REIHE INFORMATIK
5/92

MTP: A Movie Transmission Protocol for Multimedia Applications

B. Lamparter, W. Effelsberg und N. Michl
Universität Mannheim
Seminargebäude A5
6800 Mannheim

MTP: A Movie Transmission Protocol for Multimedia Applications

Bernd Lamparter   Wolfgang Effelsberg   Norman Michl
University of Mannheim, Germany
lamparter@pi4.informatik.uni-mannheim.de

Abstract

Typical color video adapters of today's PCs and workstations use 8 bits per pixel as an index into the color lookup table (CLUT). Full color pictures and movies have to be reduced to 256 colors. In order to avoid false colors between two frames of a digital movie, a novel technique for computing the CLUT's is proposed: A subset of the CLUT entries is reserved for new colors of the next frame. The paper presents an algorithm for the gradual adaption of the color lookup table during the transmission of a movie. First experience is reported in the framework of the XMovie project.

1 Introduction

In multimedia systems the transmission and presentation of digital movies plays a central role. Algorithms and protocols for the efficient transfer and correct representation of digital videos and computer-generated films must be designed and implemented. The XMovie system developed at the University of Mannheim is a testbed for such algorithms and protocols [LE91].

XMovie is a distributed system of UNIX workstations using a modified X Window System for the display of movies. In contrast to other multimedia systems, no special hardware is required. The movies are stored on disk in digital format and transmitted over a digital data network. This facilitates full integration with other media and straightforward portation to other hardware platforms.

Several other research groups investigate distributed multimedia systems, and only a small sample can be mentioned here. In the PANDORA project, a joint effort of University of Cambridge and Olivetti Research in the UK, an architecture for networked multimedia systems is developed. Workstations are interconnected by the Cambridge Fast Ring [HN88]. This ring carries fixed-size cells, and it can be used in ATM mode. The integration of video and audio data streams into the workstations is done by special-purpose hardware, called "Pandora's box". Video presentation is monochrome. Many new insights were gained from the Pandora project, especially in the area of transmission techniques for real-time data streams for audio and monochrome video [Hop90, Hop91].

Distributed multimedia systems are also investigated at the University of Lancaster. Research covers the full range of distributed multimedia computing from low level networking and protocol issues through to end user studies and multimedia applications. An end user study is being carried out in cooperation with a large chemical company (ICI) and is being used to derive a set of requirements for distributed multimedia infrastructures. Most of the research has a highly practical content and concentrates on engineering techniques to build distributed multimedia systems. For example, experimentation is being carried out into the performance characteristics of multimedia protocols. This work is being carried out in close association with the development of a range of multimedia services in a distributed environment. In order to support this work a suitable infrastructure was built. There are two main components to this infrastructure: the network emulator and the multimedia network interface unit. The emulator provides the equivalent performance and functionality to real high performance networks (the initial configuration supports FDDI and ATM-like networks). The emulator provides a test-bed for multimedia experimentation without commitment to a particular network technology [BCDW90, BHS90, BHSS90, CGHS91].

The University of California and ICSI at Berkeley also have major projects on networking support for multimedia systems. Their main emphasis is on the transmission of continuous data streams over packet switched networks [Fer91, GG91], operating system extensions for real-time data streams [ADH91], and programming abstractions for continuous media [ATW$^+$90].

IBM's European Networking Center in Heidelberg also investigates data transmission for audio and video applications. A high-speed transport system called HeiTS was designed and implemented on AIX and OS/2 workstations [SHRS90, HHS$^+$91]. It offers a transport layer service interface over high-speed networks and supports fast and reliable data transfer for continuous-media data streams. Prototype applications include a video distribution service for OS/2 workstations where users can open a movie window on their workstation screen and watch a video broadcast. This system uses DVI cards [Lut89] in all participating workstations.

In contrast to these other projects our XMovie system concentrates on movie support on workstations without special hardware. We expect that the next generation of workstations, in combination with the high-speed network technology currently being standardized, will be powerful enough to support real-time display of films, implemented entirely in software. This offers several advantages:

- immediate availablity on thousands of workstations.
- flexibility to adapt to new formats for digital films (i.e. DVI, MPEG).
- lower cost.

