[HKSt92] *Dietmar Hehmann, Thomas Käppner, Ralf Steinmetz;* **An Introduction to HeiMAT: The Heidelberg Multimedia Application Toolkit;** 3rd International Workshop on Network and Operating System Support for Digital Audio and Video, San Diego, CA, USA, November 1992, und in Lectures Notes in Computer Science, Springer-Verlag, 1993.

An Introduction to HeiMAT: The Heidelberg Multimedia Application Toolkit

Thomas Käppner
Dietmar Hehmann
Ralf Steinmetz
IBM European Networking Center
P.O.Box 10 30 68
D-6900 Heidelberg 1
Germany
{kaeppner, hehmann, steinmet} @ dhdibml.bitnet
Phone: +49-6221-404-403 / -214 / -280
Fax: +49-6221-404-450

Abstract: HeiMAT is a toolkit that supports the development of distributed multimedia applications. It provides adaptable communication services, distribution transparency, and cross platform interconnectivity (between various operating and window systems). HeiMAT makes extensive use of product and prototype level available multimedia communications and operating system environments. This paper motivates the major design decisions and presents the system structure of HeiMAT.

1 Introduction

1.1 Motivation

At the IBM European Networking Center (ENC) in Heidelberg, Germany, the HeiProjects have been established to develop prototypes that support distributed multimedia applications on RISC System/6000s under AIX as well as on PS/2s under OS/2 [Herrtwich, 1992] [Luttenberger and Steinmetz, 1992]. Within this framework three related areas have been worked on:

- HeiCoRe: The Heidelberg Continuous-Media Realm is concerned with providing local support services for multimedia on the workstations mentioned above. Those services consist of buffer management, an operating system shield, resource management, and a real-time environment for stream handlers [Herrtwich and Wolf, 1992] [Herrtwich, 1992a] that handle multimedia data. In the initial phase this system support was designed and implemented for AIX as well as OS/2. However, with the introduction of appropriate multimedia products, some of HeiCoRe's earlier prototype code will be replaced. This has already happened with IBM's Multimedia Presentation Manager/2 (MMPM/2) [IBM, 1992], i.e., on OS/2, HeiCoRe stands for MMPM/2 with an additional resource management and an operating system shield.
- HeiTS: The Heidelberg Transport System [Hehmann et al., 1991] [Herrtwich and Delgrossi, 1992] is designed to transfer continuous media data between systems over today's networks such as Token Ring or FDDI in real time. The kind of media and its properties are specified by the transport service user employing quality of service (QoS) parameters, which are then negotiated between the different HeiTS stacks. HeiTS runs as a stream handler in the HeiCoRe environment.

- HeiMAT: The Heidelberg Multimedia Application Toolkit interfaces HeiCoRe providing a uniform distribution mechanism on both platforms. It allows for abstractions of multimedia data, giving access to functions that are commonly needed in multimedia applications like synchronization and mixing of streams. On top of these generic abstractions, modules are provided that are ready to use for development of applications that belong to specific classes. It is aimed to supply AIX as well as OS/2 applications with a homogeneous interface. This paper focuses on this third project area.

Today's multimedia application products are usually programmed in conventional languages (such as C), augmented with hardware-specific multimedia libraries. Replacing any underlying continuous-media device, even with a functionally-equivalent component from another vendor, often requires re-implementing a substantial part of the application programs. If, on the other hand, applications were built using high level abstractions, they would not be affected by these replacements. These applications could even be ported to other platforms where the abstractions are provided without any effort.

1.2 Related Work

We believe that various levels of abstractions for multimedia are required starting with the low-level operating system extensions like [Herrtwich, 1990] and [Leung, 1988]. Newly available multimedia products, such as IBM's MMPM/2, Microsoft's Multimedia Extensions [Microsoft, 1991] and Apple's QuickTime [Apple, 1991] provide the next layer of abstraction for programming of applications. They are however, unlike HeiMAT, not aimed at the distributed environment, and HeiMAT goes beyond their interfaces in providing top-layer abstractions such as a video conferencing module that serves as a complete building block for multimedia applications.

