[StMe92b] *Ralf Steinmetz, Thomas Meyer,* **Multimedia Synchronization Techniques: Experiences Based on Different System Structures**; acm Computer Communications Review, Band 22, Nr.3, Juli 1992, S.90-91; Zusammenfassung von [Stei92].

Multimedia Synchronization Techniques: Experiences Based on Different System Structures

Ralf Steinmetz

IBM European Networking Center,
Tiergartenstr. 8, 6900 Heidelberg, P.O.BOX 10 30 68,
Tel: +49 6221 404280, FAX: +49 6221 404450
e-mail: STEINMET at DHDIBM1.BITNET
Germany

Abstract

"Multimedia synchronization" comprises the definition as well as the establishment of temporal relationships among audio, video and other data. This paper discusses techniques for implementing synchronization in distributed systems. As it turns out, the applied methods heavily depend on the underlying system's structure: "Hybrid" systems commonly make use of different approaches than "unified" system structures do.
The work described here is being carried out as part of HeiTS project at IBM ENC. It draws also on (unpublished) experiences with synchronization gained from the DiME project at our multimedia communications lab. We discuss some issues pertinent to synthetic synchronization, but our main focus is on live synchronization: Playing back data as it occurs/occurred somewhere (either locally or remotely).

1. Introduction: Environment and Motivation

Multimedia synchronization is needed to ensure a temporal ordering of events in a multimedia system. There are two parts to this problem: the **definition** of temporal relationships among audio, video, and other data; and the **delivery** of the data in accordance with these relationships.
Synchronization mechanisms are a well-studied topic in the area of operating systems, parallel programming languages and database technology. In distributed multimedia systems we encompass synchronization to be defined and happen between continuous media (CM) streams and discrete media (DM). In the DiME and HeiTS projects (described below), much work was devoted to defining the requirements and to analyzing "new" problems [16; 17; 23; 25; 32]. Some work in related projects was done on how best to implement multimedia synchronization [2; 5; 29]. With this paper we want to go further: Based on the experience in DiME and HeiTS, we document the implications of the system structures on the required synchronization implementation techniques.

The DiME (Distributed Multimedia Environment) project, carried out at the IBM European Networking Center in Heidelberg, was based on a *hybrid* system structure. Continuous media (i.e., audio and video) were routed over separate, dedicated channels using off-the-shelf hardware technology. Continuous and discrete media were integrated by connecting the CM equipment (e.g., CD players, VCRs) to a computer via an RS-232C interface. The devices can then be controlled by software instead of manually from the devices' control panels. For instance the audio/video data were processed in the workstation using IBM's AVC and M-Motion adapters and respective system software [20; 19]. Further experiments included the ActionMedia 750 (DVI) technology [9]. DiME dealt with distributed, transparent access to multimedia resources like cameras and stored video sequences [27; 28; 31]. DiME aimed to provide an "easy, but rich" communication service as part of an application programming interface, manipulating data streams by controlling their sources and sinks in a heterogeneous computing environment. Considerable work was devoted to dealing with synchronization in such an environment [32].

Our follow-on project, the Heidelberg High-Speed Transport System (HeiTS), is based on a *unified* digital system structure: All discrete as well as continuous media are routed through the workstation [10]. Scheduling can either be done exclusively in software or by dedicated hardware, such as the ActionMedia II (DVI), which also provides support for compression/decompression. But both solutions require real-time scheduling techniques in a time-sharing environment [11] similar to [1]. Our prototype is being developed for AIX on the RS/6000 and for OS/2 on the PS/2 simultaneously. The goals of HeiTS are to demonstrate the feasibility of current network technology for multimedia applications, to explore the limitations of current protocols, to make appropriate changes and/or enhancements, to prove new implementation concepts (e.g., upcalls), and to exploit and integrate upcoming broadband WANs and LANs [10]. The integration of various kinds of media and their processing with the close relationships demanded requires "synchronization". Synchronization in such an integrated environment as HeiTS uses different techniques than were used in DiME.

The remainder of this text is organized into three sections. Section 2 explains the concepts of "synthetic" and "live" synchronization and introduces the logical data unit (LDU) to express the granularity of a medium. The final two sections concentrate on live synchronization, discussing only some aspects of synthetic synchronization (as defined in [17]). Section 3 discusses synchronization in the context of hybrid approaches and Section 4 outlines techniques for implementing synchronization in unified systems structures such as HeiTS.