The storage, transmission and display formats for the movies are designed for Color Lookup Table (CLUT) technology. Most PCs and workstations use graphics adapters based on color lookup tables. Each pixel of a digital image is interpreted as an index into the CLUT. Each entry of the CLUT corresponds to a specific color value (RGB mixture). A typical case is the use of 8 bits per pixel, providing 256 entries in the CLUT and thus 256 different colors in one image.

The use of color lookup table technology for movies (i.e. sequences of images rather than single images) requires new techniques and algorithms:

- A technique to update the CLUT continuously during movie display in order to adapt to changes in the color contents of the images. The showing of false colors on the screen must be avoided when CLUT entries are updated.
- An algorithm to compute optimal CLUT updates for a given movie.
- A transmission protocol for continuous sequences of digital images.

These problems are addressed in the Movie Transmission Protocol MTP described in this paper. In Section 2 we present an algorithm for the gradual adaptation of the color lookup table to the changing color contents of the movie frames. Section 3 describes the integration of this algorithm into the XMovie system. Experimental results with the algorithm and the performance of the XMovie system are reported in Section 4. An outlook concludes the paper.

2 The DeltaCLUT Algorithm

There are three possibilities to generate a movie with CLUT out of a movie with full color:

- compute one CLUT for the whole movie
- compute a new CLUT for each frame
- compute a CLUT for the first frame and update it according to the color contents of subsequent frames

The first two methods both have disadvantages. Using only one CLUT for the whole movie is too poor in colors. For a single picture 256 optimally chosen colors are usually sufficient, but not for a movie with very different scenes. In the second method the workstation has to re-load the CLUT between two frames. The old frame is thus shown with the CLUT of the new frame for a short time. That has a visually very disturbing effect. Loading the CLUT after loading the frame results in the same effect: the new frame is shown with the CLUT of the old.

When the third method is used in a straightforward way the same but somewhat weaker effect will be observed. In MTP the third method was extended to avoid this problem (see Figure 1). Each CLUT contains a small amount of free entries, i.e. no pixel of the frame uses these table entries. The next frame can use and change these reserved entries. The actual frame can now be shown with the own CLUT and with the CLUT of the next frame without a visible difference, thus avoiding false colors.

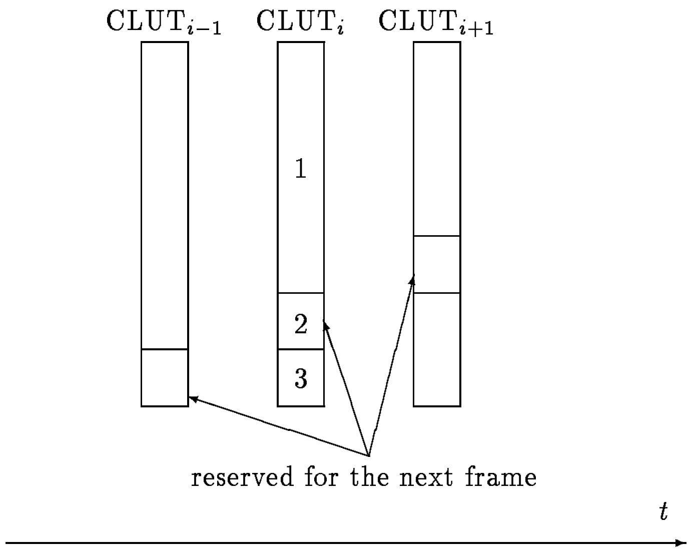

Figure 1: Free entries in the CLUTs

2.1 Computing Optimal CLUT Updates for a Given Movie

We have developed an algorithm called DeltaCLUT for the gradual adaption of the Color Lookup Table [Mic91]. The CLUT for each frame i in the movie is computed as follows. Conceptually each CLUT has three classes of entries: Class 1 comprises the entries for colors identical in frames i-1 and i, class 2 comprises entries needed for frame i-1 but not occuring in frame i, and class 3 comprises new colors of frame i. The algorithm has four steps:

1. Choose a full CLUT for frame i with a conventional CLUT algorithm, for example median-cut [HP89].
2. Compare the colors of CLUT i with the colors of CLUT $i-1$. Identical or nearly identical colors are re-used from CLUT $i-1$ (class 1).
3. Copy all CLUT entries used in frame $i-1$ but not yet in CLUT i to CLUT i (class 2). These colors are not needed by frame i, so that frame $i+1$ can overwrite them later.
4. Choose some more colors for CLUT i and load them into the entries of class 3.