Several experimental systems provide subsets of HeiMAT's functionality. Among them are ACME [Anderson et al., 1990], VOX [Arons et al., 1989][Arons et al., 1989a], [Angebranndt et al., 1991], and [Sventek, 1987]. Those systems cover the distribution aspect of HeiMAT, in that they provide access to an I/O-server within a distributed environment. HeiMAT does not only provide network transparency, but also serves as a toolkit to allow for application development that is independent of multimedia encoding and devices. Those existing toolkits that yield similar support [Anderson and Chan, 1991] are solely aimed at a homogeneous environment in terms of windowing and operating system which usually is UNIX. HeiMAT covers two different environments.

In the subsequent section we introduce our major design principles. These design decisions lead to the system structure of HeiMAT as discussed in Section 3. Section 4 describes the generic functional blocks and Section 5 introduces the concept of support for special classes of applications in HeiMAT.

2 Design Principles

In [Steinmetz and Fritzsche, 1992], the authors note that multimedia programming today is typically based on low-level constructs with strong hardware dependencies. The available multimedia operating system extensions relieve this problem to a certain degree for local applications. Application programmers who want to build, e.g., a multimedia tutoring system have to bridge a wide gap between the functions available to them and the service they want to provide. The complexity they have to deal with increases when the multimedia application is distributed and network access has to be programmed as well. We experienced this with our first OS/2 based integrated multimedia communication system [Cramer et al., 1992] [Steinmetz and Meyer, 1992]. Relying on this experience and the HeiProjects framework we designed the following set of principles.

2.1 Flexibility of Services

At the highest level it should be very easy to develop applications. The toolkit requires only the essential knowledge from the application in order to supply the demanded services. All the details involved with establishing data streams using devices in the distributed system are hidden from the application.

Multimedia applications that are to be supported by HeiMAT form a very diverse set. Even applications, which belong to one class at the first sight, can range from very simple to full featured professional systems. Imagine e.g. a simple multimedia editor for mailing purposes compared to a professional system used for editing commercials. Their requirements with regard to timeliness, quality of edited data streams, and exactness of presentation differ substantially. For some developers fast development is important, whereas others demand the usage of special compressed data types. Thus, services offered by HeiMAT should not only be easy to use, but must also be flexible enough to be useful for every application. This flexibility imposes a set of subsequently discussed transparency requirements which also belong to the design principles.

2.2 Device Transparency

The independence from multimedia devices requires to hide the characteristics of physical devices, thus, allowing for the development of portable applications and making the development much more worthwhile. It also includes the provision of a source/sink paradigm as part of the HeiMAT upper layer interface.

Generally, using audio/video data in applications causes the establishment of streams on the lower level of the system [Herrtwich and Wolf, 1992]. Since streaming of data is very illustrative, it serves also well as a programming abstraction [Steinmetz and Meyer, 1992]. Using this abstraction, an application specifies the endpoints of streams they want to establish. However, details of the stream, which the application is not interested in, can be hidden by HeiMAT.

Information about endpoints can be specified using the level of detail demanded by the application. Imagine e.g. a voice mail system establishing a data stream from the central server to the workstation. It can issue a request to HeiMAT naming the involved devices and a quality for the data stream of *high quality voice*. HeiMAT will map this specification to lower level data formats which not only correspond to the expressed general quality, but are also appropriate for the devices. In contrast, a more powerful retrieval system can utilize special audio formats and compression schemes by explicitly requesting them.

Independently of whether the information about endpoints and filters is specified at a high or low level, we can distinguish three different origins:

- The application is not interested in that specification: HeiMAT will select an appropriate value.
- The application wants to specify it by itself.
- The application lets the user make the decision.

Programmers of applications can decide which one of the alternatives is used. If parameters are simply omitted, HeiMAT will assign a value. Using the corresponding interface functions of HeiMAT, the application can specify any value. In order to let the user decide about the information, HeiMAT provides elements that an application can construct a user interface of. These elements, applied by an application, will let the user decide on a specification transparently for the application.

2.3 Presentation Transparency

Today's coding technology and standards are still evolving, compete again de-facto standards, and provide diverse algorithms for the same coding problems. The ISO JPEG standard defines compression and coding of single images like many available de-facto standards. ISO MPEG video specifies compression and competes with, e.g., CCITT H.261 and the DVI de-facto compression standard. MPEG-2 is still to be defined for video compression with higher quality. MPEG audio covers audio compression similar to CCITT (e.g. G.721 and 722). However, it also allows for a compressed data rate of more than 64 kbit per second.