2. Synthetic and Live Synchronization

We understand "multimedia" according to [13; 30; 34]: "A multimedia system is characterized by the integrated computer-controlled generation, manipulation, presentation, storage, and communication of independent discrete (DM) and continuous media (CM)".

Digitalization and **Synchronization** are the key to the goal of "integration". It is important to distinguish between **live synchronization** and **synthetic synchronization**.[1] Often authors confuse the two modes of synchronization, and it is not clear of which type of synchronization they mean. These two types of synchronization require separation when an application uses both. However, it is only recently that this basic differentiation began to appear. The following application scenarios illustrate the difference:

[1] The following exposition is based on [17], discussions held at the Berley multimedia workshop [4], and on ongoing joint work on multimedia synchronization between L.Ludwig, Vicor, Palo Alto, CA; and the author.

1. A multimedia call: A camera and a microphone are attached to a workstation and the audio and video data are presented simultaneously at a remote workstation. A telepointer or screen sharing may be used in real-time to discuss a spreadsheet or edit a business report.

2. Surrogate travel through a planned building: The user performs actions like "turn right and open door" by pressing appropriate buttons on a graphical user interface. The sound of the door opening and an appropriate video are presented. Supplementary information appears as overlay text.

The first example deals with some information that should be presented in the same (or nearly the same) order as it was originally collected. It is therefore an example of **live synchronization.** Here this involves synchronization of audio, video, and application events (for example, the movement of the remote telepointer in time with the verbal comments of the person controlling the pointer). With the current interest in real-time, point-to-point connections with audio and video channels, live synchronization has become an important research area [3; 36].

The second example illustrates the use of synchronization in a retrieval scenario. We call this **synthetic synchronization.** At presentation time, the various pieces of information (segments) must be properly ordered and synchronized in time. One way to do this is through *absolute synchronization operations*, for example, "present segment A now/at time T." Alternatively, *relative synchronization operations* such as "present segment A after/simultaneously with/etc. segment B," can be used [24]. These operations can apply either to the information as a whole or to individual information entities, identified by "event stamps" [32]. In some cases, either absolute or relative synchronization could be used. For example to synchronize a mouse event with other information, we use an absolute synchronization operation: "present picture now". The second possibility is to view it as an event establishing a relationship to other information and requiring a relative operation for synchronization: "present mouse pointer (at position) during frame X." In any case, in order to schedule presentations, the system must internally translate relative synchronization operations to absolute operations. In the example above, the system knows the time for presenting frame X, and can use an absolute synchronization operation to present the mouse pointer at this time. Languages and programming interfaces for describing the synthetic synchronization of multimedia events are just now under study, development or available in first versions [8; 20; 23]. In both synchronization modes, the system needs support for absolute synchronization. The usage of relative or absolute synchronization is determined by the system¡s synchronization support and the application¡s context.

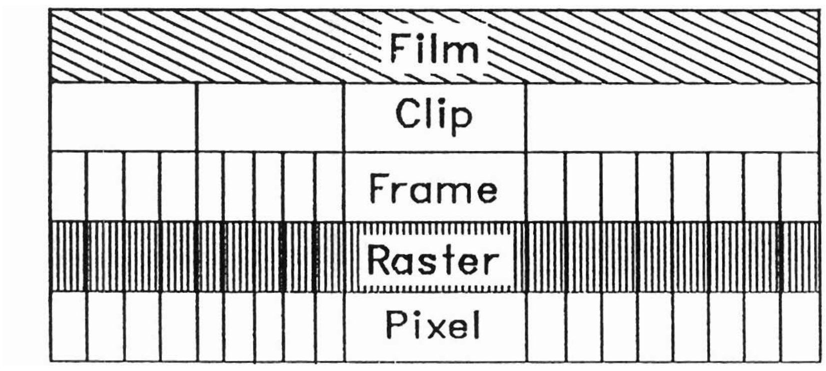

Figure 1. Granularity of Media: LDUs

In either live or synthetic synchronization, relationships between units of information must be specified. To go beyond abstract theory, these "units" must be identified. Let us therefore introduce the concept of the "granularity" of a media stream. For instance, synchronization could be performed between pixels of a video stream or between complete images (see Figure 1). For the following discussion, let us view CM streams as a sequence of logical data units (LDUs). Each kind of medium has an associated LDU. The LDU of a video stream c~n either be a single image or a well-defined sequence of images. An audio sample or an audio block (1/75 sec. duration) are two possible LDUs for audio streams. In the following, the single image is the video LDU and the audio block is the audio LDU.

