respectively. In the snapshot shown in Figure 11, it holds for example $l = p + 9$ and $t = p - 5$.

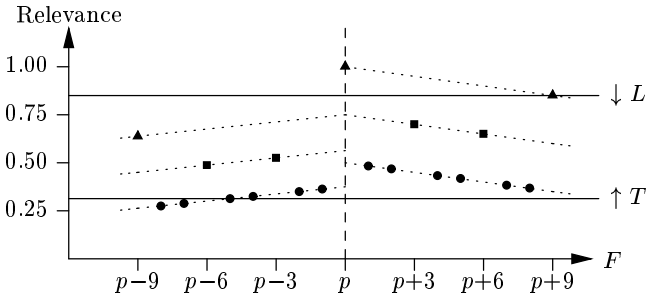

Figure 11: Load and toss thresholds L and T

The function repeatedly selects a load candidate l (line 5) and loads it into the buffer (line 11), causing the load threshold $L = \max_{f \in F \setminus b} \gamma(f,p)$ to gradually decrease. As long as there is not enough buffer space for the load candidate l, however (line 6), a toss victim t is selected (line 7) and tossed out of

the buffer (line 9), causing the toss threshold $T = \min_{f \in b} \gamma(f,p)$ to gradually increase. Since frame sizes might vary heavily, it might indeed be necessary to toss several frames t out of the buffer before being able to load the load candidate l.

Listing 4.1 LoadMostRelevantFrames

```
1  Buffer b;
2
3  LoadMostRelevantFrames(int p) {
4      while (true) {
5          let l ∈ F \ b where γ(l,p) = max_{f∈F\b} γ(f,p);
6          while (b.full(l)) {
7              let t ∈ b where γ(t,p) = min_{f∈b} γ(f,p);
8              if (γ(t,p) > γ(l,p)) { b.undo(); return; }
9              b.toss(t);
10         }
11         b.load(l);
12     }
13 }
```

If the toss threshold $T = \gamma(t,p)$ becomes greater than the load threshold $L = \gamma(l,p)$ (line 8), i. e., load and toss thresholds meet each other, a stable state is reached causing the function to return without loading the current load candidate l anymore. In that case, one or more toss victims t might already have been tossed out of the buffer in order to make room for l. Since the available buffer space is still not sufficient for l, however, these victims might be put back into the buffer in order to avoid unused buffer space. For that reason, the method call `b.toss(t)` (line 9) only *tentatively* tosses the frame t, e. g., by *marking* it for deletion, while the actual tossing is performed implicitly by the next call to `b.load(l)` (line 11). Alternatively, the call to `b.undo()` (line 8) undoes all tentative tossing operations, e. g., by unmarking all marked frames, causing them to remain in the buffer.

Remark: Even though `b.toss(t)` does not actually toss frame t out of the buffer, it will be treated as removed by subsequent calls to `b.full()`, of course. Furthermore, t is virtually removed from the set b to avoid that it becomes the toss victim in line 7 again and again.

Optimizations

The algorithm presented so far, especially the selection of the load candiate l and the toss victim t (line 5 and 7, respectively), is described in a rather abstract way to make it more comprehensible. Hence, it is not directly usable for a real-world implementation, since a straight-forward implementation of the "where clauses" $\gamma(l,p) = \max_{f \in F \setminus b} \gamma(f,p)$ and $\gamma(t,p) = \min_{f \in b} \gamma(f,p)$ would require a loop over all frames $f \in F \setminus b$ and $f \in b$, respectively, and a computation of the global relevance value $\gamma(f,p)$ for each such frame f. Given a typical frame rate of 20 to 30 frames per second, a 10 minute movie comprises 12,000 to 18,000 frames for which $\gamma(f,p)$ would have to be computed. Fortunately, the typical properties of generic relevance functions mentioned before, allow a significant reduction of the number of necessary computations, as will be briefly explained in the following.