Applications need to be developed without constraining to certain coding standards. They should be allowed to specify quality parameters in a manner, which is independent of the real encoding of data. Requesting a new data stream, the application can connect devices with different data types. If a conversion needs to be applied, HeiMAT will insert a filter in the data stream converting from source to sink type. This filter is fully transparent to the application. Thus, HeiMAT allows applications to use lower levels of specification and even explicit usage of different encoding schemes along a data stream will be handled gracefully by the toolkit.

Explicit specification of filters (e.g. mixer, multiplexer) can either be made by type or by name of the device to be used. If only the type of the device is specified, the application is not interested in location and implementation of that specific unit. On the other hand using device names, an application can choose from several different implementations of the same type of a device (e.g. choose distributed or centralized mixing for their respective advantages [Vin et al., 1991][Anderson and Chan, 1991]).

2.4 Distribution Transparency

Applications do not need not know about the actual location of devices. For the application there should not exist any difference between usage of local and remote devices. Since development of applications is more complex being aware of their distributed nature, HeiMAT must provide mechanisms to hide the distribution from the application where desired. Hiding the distribution means

- Dispatching requests of an application to the corresponding remote servers
- Managing necessary transport connections for multimedia data.

Imagine a distributed application, which establishes a data stream between two or more hosts. Such an application can be realized by one application entity residing on any host. The application would have to contact the respective servers, which are responsible for devices on the involved hosts. However, HeiMAT supports device names consisting of a host and a device name local to that host. Any request with regard to a specific device is routed to the responsible server by HeiMAT, thus leaving the application unaware of the need to communicate with possibly several servers.

During establishment of data streams HeiMAT will set up a transport connection, wherever a data stream crosses host boundaries, thus completing distribution transparency.

Hence, applications can be developed totally unaware of their distributed nature. Changing device names is sufficient to switch between local and remote operation. The application can run on any host accessible in the network without any changes. However, applications are free to use their knowledge about device distribution if they decide to do so. For example, an application could insert special compression and decompression filters in the data stream before and after the network devices, respectively.

Note, HeiMAT does not preclude any architecture for an application. Distributed applications based on the client/server approach can also be build easily on top of HeiMAT. The advantage of this approach is that local and remote execution of functions can be triggered independently. For example, a *stop stream* operation could be performed locally, immediately freezing a video image on the screen, whereas in the distribution-transparent case, the request would always be sent to the application, wherever it resides. This will cause the video not to stop before one round-trip time.

3 The System Structure

This set of design principles of HeiMAT together with the availability of HeiMAT on two different platforms imposes hard design requirements on it's system structure. HeiMAT must be a seamless integrator of the available paradigms on both platforms.

The OS/2 platform provides the MMPM/2 as local multimedia extension to the operating system. MMPM/2 allows the definition of sources and sinks of streams to be used by the application. HeiCoRe adds resource management and an operating system shield. The resource management provides the reservation and scheduling of reserved resources in a distributed system [Vogt et al., 1992]. The implementation of the full HeiCoRe environment as a port from AIX with adaptation to the MMPM/2 is in progress. MMPM/2 closely interacts with the Presentation Manager, the OS/2 native window system. This window system was designed for fast response with a large set of functions in a local environment.

For the input and output of discrete media in a UNIX environment, the X window system is the most prominent and widely used system. It provides abstractions from both the hardware dependencies of text and graphics I/O and network transport. It introduces various layers of abstraction from the basic Xlib over the X toolkit to a widget set such as OSF/Motif. Most application programmers base their code on the highest abstractions; however, they can access the lower layers if they need to do so. We envisage multimedia support in a distributed environment to be architected similar to X. In addition to the traditional X server, handling discrete media, there is another server in that architecture, which we call the AV server. It deals with continuous media and communicates with X for presentation on the common display for video output.