3. Synchronization in Hybrid Distributed Multimedia Systems

Early prototypes of distributed and local multimedia systems, such as the Integrated Media Architecture Laboratory (IMAL), conceived at Bell Communications Research in Red Bank [18]; or the Muse and Pygmalion system of MIT's Project Athena [15; 6; 21], were based on a "hybrid" system structure [12]. In this structure, CM is not only processed by the devices used by workstations for general display, sound output, communication, etc. Most of real-time processing is kept out of the main CPU and host operating system. For instance, instead of sending video data to a workstation via a LAN for presentation in a window on the display, video data are sent over special cables directly to a separate video monitor. These devices are, however, attached to the workstation and controlled only by the workstation's software.

Live synchronization between various CM streams is directly performed by the dedicated processing devices, e.g. audio and video being transmitted as analog TV-signals over MIT's campus net in Project Athena. No further action is necessary.

The correlation between various DM streams, objects or information units in the live synchronization mode occurs very rarely in these hybrid environments. A rough implementation can be achieved by time-stamping DM objects and playing them back on cue from timer events at the remote workstation. Difficulties with establishing a global time can be solved by various techniques using software, hardware, or a combination of the two [26]. An example of a hardware solution is for each workstation to derive its local time from a master time signal. This time signal could be broadcast over WANs, or be received by radio (e.g., from WWV in the US or from the DCF 77 AM sender in Germany).

In hybrid structures, it is very difficult to achieve tight synchronization between DM and CM. DM and CM data are transmitted over different networks and processing nodes having different end-to-end delay characteristics. End-to-end delay over CM paths is typically shorter than that for DM. As it turns out, it is difficult and expensive (in terms of buffer capacity) to delay CM data delivered from devices like cameras or microphones. If DM is faster than CM, buffering and time stamping as described above could be used to slow down the data stream, but this situation very rarely occurs.

In the DiME project, everything was presented as soon as it arrived. When CM had a smaller delay than DM, no buffering was performed. DM LDUs never arrived sooner than the related CM LDUs.

In synthetic synchronization, where data is retrieved from external storage devices, one must contend with another type of delay, that of the control signals to the storage devices. The physical control paths of the attached devices range from slow RS-232 C interfaces to relatively fast SCSI interconnections. Between the issuing of a "start" command by a workstation application until the commencement of physical delivery of a video sequence, we experienced a maximal delay of about 500 msec. If positioning must be done by, say a VCR, this could take considerably longer.

The main reasons for this delay are:

- The system software is in principle not designed to cope with the real-time demands of physical interfaces such as the RS-232.
- The same device driver is often used for controlling many devices at the same time. A shared RS-232 C interface[2] may introduce access conflicts between the various devices. We found this to be negligible.
- Most of the external devices process queued work requests in an "as fast as possible" mode. The control interfaces do not include options to specify when information is to be presented (via commands like "play at time T").

Decoupling the seek time and the playback delay (e.g., with separate interface commands) allows for a nearly deterministic behavior of the whole system. More interesting is the decomposition of the end-to-end delay d_{tot} into a fixed component d_{fix} and a variable part d_{var}:

$$d_{tot} = d_{fix} + d_{var}$$

The variable part with its distribution originates from the above mentioned phenomena. Fortunately, the variance is not considerable and we can assume system- and device-specific values for d_{fix} to derive d_{tot} (we experienced typically about 500 msec). In multimedia retrieval applications such a delay can easily be tolerated. Note, in a distributed environment, additional delay is introduced by system and communication software.

Synchronization in hybrid approaches is most commonly presented at the API as a device-dependent set of library functions. Some libraries just map the direct control functions of the attached device onto functions in a high-level language. More sophisticated interfaces are based on client/server functionality, but introduce another component of the variable delay d_{var}. However, this is typically relatively minor. The same applies to any object-oriented interface: Such a presentation of data and control introduces some additional processing to dispatch any command to an external device. As most of the object-oriented APIs in hybrid system structures are not implemented in a real-time environment, additional (often significantly variable) delay may be introduced.