Consider, for example, a generic relevance function $\rho(x)$ which is zero-valued for $x < 0$ and monotonously decreasing for $x \geq 0$, and an arbitrary set $M \subseteq \mathbb{Z}$ of integers which is partitioned into two subsets $M^- = \{x \in M \mid x < 0\}$ and $M^+ = \{x \in M \mid x \geq 0\}$ containing negative and non-negative members of M, respectively. If $l = \min M^+$ and $r = \max M^+$ denote the borders of the subset M^+, then the extreme values $\min_{x \in M} \rho(x)$ and $\max_{x \in M} \rho(x)$ of the generic relevance function ρ can be computed as follows:

$$\min_{x \in M} \rho(x) = \begin{cases} 0, & \text{if } M^- \neq \emptyset, \\ \rho(r), & \text{if } M^- = \emptyset, \end{cases}$$

$$\max_{x \in M} \rho(x) = \begin{cases} \rho(l), & \text{if } M^+ \neq \emptyset, \\ 0, & \text{if } M^+ = \emptyset. \end{cases}$$

That means, that at most one function value $\rho(x)$ has to be computed in order to determine the minimum or maximum value of ρ over an arbitrarily large set M.[4]

Applied to a dynamic relevance function $\delta(f,p)$ which is constructed by translating and restricting a generic relevance function ρ to a presentation set $S \subseteq F$, this means, that at most one function value $\delta(f,p)$ has to be computed in order to determine the minimum or maximum value of δ over an arbitrary subset of S. In particular, if $b \subseteq F$ is any subset of frames, $\min_{f \in S \cap b} \delta(f,p)$ and $\max_{f \in S \setminus b} \delta(f,p)$ can be determined by computing at most one function value $\delta(f,p)$.[5]

Finally, if a global relevance function γ is defined as the weighted maximum of a set of dynamic relevance functions $\delta_1, \ldots, \delta_m$ with corresponding presentation sets $S_1, \ldots, S_m$ and weighting factors $\pi_1, \ldots, \pi_m$, its maximum value $\max_{f \in F \setminus b} \gamma(f,p)$ can be determined by simply comparing the weighted maximum values $\pi_j \cdot \max_{f \in S_j \setminus b} \delta(f,p)$ of the dynamic functions δ_j $(j = 1, \ldots, m)$. Since each of these maximum values, as explained above, requires at most one function evaluation, the global maximum value needed in line 5 of the algorithm (Listing 4.1) can be computed very efficiently, too.

If the presentation sets $S_1, \ldots, S_m$ are pairwise disjoint, it can be shown that the minimum value $\min_{f \in b} \gamma(f,p)$ of the global relevance function γ, needed in line 7 of the algorithm, can also be determined by comparing the weighted minimum values $\pi_j \cdot \min_{f \in S_j \cap b} \delta_j(f,p)$ of the dynamic functions δ_j $(j = 1, \ldots, m)$ and thus can be computed very efficiently,

[4] Due to symmetry properties, the same holds for a generic relevance function ρ which is monotonously increasing for $x \leq 0$ and zero-valued for $x > 0$.

[5] If a dynamic relevance function $\overleftarrow{\delta}_s^P$ is modified as shown in Figure 9, this statement does no longer hold: Since the function's monotony is partially violated, its extreme values over a set of frames might not occur *exactly* at, but only *near* the borders of the set. Nevertheless, they can be determined by computing *a few* function values near the borders, where the exact number depends on the maximum number of consecutive P frames in the video.

too. Otherwise, things become a bit more difficult, since a frame $f \in b$ might belong to more than one presentation set S_j causing it to possibly possess multiple positive relevance values $\delta_j(f,p)$. If one of them represents the global minimum value over all frames $f \in b$ and all relevance functions δ_j, it would be selected as the toss victim, despite the fact that it might possess another, greater dynamic relevance, too. In order to avoid this problem, *virtual frames* are introduced as pairs $(f,j) \in F \times \{1,\ldots,m\}$ belonging to at most one "virtual presentation set" $S_j \times \{j\}$ (which is isomorphic to the real presentation set S_j). Technically speaking that means, that the buffer b might contain multiple *references* (representing virtual frames) to the same physical frame $f \in F$, and that a physical frame is only tossed out of the buffer if the last virtual frame which references it is tossed.

In principle, static relevance functions needed to support bookmarks can be treated completely analogously to dynamic functions. If their number becomes large, however, even more sophisticated optimizations can be employed. Since static relevance values do not change with the current presentation point, it is possible to precompute and store them in a global sorted list which can be treated like a *single* monotonous function.

Video Stream Construction

In parallel with the preload and replacement function just described, a video stream construction function `ConstructVideoStream()` is executed which re-assembles the individual frames contained in the buffer into a coherent MPEG video stream which can be handed to a player. Typically, this function is called by the player (via an appropriate interface) whenever it needs another group of pictures to display, i. e., when the current presentation point p has changed. Therefore, `ConstructVideoStream()` in turn calls `LoadMostRelevantFrames()` in a parallel thread in order to update the buffer contents. If `LoadMostRelevantFrames()` has not finished when `ConstructVideoStream()` is called the next time, it is interrupted and called again using the new current presentation point p.