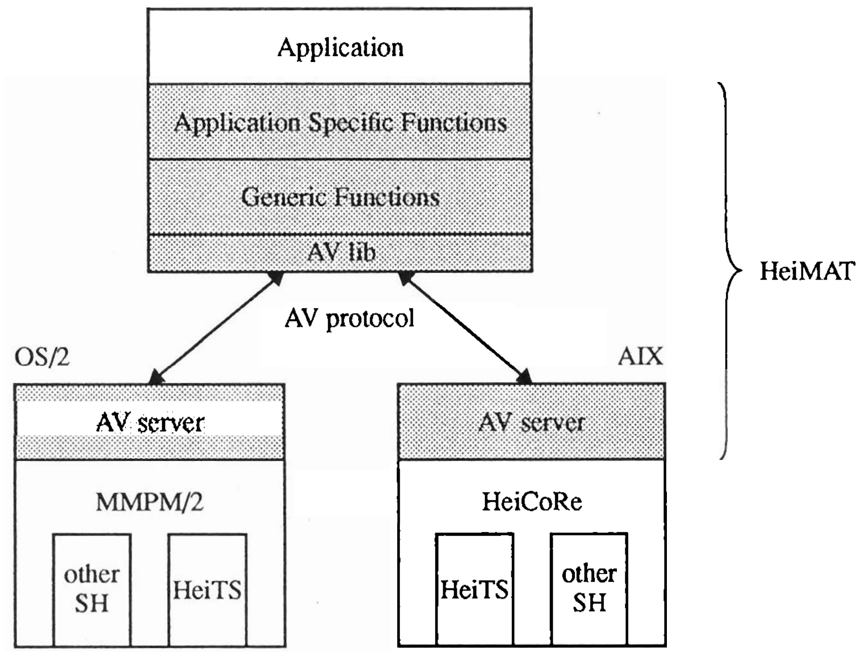

Fig. 1: HeiMAT and it's enviroment on different platforms.

The AV server encapsulates the functionality of HeiCoRe providing access to all types of stream-oriented multimedia devices/filters through a consistent interface (see Fig. 1). As in X, communication between server and application program (client) is supported by an AV protocol which itself is hidden by an AVlib. A typical function set provided by the AVlib includes operations for creation, modification, connection, control, and destruction of the logical multimedia devices, which run in a dedicated real-time environment.

The base layer of HeiMAT comprises the top layer of the AV server that handles the AV protocol, thus, HeiMAT being the first level that allows for remote access to HeiCoRe functionality. On top of this base layer HeiMAT provides services that belong to two distinguished layers:

- Generic functions are common to most multimedia applications (see Section 4). They are provided in a network and device transparent fashion and form a general basis for the development of applications.
- Application specific modules are built on top of generic functions (see Section 5). They provide support for specific classes of applications. The classes include conferencing, collaboration, editing, mailing, and retrieval of multimedia data.

However, HeiMAT will also support the development of multimedia interfaces through user interfacle elements, which partly are already provided in MMPM/2. For this purpose in AIX, HeiMAT itself uses the X toolkit stack. Thus, HeiMAT is residing on both the AV server and the X toolkit stack to provide it's services.

In OS/2 the base layer of HeiMAT is being built around the native Media Device Interface which is a component of the MMPM/2 capabilities in HeiCoRe. It distributes calls to the local multimedia devices. In the first release of the distributed OS/2 system, HeiCam (Heidelberg Remote Camera Control) uses this transparent distributed access to remote multimedia devices.

4 Generic Functions of HeiMAT

Generic functions are used in all types of applications. They are provided according to the design principles and form the basis for application development.

4.1 Abstract Devices

Audio and video devices differ in their capabilities and interfaces. To allow applications to work device independently, logical devices are introduced, which hide the detailed characteristics of physical devices. Classes of devices are defined, in which they are grouped by type (e.g. audio input, mixer, multiplexer, weaver). Every member of a class provides a similar interface to the application. However, devices can implement subsets of the class' functionality and can generally differ in their implementation. It turned out to be a rather difficult task to define a common interface across several similar audio or video devices [Steinmetz and Fritzsche, 1992].

In order to give the user control over a device, HeiMAT provides user interface elements, readily applicable by the application. Thus, the user can control e.g.