4. Synchronization in Unified Digital Distributed Multimedia Systems

Achieving live synchronization between CM streams is generally more challenging in a "unified" system structure, where CM and DM are routed through the same network and workstation. To the user, or even at a high-level programming interface, all data appear to be processed under full control of the application. CM should be handled similar to DM. It is a very common goal of the system designers to suggest this unified impression at the user interfaces in order to reduce the view of the system's complexity.

On the other hand, in most of the application scenarios, audio and video processing demands a well-defined end-to-end delay and imposes bandwidth requirements [35]. Real-time coding, mixing, and compression (like JPEG, MPEG, DVI) often requires dedicated hardware and software. Therefore, we encompass unified multimedia workstations to include such specialized components along

[2] The "port expander" of DiME, a software-controlled, bidirectional, stand-alone one-in-to-many-out RS-232 C switch is one type of shared interface.

with general-purpose components. Note, such a component can be a DSP within a workstation or, even a chip comprising four CPUs and, e.g, a DVI coding/decoding special purpose DSP. We do not foresee the need (or the technology) for one cost-effective, general-purpose engine to do all types of CM and DM processing in real-time.

In a traditional DM environment, software makes use of system support for time-sharing. Similarly, in CM processing, scheduling and reservation can be performed by an operating system that provides a RTE (real-time environment). Processes or threads running in such an environment are, in general scheduled according to real-time scheduling techniques (earliest deadline first, rate monotonic, etc.). With this approach, LDUs can be used as data types, abstract data types or objects (to hide time constraints from the user) [33].

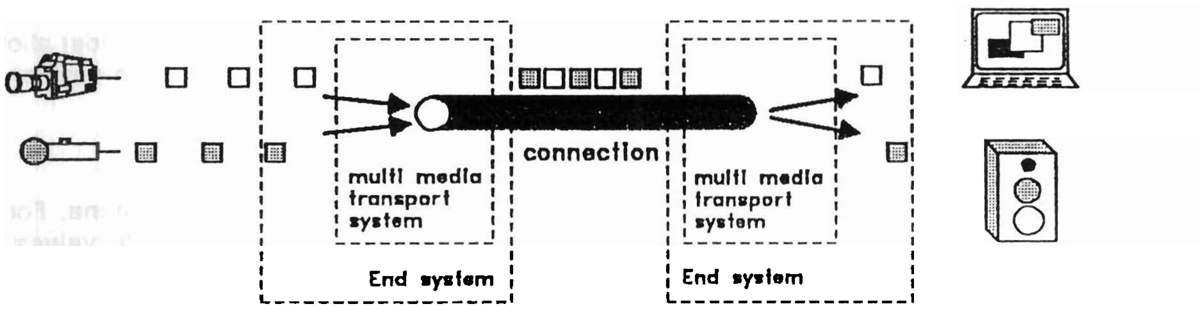

Figure 2. Interleaved data streams

Let us now focus on synchronization between data streams. In a distributed multimedia system where multiple, related streams originate in the same workstation, a very common and straightforward approach is to interleave the different LDU streams (e.g. audio and video) into one composite LDU stream. These combined LDUs are, e.g, time stamped (together) at the source and separated at the sink and presented according to the time stamp. In our first (PS/2-based) HeiTS prototype we make use of the ActionMedia II audio visual attachment (DVI) and apply this method (see Figure 2) to cope with lip synchronization. Synchronization was easy to implement using today's operating systems and networks. This system was shown at CeBIT'92 (the European COMDEX) in Hannover, Germany.

We found that any audio glitch is perceived immediately and can not be tolerated, whereas a small variance in the video rate leading to, e.g., display of same picture twice, is difficult to perceive. Audio imposes more stringent requirements than video does. In the design of HeiTS and subsequent prototypes we take this into account as follows:

1. We define different quality of service (QOS) at connection set-up for audio, video and other CM streams. and choose the QOS parameters to ensure that there are no audio faults and either no or very few video glitches. We can thus assure that whenever a shortage of resources occurs, it will first affect the video connections.