The first frame is a special case. It is handled by reserving a fixed but arbitrary number of entries.

The algorithm can be adapted to a given movie with five parameters: The first parameter is the definition of "nearly identical colors", given as the allowed Euklid distance. The second is a minimal number of entries to copy from the previous frame. The third parameter gives the maximal amount of entries to be reserved, that is the sum of copied and newly choosen colors. The fourth parameter gives the amount of entries available to the algorithm; the difference between this parameter and the full size of the physical colormap gives other applications on the screen a chance to use their own colors. The fifth parameter adjusts the median cut algorithm.

Median cut is a recursive algorithm to find the significant colors of a full color frame. First all colors are collected in a three dimensional space spawned by the three color components. Then the resulting bounding box is divided into two parts along the longest axis. The resulting two boxes contain approximately the same amount of colors. These boxes are now recursively divided like the first. The recursion stops after generating a given amount of boxes (parameter five). The color representing each box is the middle value of all colors in the box. The next step merges these colors according to the following two conditions:

I. merge colors with small differences

II. merge colors representing a small number of original colors.

The first condition prevents the merging of less used, but important colors, for example a line in the frame. The second condition protects frequently used, but nearly identical colors, for example used in a slowly changing background.

The second step of the DeltaCLUT algorithm searches nearly identical colors in CLUT i-1 according to parameter one. The colors found are copied to CLUT i (class 1). In step three all other colors used in frame i-1 are copied to CLUT i, hence all colors used in frame i-1 are now in CLUT i, but the colors copied in step three are marked as not useable (class 3). Later they can be overridden by CLUT i+1 without disturbing frame i. In the fourth step more colors remaining from the output of the median cut algorithm are merged until CLUT i is filled according to parameter three. In these step entries not used in frame i-1 are filled (class 3).

2.2 Interaction with other X Window Applications

When a movie is displayed on a screen together with other application windows, all applications must share the same physical CLUT. If a private CLUT is loaded into the physical CLUT whenever an application is active, other window applications are probably shown in false colors.

One possible way to solve this problem is to use the standard colormap offered by X Windows. But this implies, that all indices have to be resorted during the movie, or, if not enough colors are available, the movie would have to be redithered. To avoid these problems an own colormap is used. In this map the first sixteen entries are left free, because most applications under X Windows are using these entries for frames, menue items, buttons, etc.

3 Integration of the DeltaCLUT algorithm into the XMovie System

3.1 Architecture of the XMovie System

The XMovie System is a distributed test bed for integrated transmission and presentation of digital movies. It consists of interconnected UNIX workstations. In our current implementation, the network is a standard Ethernet. An FDDI-ring has been installed recently, and XMovie is currently being ported to FDDI. The architecture of the system is shown in Figure 2.

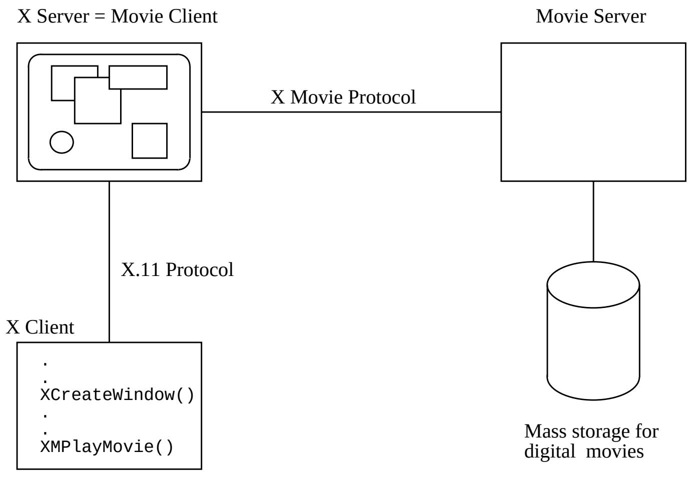

Figure 2: Architecture of the XMovie system

The main components of XMovie are the Movie Server, the Movie Client and the X Client. These three components can run on one, two or three different UNIX systems depending on the requirements of the application.