- start and stop of a device by a button,
- volume and balance of a stereo device by circular sliders and mixing ratios of several channels by slider bars,

all in a device control window, communicating with HeiMAT transparently for the application.

4.2 Data-Type-Related Functions

During establishment of a data stream HeiMAT has to arbitrate the capabilities of the devices/filters involved. If data types that the devices can work with do not match, HeiMAT tries to apply a conversion. Data type conversion comes in different flavors:

- Degradation of quality: This has to be applied, if in a data stream the quality of the data at the source is higher than the quality which can be displayed at the sink (e.g. the source supplies an stereo audio signal and the sink can play mono only). Generally, downgrading the quality should be done at the source, since degraded streams use less resources in the workstations as well as in the network [Anderson and Chan, 1991]. Note however, the JPEG hierarchical coding mode comprises the joint compression with several resolutions. Since the compression makes use of the redundancy at the different resolutions, the choice of the picture to be displayed can be done at the sink only.
- Multiplexer/Demultiplexer: A data stream contains either one medium only or con-

sists of several interleaved media (e.g. PCM audio and RTV video in an DVI AVSS stream). The conversion between these kinds of formats is referred to as weaving and unraveling.

- Mixer: When several audio streams arrive at a single sink, they generally must be mixed digitally, since only one device is used to apply the conversion to an analog signal.
- Compression/Decompression: In today's multimedia systems, multimedia data streams are normally compressed, due to storage and data rate constraints. Compression and decompression have to be negotiated and applied by HeiMAT. This can include conversion between different compressed data formats.

4.3 QoS Management

During establishment of a data stream, HeiMAT derives the QoS parameters that are valid for the lower layer services. These are passed to HeiCoRe, where they are used to evaluate whether and how the necessary resources can be provided by the real-time environment.

QoS parameters do not only depend on the kind of media used in a data stream or on the classes of service the application provides (as defined in CCITT Study Group XVIII, Draft Recommendation I.211, which include conversational, distribution, retrieval).

The great diversity of CCITT services and applications within individual CCITT service classes yield a great variability of communication requirements. Hence, it remains difficult to classify applications such as distributed tutoring or joint editing based on the CCITT scheme alone.

Other groups of performance criteria have been proposed by [Wright and To, 1990]: Delay sensitive, loss sensitive, and delay and loss sensitive. However, this classification is very coarse.

We suggest that a service must be treated within its context; coding and compression must be taken into account. The QoS depends on the kind of media as well as the service class, the type of sink and source (live -nonpersistent- or stored -persistent-), as well as the configuration of the application [Steinmetz and Meyer, 1992].

4.4 Synchronization

Most applications need synchronization for the correct display of several separated data streams. HeiMAT provides abstractions to group data streams in order to (1) express time relations between them and (2) allow for concurrent execution of operations on all streams in a group.

Synchronization can be expressed (1) implicitly, by combining audio and video (which easily solves many synchronization issues), or (2) explicitly using separate connections for different streams.

Utilizing HeiTS for transport connections, synchronization can be guaranteed by imposing the same end-to-end delay on related streams (choosing an absolute end-to-end delay and limiting the jitter of the data information units to about 0 msec). In practice, it is neither possible nor necessary to guarantee service with such tight tolerances. Audio can be played ahead of video for about 120 msec, and video can be displayed ahead of audio for about 240 msec [Murphy, 1990]. Both temporal skews will sometimes be noticed, but can easily be tolerated without any inconvenience by the user. Note, this asymmetry is very plausible: In a conversation in which two people are located 20 m apart, the visual impression will always be about 60 msec ahead of the acoustics due to the fast light propagation compared to the acoustic wave propagation.

In the next version a more sophisticated implementation of synchronization allows to guarantee timing relations between various sources: A logical time system (LTS) is being introduced as suggested by [Anderson et al., 1990]. Presentation of the data is performed based on a comparison of the LTS with the real-time clock of the destination workstation.

5 Application Specific Functions

The upper layer modules are described in the following by looking at modules designed to provide parts of the functionality needed for conferencing.

An application-specific module can be understood as a complete application itself. However, the module can as well be used as one building block in other applications. Imagine e.g. the development of a banking application in which the customer at an automatic teller machine communicates with specialists located at the remote support center. By utilizing this conferencing module, the development of the banking application can be drastically reduced towards adaptation and enhancements of the available conferencing module.