2. In HeiTS we experiment with rate monotonic scheduling where audio LDUs occur with higher frequency than video LDUs and the processing of audio LDUs is scheduled with higher priority than that of video LDU processing. Even with an additional segmenting of video LDUs (original size 5 kByte) into 2 data units (about 4 kByte is the largest size to be processed as one packet in all layers of our multimedia communication system) audio is prioritized over video.

3. We use resource management during call establishment to help prevent glitches [14].

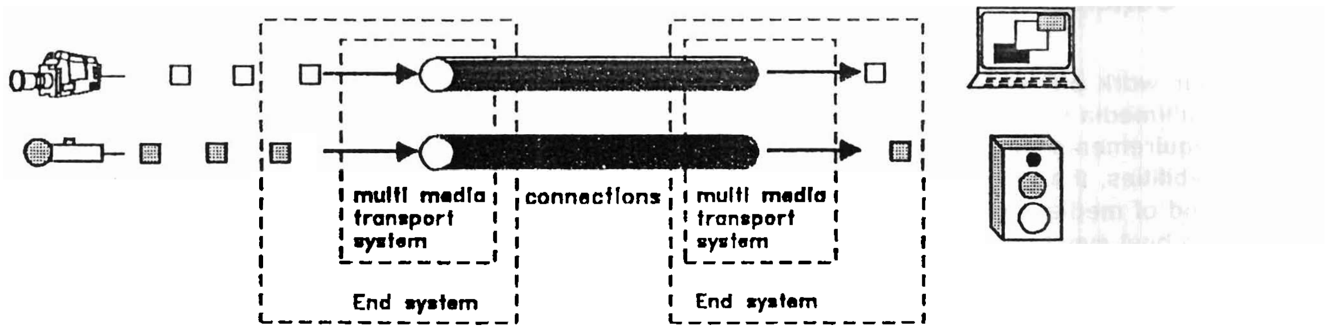

Figure 3. Live synchronization over various connections

The evolving HeiTS prototype (on the PS/2 and on the RS/6000) has the ability to either (1) combine audio and video, or (2) use separate connections for different CM (or DM) streams, as shown in Figure 3. By imposing the same end-to-end delay on related CM streams (by choosing an absolute end-to-end delay and limiting the jitter of the LDUs to 0 msec), live synchronization can be guaranteed. 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. Both temporal skews will sometimes be noticed, but can easily be tolerated without any inconvenience by the user [22].[3] This asymmetry is very plausible: In a conversation where 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.

Another (very elaborate) implementation of live synchronization allows these effects to be used to advantage and even guarantees synchronization between various sources: A logical time system (LTS) is introduced. Presentation of the data is performed based on a comparison of the LTS with the real-time clock of the destination workstation ([2] introduces and uses LTS in the ACME client server approach). All LDUs are time-stamped at the sources explicitly or implicitly (by the block or sample number). Sinks and sources interacting in a logical group are tied to the same LTS. The RTE provides the presentation of LDUs according to the LTS and the current real time. Scheduling techniques as discussed above are capable of providing this in-time playback. For synthetic synchronization, the application defines an interaction time and an event to happen. From the implementation point of view, application code can be linked into the RTE, or time-sharing code can be called from the RTE (upcall).

Note that we need and have isochronous data streams at the sink devices. It is not necessary to have isochronous communication in all of the components involved in a communication path. For instance, it is sufficient to have upper bounds on delay for each individual component. With this assumption, buffering can be used to achieve isochronism, but this can lead to a waste of storage, especially for video. Restricting jitter ("jitter control") at intermediate gateways drastically reduces total buffer requirements. Ferrari's approach [7] can also be applied to system components in the end-systems, leading to what we call "weak isochronous" behavior.

---

[3] These values are taken from of a graph showing "annoyance" with different audio/video time skews.

5. Conclusion

Our work on synchronization is being performed as part of the HeiTS project, the high-speed multimedia transport system at IBM ENC. All the experience to data indicates that synchronization requirements should not be viewed as an isolated issue. The system structure, the hardware capabilities, the operating system capabilities, the communication subsystem and its protocols, the kind of media, the coding techniques, and even the envisaged types of applications, all influence the best synchronization techniques to use.