The **Movie Server** is able to store and replay sequences of digital images (digital films). It maintains a movie directory. On request of the Movie Client a sequence of images is sent over the network to the Movie Client. The transmission protocol is called MTP (Movie Transmission Protocol). It was developed specifically for this purpose. The **Movie Client** is an extended version of the standard X Server of the X Window System. The extension implements a new set of functions for the purpose of displaying movies in a window on the screen. Examples of new functions are XMOpenMovie, XMPlayMovie, and XMShowSinglePicture. The extension was integrated into the source code of the X Window System. The third component of the system is the **X Client**. Similar to the X Server, it is an extension of the standard X Client. The set of new functions mentioned above was integrated into the Xlib function library of the X Client so that a programmer can now invoke the new movie functions just like other Xlib functions.

3.2 Integration of Movies into a Window System

For the workstation user the movie should appear in a window on the screen, without disturbing other windows. The handling of the movie window should be similar to the handling of other windows. Movie windows are not yet supported in current window systems, such as the X Window System. An integration of movie windows can be done in the following ways:

- Hardware can be added (blue-box or realtime digitizing of analog video sources) [Bru89, Lut89].
- The window software is accelerated so that a sequence of still images can be passed to it, and it will draw them at 25 images per second.
- The window system is extended by a movie service.

The goal of the XMovie project is to avoid special hardware; hence we only elaborate on the other two possibilities.

Each window system has a mechanism to draw raster images, so in principle we can show movies by drawing images fast, one after the other [Lof90]. But each image causes considerable overhead because it invokes a large number of subroutines of the window system until it is finally presented. Also the programmer of the client has to control the flow of images explicitly. Therefore we considered this solution to be unacceptable.

A more promising approach is the integration of a movie window as a primitive of the window system. In case of X this implies full control of the movie window by the X Server process. The X Server process manages the connection establishment and release to the Movie Server and controls the movie transmission. We see many advantages compared to the solution above:

- The communication overhead between X Client and X Server is lower.
- The code for movie presentation (i.e. presentation of sequences of images) does not have to be repeated in each client program.
- The X Server can be better adapted to the hardware, so the performance and hence the movie quality is better.
- The manipulation of the movie window (open, close, move, ...) works as usual (for the user and for the programmer).

Thus prototype of the XMovie System is based on the latter solution. The subroutine library of the X Window System (Xlib) was extended to provide the following new functions [Kel91]:

XMListMovies(): Requests the X Server to send a list of stored movies and images

XMFreeMovieInfo(): Frees the memory allocated with **XMListMovies**

XMOpenMovie(): Opens a movie connection

XMPlayMovie(): Starts a movie in a window

XMShowSinglePicture(): Shows a single frame of a movie

XMStopMovie(): Stops a running movie

XMDestroyMovie(): Frees all resources allocated for the movie (memory, communication channels)

3.3 Reliable Transfer

In the MTP protocol a movie consists of a sequence of (ΔCLUT, frame)-pairs. Whereas bit errors in the frame part can be ignored because of the high refresh rate, bit errors in the ΔCLUT part are a problem: when an update to the CLUT is lost in the network, subsequent frames are shown in false colors.

The appropriate solution for this problem is the provision of a reliable, fast and isochronous transport protocol, as proposed in most upcoming standards for high-speed networks [HH91]. An interim solution is the inclusion of forward error recovery for the ΔCLUT part of movie packets; conventional error recovery by retransmission is not appropriate for movies since the isochronous flow of images would be interrupted. We are currently working on a forward error recovery scheme for MTP.

4 Experience

The DeltaCLUT algorithm and the MTP protocol described above were implemented in the framework of the XMovie system. Our workstations are DECstations 5000/120 under ULTRIX, with X Windows Rel. 5.