On the other hand, a module usually comprises several independent application specific functions. Thus, an application can reuse some of the functions provided and implement others on top of it. That yields full flexibility and support for the application.

For conferencing we envisage the following functions:

Conference Environment

Conference directories are utilized to store information about potential participants in conferences. Group information in the directory indicates a special relationship between participants. It contains the information who may assume which role in a conference instantiated on that group. Groups allow users to conveniently invoke conferences with people they need to meet more often.

Conference Scheduling

Scheduling of conferences allows to make appointments for and to plan conferences. These mechanisms are not only needed to announce future conferences to many participants, but can also be used to reserve scarce resources in advance.

Conference Management

There are two different approaches to conference management: Centralized and distributed architectures. Independently of the approach many aspects are involved:

- Invitation protocols must be provided, which can handle concurrent requests by several participants.
- Negotiation is needed about media types to be used, roles participants can assume in a conference, whether mixing of media streams is applied, which floor mecha-

nisms are used etc.

- Streams are established, using the lower layer functions of HeiMAT.
- Floor passing comprises mechanisms and strategies to determine which participant in a conference can provide input at a given time.

All of above functions must be accessible via natural graphical interfaces, e.g. a virtual conference room (represented by a window), which can be left by participants simply by dragging out their picture.

6 Concluding Remarks

HeiMAT, the Heidelberg Application Toolkit, will provide a set of functions that greatly simplifies development of multimedia applications. In analogy to the X Window toolkit stack it will allow application programmers to use a high level of abstraction by which details of distribution and multimedia devices are handled transparently. However, applications can be build refining this high level specification and thus utilizing advanced features of the underlying system. The system structure allows for the integration of two different system environments, AIX and OS/2. Note, the described work is still in progress.

We would like to thank Ralf Guido Herrtwich for his many useful comments on an earlier version of this paper.

7 References

[Arons et al., 1989]
Arons, Barry; Binding, Carl; Lantz, Keith; Schmandt, Chris: "A Voice and Audio Server for Multimedia Workstations", Proceedings of Speech Tech '89, May 1989.

[Arons et al., 1989a]
Arons, Barry; Binding, Carl; Lantz, Keith; Schmandt, Chris: "The VOX Audio Server", 2nd IEEE COMSOC International Multimedia Communications Workshop, Montebello, Quebec, Canada, Apr. 1989

[Anderson et al., 1990]
Anderson, David; Govindan, Govindan; Homsy, George: "Abstractions for Continuous Media in a Network Window System", Technical Report UCB/CSD 90/596, UC Berkeley, Sep. 1990.

[Anderson and Chan, 1991]
Anderson, David P.; Chan, Pamela: "Toolkit Support for Multiuser Audio/Video Applications", in: Herrtwich, R.G. (Ed.): Proc. Second International Workshop on "Network and Operating Systems Support for Digital Audio and Video", Heidelberg, Germany, Nov. 1991, 230-241.

[Angebranndt et al., 1991]
Angebranndt, Susan; Hyde, Richard L.; Luong, Daphne Huetu; Siravara, Nagendra; Schmandt, Chris: Integrating Audio and Telephony in a Distributed Workstation Environment", Proceedings 1991 Summer Usenix Conference, 59-

74.

[Apple, 1991]
Apple: "QuickTime Developer's Kit Version 1.0", Apple Document Number 030-1899.

[Cramer et al., 1992]
Cramer, Andreas; Farber, Manny; Hehmann, Dietmar; Jungius, Christiane; Luttenberger, Norbert; Markgraf, Frank; McKellar, Brian; Mengler, Stefan; Meyer, Thomas; Reinhardt, Kurt; Sander, Peter; Sandvoss, Jochen; Schütt, Thomas; Schulz, Werner; Steinbeck, Werner; Steinmetz, Ralf; Stüttgen, Heiner; Vogt, Carsten: "The Heidelberg Multimedia Communication System: Multicast, Rate Enforcement and Performance on Single User Workstations", IBM ENC Technical Report no.43.9209, July 1992.