4.4 Temporal Adaptation

When applying the MPEG-L/MRP buffering technique for transmitting an MPEG video over a network connection, it might happen that some frames cannot be used by an MPEG-player, be it because they are not delivered in time or because they are broken due to transmission failures. Many existing MPEG players are not able to cope with missing frames or at least cannot maintain a timely or continuous presentation, respectively. Therefore, the player must be provided with an MPEG stream in which the missing frames are replaced by corresponding surrogate frames. The structure of the MPEG format allows to define a frame that is able to reference the whole content of a preceding I or P frame. By providing such a frame as a surrogate for a missing one, a correct stream can be constructed that maintains the time synchronous playout. However, to maintain the inter-frame dependencies of the original stream, it must be assured that each of the surrogate frames is inserted at the right position in the stream. By constructing a syntactically correct MPEG video stream with surrogate frames, any available MPEG player can present the modified video. In our implementation, this kind of temporal adaptation is applied to complete the MPEG stream when frames do not arrive in time and deliver the adapted stream to the MPEG player.

4.5 Physical Storage Management

In order to efficiently realize the buffering of frames in the MPEG-L/MRP algorithm, an adequate physical storage management has to be employed. Due to the high variability of frame sizes in MPEG videos, this is a non-trivial task. We analyzed several memory management strategies in search for an appropriate storage management of MPEG frames in the client's memory. The requirements a good physical storage manager must meet are efficient management and good utilization of the buffer and a high degree of reusage of freed memory. Therefore, a dynamic management of the buffer is indispensable. A well-known technique for dynamic buffer management is the binary buddy algorithm [9]. Peterson and Norman [13] analyzed different variants of buddy systems (binary, Fibonacci, and weighted) and showed that they all suffer from a high fragmentation. Due to this reason, a more suitable memory management algorithm had to be found. In a simulation, we compared a bunch of buddy systems, namely DTSS buddy [10], Starburst buddy [11], exact buddy, and as a reference system the binary buddy. The simulation evaluated the internal and external fragmentation, reusability of the memory, and the number of used buddy segments. We made diverse test runs with different MPEG videos and varying parameters. In a nutshell one can say, that the Starburst buddy was the most suitable algorithm for storing MPEG frames in a client buffer. Although it performs slightly worse than the genuine buddy system, it is more efficient in the utilization and the reusability of the memory. So we decided to implement the physical storage management of the MPEG-L/MRP algorithm based on the Starburst buddy strategy.

5. IMPLEMENTATION

Since L/MRP has been developed as a typical client/server architecture, we have implemented the MPEG-L/MRP strategy employing a client/server architecture, too. The client buffer following the MPEG-L/MRP buffer management strategy has been implemented using Java and the Java Media Framework (JMF) [16]. The JMF offers an easy to use and extendable API for the presentation of continuous media. We realized the client buffer providing an interface that meets the requirements of the JMF players such that the standard MPEG player of JMF can be connected. The client buffer requests the data from a server corresponding to our strategy and delivers a correct MPEG-1 data stream to the player. On the server side we employ the Informix Dynamic Server / Universal Data Option (IDS/UD) to manage – among media of other types – the MPEG videos. Due to its extensibility, manifesting in so called DataBlades, the IDS/UD can be extended to support frame-based access to MPEG videos. On top of the IDS/UD we implemented a server in Java to perform the task of transfering the requested frames from the media server to the client buffer.

6. CONCLUSION

In this paper, we presented the MPEG-L/MRP buffering technique providing *interactive* and *adaptive* streaming of

MPEG videos that exploits MPEG-specific features such as different frame types with different importance and their inter-frame dependencies.

The major focus of the strategy is the suitable support for *interactive* video streaming. With this, applications like our multimedia information system in cardiac surgery, that delivers interactive multimedia learning material over the Internet, can be supported in such a way that a user feels much more comfortable with the system. With MPEG-L/MRP, we deliver a much better *Quality of Presentation* due to the smooth reactions to the expected frequent user interactions. Consequently, we deliver a much better *Quality of Information* to the user who – not to be forgotten – might have to pay for the multimedia material presented. Additionally, the different frame types of MPEG open a new possibility, compared to homogeneous streams like Motion-JPEG, to adapt the stream to different bandwidths. Besides the interaction, we presented how the MPEG-L/MRP algorithm exploits the different frame types and the different importance of the frames for an *adaptive* delivery of frames to the presentation client. Another advantage of the employment of a standard video format like MPEG-1 is, that one is no longer bound to the proprietary video formats of commercial applications. Since MPEG-2 offers further possibilities for adaptation, we consider applying an adapted version of MPEG-L/MRP to this format.