4.1 Experience with the DeltaCLUT Algorithm

Figures 3 to 7 show statistical results of the reservation of entries with the DeltaCLUT algorithm. The white rectangles are the copied entries (class 1 in Figure 1), the black retangles are the newly choosen colors (class 3). The rest to the 240 line are the entries left free for the next fame (class 2). Difference, Minimum and Maximum are the parameters explained in Section 2.1.

Figures 3 to 6 show a frame sequence causing trouble for the color adaptation. It is a sequence with seven different raytraced frames, created artificially to experiment with color adaption after a cut in a movie. We can see five frames out of a movie showing flying butterflies ("0", ..., "4") two single frames showing a molecule model of the DNA ("D") and two showing fish in an aquarium ("f"). All tests were made with a colormap of size 240. The figures show that very few colors can be re-used after a cut in the movie (Figure 3). Because of that, the algorithm was forced to re-use a minimum of 60 colors. This produces a picture of better quality because the dithering algorithm could now choose out of a greater amount of colors. Figure 4 shows that a maximum of 200 colors is too high. Even the second picture could not fill up the 200 entries with new colors. After the cuts the frames could not choose enough own colors, resulting in lower quality. The best results are achieved with an allowed distance of 10, a minimum of 60 and a maximum of 180 colors (see Figure 6).

Figure 7 shows the statistics of a 100 frame sequence of flying butterflies. The amount of new colors ranges from 12 to 60.

Our experiments showed that the color quality of single raytraced frames is only slightly poorer compared with pictures using all 256 colors.

The DeltaCLUT algorithm is rather time consuming due to the embedded median-cut algorithm. The DeltaCLUT part itself is quite fast. Our workstations use about one hour for the 100 frame sequence. This time is nearly independent of the size of the frames because the amount of different colors in a frame is independent of its size.

The dithering of the frames can be done with any standard dithering algorithm. The time is linear with the number of the pixels in the movie because the algorithm has to find the index for each pixel. The time for 100 frames of size 156$\times$115 is about 15 minutes.