Synchronization is part of many system components (operating and communication systems, multimedia documents, databases, presentation techniques, etc.). Nevertheless it is not compulsory to solve the same problem in each component; a system-wide solution should be attempted. By "system-wide" we mean the various local components as well as distributed solutions. In HeiTS we drive and are driven by two different platforms: OS/2 and AIX, and we are working on a common strategy for the implementation.

Acknowledgements

This work gained considerably from discussions with Thomas Meyer and from several hints from Manny Farber.

References

[1] *David P. Anderson;* **Meta-Scheduling for Distributed Continous Media;** Computer Science Division (EECS) Report No. UCB/CSD 90/599, U. C. Berkeley, Berkeley CA, October 1990.

[2] *David P. Anderson; George Homsy;* **Synchronization Policies and Mechanisms in a Continuous Media I/O Server;** International Computer Science Institute, Technical Report no. 91-003, Berkeley, 1991.

[3] *David P. Anderson; George Homsy;* **Abstractions for Continuous Media in a Network Window System;** International Conference on Multimedia Information Systems, Singapore, January 1991.

[4] **International Workshop on Network and Operating System Support for Digital Audio and Video;** International Computer Science Institute (ICSI), Berkeley, CA, Nov. 8-9, 1990.

[5] *Gerold Blakowski, Hübel, Langrehr;* **Tools for Specifying and Executing Synchronized Multimedia Presentations;** 2nd International Workshop on Network and Operating System Support for Digital Audio and Video, Heidelberg, Nov. 18-19, 1991.

[6] *Georg Champine, Daniel Geer, William Ruh;* **Project Athena as a Distributed Computer System;** IEEE Computer, vol.23 no.9, September 1990, pp.40-51.

[7] *Domenico Ferrari;* **Design and Application of a Delay Jitter Control Scheme for Packet-Switching Internetworks;** 2nd International Workshop on Network and Operating System Support for Digital Audio and Video, Heidelberg, Nov. 18-19, 1991.

[8] *Charles F. Goldfarb;* **HyTime: A Standard for Strucured Hypermedia Exchange;** IEEE Computer, vol.24, no.8, Aug.1991, pp. 81-84.

[9] *Kevin Harney, Mike Keith, Gary Lavelle, Lawrence D. Ryan, Daniel J. Stark;* **The i750 Video processor: A Total Multimedia Solution;** Communications of the ACM, vol.34, no.4, April 1991, pp.64-78.

[10] *Dietmar Hehmann, Ralf Guido Herrtwich, Werner Schulz, Thomas Schütt, Ralf Steinmetz;* **HeiTS - Architecture and Implementation of the Heidelberg High-Speed Transport System;** 2nd International Workshop on Network and Operating System Support for Digital Audio and Video, Heidelberg, Nov. 18-19, 1991.

[11] *Ralf Guido Herrtwich;* **The Role of Performance, Scheduling, and Resource Reservation in Multimedia Systems;** Proc. Operating Systems in the Nineties and Beyond, Lecture Notes of Computer Science, Springer, 1991.

[12] *Lutz Henkel, Heinrich J. Stüttgen;* **Transportdienste in Breitbandnetzen**; GI Conference, Communication in Distributed Systems, February 1991, Mannheim, Germany.

[13] *Ral' Guido Herrtwich, Ralf Steinmetz;* **Towards Integrated Multimedia Systems: Why and How**; Informatik-Fachberichte, no. 293, Springer Verlag, 1991, pp.327-342.

[14] *Ralf Guido Herrtwich, Ramesh Nagarajan, Carsten Vogt;* **Guaranteed-Performance Multimedia Communication Using ST-II Over Token Ring**; submitted to "International Conference on Distributed Computer Systems", June 92, to appear as ENC technical report.

[15] *Matthew E. Hodges, Russel M. Susnett, Mark S. Ackerman;* **A Construction Set for Multimedia Applications**; IEEE Software Magazine, Jan. 1989, pp. 37-43.

[16] *T.D.C. Little, A. Ghafoor;* **Synchronization and Storage Models for Multimedia Objects**; IEEE Journal on Selected Areas in Communication, vol.8, no.3, Apr. 1990, pp. 413-427.

[17] *Thomas D.C. Little, Arif Ghafoor;* **Network Considerations for Distributed Multimedia Objects Composition and Communication**; IEEE Network Magazine, vol.4 no.6, Nov. 1990, pp. 32-49.