[Herrtwich, 1990]
Herrtwich, Ralf Guido: "Time Capsules: An Abstraction for Access to Continuous-Media Data", IEEE Real-Time Systems Symposium, Orlando, December 5-7, 1990, pp.11-20.

[Hehmann et al., 1991]
Hehmann, Dietmar, Herrtwich, Ralf Guido, Schulz, Werner, Schütt, Thomas, Steinmetz, Ralf: "Implementing HeiTS: Architecture and Implementation Strategy of the Heidelberg High-Speed Transport System", in: Herrtwich, R.G. (Ed.): Proc. Second International Workshop on "Network and Operating Systems Support for Digital Audio and Video", Heidelberg, Germany, Nov. 1991, 33-44.

[Herrtwich, 1992]
Herrtwich, Ralf Guido: "The HeiProjects: Support for Distributed Multimedia Applications", IBM ENC Technical Report No. 43.9206, 1992.

[Herrtwich, 1992a]
Herrtwich, Ralf Guido: "An Architecture for Multimedia Data Stream Handling and Its Implication for Multimedia Transport Service Interfaces", 3rd IEEE Workshop on Future Trends of Distributed Computing Systems, Taipei, April, 1992.

[Herrtwich and Delgrossi, 1992]
Herrtwich, Ralf Guido, Delgrossi, Luca: "Beyond ST-II: Fulfilling the Requirements of Multimedia Communication", 3rd Intl. Workshop on Network and Operating System Support for Digital Audio and Video, San Diego, Nov. 1993.

[Herrtwich and Wolf 1992]
Herrtwich, Ralf Guido; Wolf, Lars: "A System Software Structure for Distributed Multimedia Systems", 5th ACM SIGOPS European Workshop, Le Mont Saint-Michel, France, September 1992.

[IBM 1992]
IBM: "Multimedia Presentation Manager/2: Programming Reference" IBM

Document Number 41G2920.

[Leung et al., 1988]
Leung, W. H.; Luderer, G. W. R.; Morgan, M. J.; Roberts, P. R.; Tu, S.-C.: "A Set of Operating System Mechanisms to Support Multi-Media Applications", Proc. Intern. Seminar on Digital Comm., Zurich, Mar. 1988, 71-76.

[Luttenberger and Steinmetz, 1992]
Luttenberger, Norbert; Steinmetz, Ralf: "Videocommunication over the IBM Token Ring", leaflet for CeBIT '92 and Didacticum no. 14, July 1992, 20-23.

[Microsoft 1991]
Microsoft Corporation: "Microsoft Windows: Multimedia Programmer's Reference", Microsoft Press, 1991.

[Murphy, 1990]
Murphy, Alan: "Lip Synchronization" Personal Communication on a Set of Experiments.

[Steinmetz and Fritzsche, 1992]
Steinmetz, Ralf; Fritzsche, J., Christian: "Abstractions for Continuous-Media Programming", Computer Communications, vol. 15, no. 4, July/August 1992.

[Steinmetz and Meyer, 1992]
Steinmetz, Ralf; Meyer, Thomas: "Modelling Distributed Multimedia Applications", Intl. Workshop on Adv. Comm. and Appl. for High Speed Networks, Munich, Germany, March, 1992.

[Sventek 1987]
Sventek, J. S.: "An Architecture for Supporting Multimedia Integration", IEEE Computer Society Office Automation Symposium, Apr. 1987, pp.46-56.

[Vin et al., 1991]
Vin, Harrick M; Rangan, P. Venkat; Ramanathan, Srinivas: "Hierarchical Conferencing Architectures for Inter-Group Multimedia Collaboration", Proceedings of Conference on Organizational Computing Systems (COCS'91), November 1991.

[Vogt et al., 1992]
Vogt, Carsten; Herrtwich, Ralf Guido; Nagarajan, Ramesh: "HeiRAT: The Heidelberg Resource Administration Technique, Design Philosophy and Goals", IBM ENC Technical Report no.43.9101, March 1991.

[Wright and To 1990]
Wright, David J.; To, Michael: "Telecommunication Applications of the 1990s and their Transport Requirements", IEEE Network Magazine, vol.4, no.2, March 1990, pp.34-40.