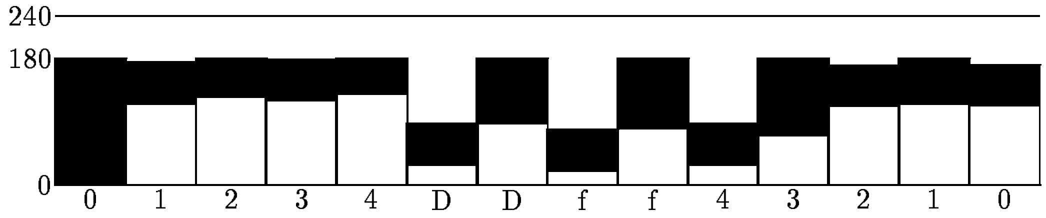

Figure 3: Difference 10, Maximum 180

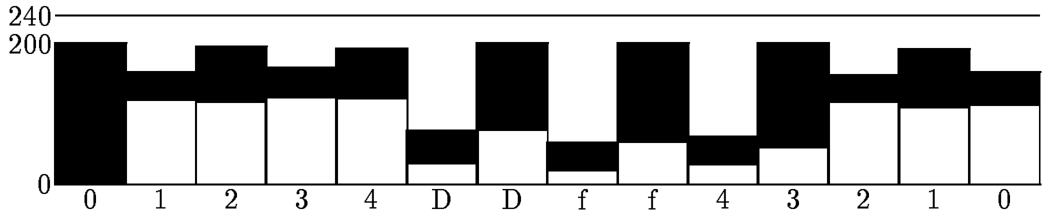

Figure 4: Difference 10, Maximum 200

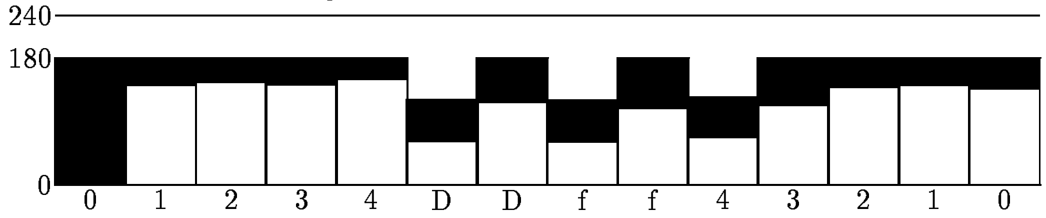

Figure 5: Difference 20, Minimum 60, Maximum 180

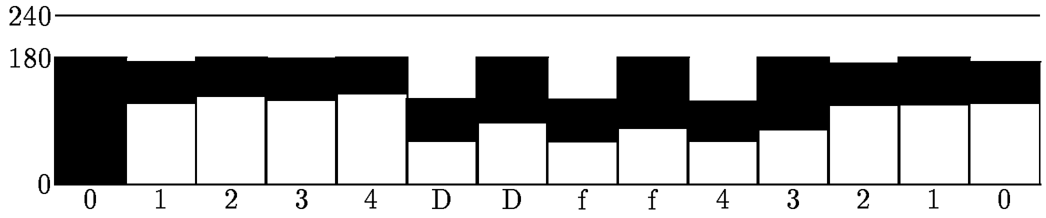

Figure 6: Difference 10, Minimum 60, Maximum 180

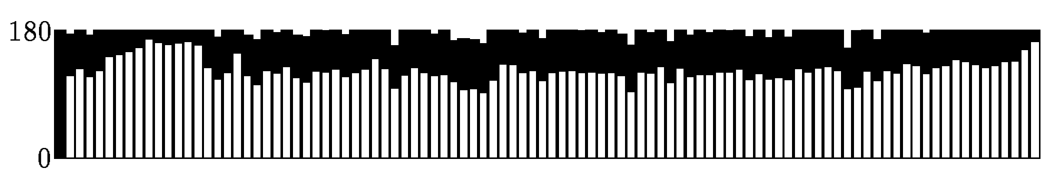

Figure 7: Difference 10, Minimum 60, Maximum 180

4.2 Experience with the Transmission of Movies

For first experience MTP was implemented over Ethernet and UDP since no high-speed network adapters were available for our workstations, and TCP turned out to be too slow.

For our performance experiments we used the two digital movies mentioned earlier with image sizes of 156x115 and 220x220 pixels. On an X Windows screen the movies could be presented at 7-16 frames/s. Measurements have shown that currently the Ethernet adapter is the bottleneck of our system [LE91]. We expect that larger image sizes at a rate of 25 frames/s will be possible without software modifications once a faster network is available.

5 Conclusions and Outlook

We have presented an algorithm for the continuous adaptation of a color lookup table for digital movies. It allows the presentation of films with good color quality using standard color lookup table graphics adapters while preventing the showing of false colors between the frames.

The XMovie system is a test bed for digital image transmission and presentation in a network of UNIX workstations. It is entirely based on standard hardware and uses standard network technology and standard graphics adapters with color lookup tables.

As expected, experience has shown that standard Ethernet technology and TCP/IP protocols, are not suitable for movie transmission. The transmission of digital movies will require new protocols which are at the same time isochronous, i.e. have a very low delay jitter, and provide high throughput.

For the presentation of movies on the workstation screen, extentions to the X Window System were presented. Experience with XMovie has shown that the presentation of digital movies in a network of standard UNIX workstations is feasible without special-purpose hardware.

Acknowledgements

We wish to thank Prof. A. Schmitt, W. Leister, and A. Stößer from the Institut of Operating and Dialog Systems of the University of Karlsruhe from where we got the butterfly scene (generated with the VERA Raytracing System). We also thank E. I. du Pont de Nemours & Company for the DNA movie and fish picture, which are part of the "ImageMagick" packet.

References

[ADH91] D.P. Anderson, L. Delgrossi, and R.G. Herrtwich. Process Structure and Scheduling in Real-Time Protocol Implementations. In *Kommunikation in Verteilten Systemen,* Mannheim. Informatik-Fachberichte, Springer-Verlag, 1991.

[ATW+90] D.P. Anderson, S. Tsou, R. Wahbe, R. Govindan, and M. Andrews. Support for Continous Media in the Dash System. In *Proc. Internat. Conf. on Distributed Computing Systems,* Paris, 1990.