[18] *L.F.Ludwig, D.F.Dunn;* **Laboratory for Emulation and Study of Integrated and Coordinated Media Communication**; Frontiers in Computer Technology, Proc. of the ACM SIGCOMM '87 Workshop, Aug. 11-13,1987.

[19] **M-Motion Video Adapter/A, User's Guide, Product Description**; IBM 1990.

[20] *Daniel J. Moore;* **Multimedia Presentation Development using the Audio Visual Connection**; IBM Systems Journal, Vol.29, No.4, 1990,pp.494-508.

[21] *W.E. Mackay, W.- Treese, D. Applebaum, B. Gardner, B. Michon, E. Schlusselberg, M. Ackermann, D. Davis* **Pygmalion: An Experiment in Multimedia Communication** Proceedings of SIGGRAPH 89, Boston, July 1989.

[22] *Alan Murphy;* **Lip Synchronization**; Personal Communication on a Set of Experiments, 1990.

[23] *Cosmos Nicolaou;* **An Architecture for Real-Time Multimedia Communication Systems**; IEEE Journal on Selected Areas in Communication, vol.8, no.3, April 1990, pp.391-400.

[24] *Jonathan B. Postel, Gregory G.Finn, Alan R. Katz, Joyce K.Reynolds.* **An Experimental Multimedia Mail System.** ACM Transactions on Ofiice Information Systems, vol.6, no.1, Jan. 1988, pp. 63-81.

[25] *K. Ravindran;* **Real-time Synchronization of Multimedia Data Streams in High Speed Packet Switching Networks**; Technical Report, Department of Computing and Informations Sciences, Kansas State University, Manhatten, KS 66506, USA, February 1991.

[26] *Parameswaran Ramanathan, Kang G. Shin, Ricky W. Butler;* **Fault Tolerant Clock Sysnchronization in Dstributed Systems**; IEEE Computer, vol.23, no.10, October 1990, pp.33-42.

[27] *Johannes Rückert, Hermann Schmutz, Bernd Schöner, Ralf Steinmetz;* **A Distributed Multimedia Environment for Advanced CSCW Applications**; IEEE Multimedia '90, Bordeaux, Nov. 15-17, 1990.

[28] *Ralf Steinmetz, Reinhard Heite, Johannes Rückert, Bernd Schöner;* **Compound Multimedia Objects - Integration into Network and Operating Systems**; International Workshop on Network and Operating System Support for Digital Audio and Video, International Computer Science Institute, Berkeley, Nov. 8-9, 1990

[29] *Doug Shepherd, Michael Salmony.* **Extending OSI to Support Synchronisation Required by Multimedia Applications.** Computer Communications, vol.13, no.7, Sept. 1990, pp.399-406.

[30] *Ralf Steinmetz, Johannes Rückert, Wilfried Racke;* **Multimedia-Systeme**; Informatik Spektrum, Springer Verlag, vol.13, no.5, 1990, pp.280-282.

[31] *Bernd Schöner, Johannes Rückert, Ralf Steinmetz;* **Media Related Requirements for Communication Services**; 6th IEEE International Workshop on Telematics, Corea, September 1991.

[32] *Ralf Steinmetz;* **Synchronization Properties in Multimedia Systems**; IEEE Journal on Selected Areas in Communication, vol. 8, no. 3, April 1990, pp. 401-412.

[33] *Ralf Steinmetz, Christian Fritzsche;* **Abstractions for Continuous-Media Programming**; 2nd International Workshop on Network and Operating System Support for Digital Audio and Video, Heidelberg, Nov. 18-19, 1991; and to appear in Computer Communications, vol. 15, no. 4, Juli/August 1992.

[34] *Ralf Steinmetz, Ralf Guido Herrtwich;* **Integrierte verteilte Multimedia-Systeme**; Informatik Spektrum, Springer Verlag, vol.14, no.5, October 1991, pp.280-282.

[35] *Ralf Steinmetz, Thomas Meyer;* **Modelling Distributed Multimedia Applications**; IEEE Int. WS on Advanced Communications and Applications for HS Networks, München, March 1992.

[36] *Martina Zitterbart;* **High-Speed Transport Components**; IEEE Network Magazine, vol.5, no.1, January 1991, pp.54-63.