As we are concerned with the construction of multimedia repositories, MPEG-L/MRP forms one building block of such a repository that provides comprehensive support for different kinds of multimedia information systems. On each architectural level and on each granularity of the components of the repository envisioned, we try to build modules that follow the overall goal to support multimedia applications that, with regard to the quality of information and with regard to the quality of delivery and presentation, is optimally adapted to the user context and system environment. In this context, to please a user with a continuous, smooth, and best quality presentation, MPEG-L/MRP is an important module to provide smooth interactive delivery of MPEG-1 videos.

Acknowledgements We would like to thank Joachim Wiedmann for his contributions to the design and implementation of the MPEG-L/MRP strategy and Michael Menth for his valuable comments on the topic.

7. REFERENCES

[1] Apple. Quicktime streaming. http://developer.apple.com/techpubs/quicktime/qtdevdocs/RM/pdfframe.html.

[2] Emblaze. Streaming Audio and Video – Solutions over IP Networks. http://www.emblaze.com.

[3] P. Halvorsen, V. Goebel, and T. Plagemann. Q-L/MRP: A Buffer Management Mechanism for QoS Support in a Multimedia DBMS. In *International Workshop on Multimedia Database Management Systems*, Dayton, 1998.

[4] C. K. Hess and R. H. Campbell. Media Streaming Protocol: An Adaptive Protocol for the Delivery of Audio and Video over the Internet. In *Proc. of IEEE International Conference on Multimedia Computing and Systems (ICMCS'99)*, Florence, Italy, June 1999.

[5] S. Hollfelder and H.-J. Lee. Data Abstractions for Multimedia Database Systems. Technical Report 1075, GMD, Darmstadt, Germany, 1997.

[6] ISO. *Information Technology – Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1,5 Mbit/s – Part 1: Systems.* ISO/IEC 11172-1, 1993.

[7] ISO. *Information Technology – Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1,5 Mbit/s – Part 2: Video.* ISO/IEC 11172-2, 1993.

[8] W. Klas, C. Greiner, and R. Friedl. Cardio-OP — Gallery of Cardiac Surgery. In *Proc. of IEEE International Conference on Multimedia Computing and Systems (ICMCS'99)*, Florence, Italy, June 1999.

[9] D. E. Knuth. *The Art Of Computer Programming*, volume 1: Fundamental Algorithms. Addison-Wesley, Reading, Massachusetts, second edition, 1973.

[10] P. D. L. Koch. Disk File Allocation Based on the Buddy System. *ACM Transactions on Computer Systems*, 5(4), 1987.

[11] T. J. Lehman and B. G. Lindsay. The Starburst Long Field Manager. In *Proceedings of the 15th VLDB Conference*, Amsterdam, 1989.

[12] F. Moser, A. Kraiß, and W. Klas. L/MRP: A Buffer Management Strategy for Interactive Continuous Data Flows in a Multimedia DBMS. In *Proceedings of the 21st VLDB Conference*, Zürich, Switzerland, 1995.

[13] J. L. Peterson and T. A. Norman. Buddy Systems. *Communications of the ACM*, 20(6):421–431, 1977.

[14] Real Networks. Real Video Technical White Paper, Feb. 1997. http://www.real.com/devzone/library/whitepapers/overview.html.

[15] Real Networks. SureStream - Delivering Superior Quality and Reliability, Feb. 2000. http://www.realnetworks.com/devzone/library/whitepapers/surestrm.html.

[16] SUN Microsystems. Java Media Framework 2.0, 1999. http://java.sun.com/products/java-media/jmf/2.0.

[17] VDOnet. VDOLive. http://www.vdo.net.

[18] G. K. Wallace. The JPEG Still Picture Compression Standard. *Communications of the ACM*, 34(4):30–44, 1991.

[19] J. Walpole, R. Koster, S. Cen, C. Cowan, D. Maier, D. McNamee, C. Pu, and D. Steere. A Player for Adaptive MPEG Video Streaming over the Internet. In *Proceedings of the 26th Applied Imagery Pattern Recognition Workshop (AIPR-97), SPIE*, Washington DC, USA, October 1997.