[BCDW90] G. S. Blair, G. Coulson, P. Dark, and N. Williams. Engineering Support for Multimedia Applications in Open Distributed Processing (Extended Abstract). In *Proc. 3rd IEEE COMSOC International Multimedia Workshop (Multimedia '90)*, November 1990.

[BHS90] G. S. Blair, D. Hutchison, and W. D. Shepherd. Distributed System support for Heterogeneous, Multimedia Environments. In *International Workshop on Operating System Support for Digital Audio and Video, Berkeley, USA*, November 1990.

[BHSS90] F. Ball, D. Hutchison, A. Scott, and W.D. Shepherd. A Multimedia Network Interface. In *International Workshop on Operating System Support for Digital Audio and Video, Berkeley, USA*, November 1990.

[Bru89] T. Brunhoff. *VEX: Video Extension to X, Version 5.5.* Textronix, Inc., 1989.

[CGHS91] G. Coulson, F. Garcia, D. Hutchison, and D. Shepherd. Protocol support for distributed multimedia applications. In *Second International Workshop on Network and Operating System Support for Digital Audio and Video, Heidelberg.* Springer-Verlag, Heidelberg, November 1991.

[Fer91] D. Ferrari. Design and Applications of a Delay Jitter Control Scheme for Packet-Switching Internetworks. In *Second International Workshop on Network and Operating System Support for Digital Audio and Video, Heidelberg.* Springer-Verlag, Heidelberg, November 1991.

[GG91] M. Gilge and R. Gusella. Motion Video Coding for Packet-Switching Networks – An Integral Approach. In *Proc. SPIE Conference on Visual Communications and Image Processing,* Boston, November 1991.

[HH91] R. Händler and M. N. Huber. *Intergrated Broadband Networks.* Addison Wesley, 1991.

[HHS+91] D. Hehmann, R.G. Herrtwich, W. Schulz, Th. Schütt, and R. Steinmetz. Implementing HeiTS: Architecture and Implementation Strategy of the Heidelberg High-Speed Transport System. In *Second International Workshop on Network and Operating System Support for Digital Audio and Video, Heidelberg.* Springer-Verlag, Heidelberg, November 1991.

[HN88] A. Hopper and R.M. Needham. The Cambridge Fast Networking System. *IEEE Trans. on Comp.*, 37(10), October 1988.

[Hop90] A. Hopper. Pandora – an Experimental System for Multimedia Applications. *Operating Systems*, 24(2), April 1990.

[Hop91] A. Hopper. Design and Use of High-Speed Networks in Multimedia Applications. In A. Danthine and O. Spaniol, editors, *Proc. Third IFIP WG 6.4 Conference on High Speed Networking.* North Holland, 1991.

[HP89] H. Hild and M. Pins. Variations on a Dither Algorithm. In *EUROGRAPHICS'89*, pages 381–392. North-Holland, Amsterdam, 1989.

[Kel91] R. Keller. Erweiterung des X11-Servers zur Bewegtbilddarstellung. Master's thesis, Lehrstuhl für Praktische Informatik IV, Universität Mannheim (in German), 1991.

[LE91] B. Lamparter and W. Effelsberg. X-MOVIE: Transmission and Presentation of Digital Movies under X. In *Second International Workshop on Network and Operating System Support for Digital Audio and Video, Heidelberg.* Springer-Verlag, Heidelberg, November 1991.

[Lof90] G. Loff. Konzeption, Entwurf und Implementierung eines netzweiten Videodienstes. Master's thesis, Universität Karlsruhe (in German), 1990.

[Lut89] A. Luther. *Digital Video in the PC Environment.* McGraw Hill Book Company, New York, 1989.

[Mic91] N. Michl. Dynamische Änderung der Farbtabelle bei Bewegtbildanwendungen. Master's thesis, Lehrstuhl für Praktische Informatik IV, Universität Mannheim (in German), 1991.

[SHRS90] R. Steinmetz, R. Heite, J. Rückert, and B. Schöner. Compound Multimedia Objects – Integration into Network and Operating Systems. In *International Workshop on Network and Operating System Support for Digital Audio and Video,* ICSI, Berkeley, November 1990.