A Media Synchronization Survey: Reference Model, Specification, and Case Studies

Gerold Blakowski and Ralf Steinmetz, *Senior Member, IEEE*

Abstract— **Researchers have addressed multimedia synchronization from various perspectives. The major issues include how to specify and how to implement synchronization. Especially in the specification area a variety of techniques have been published and implemented. This survey summarizes briefly synchronization requirements, presents a multimedia synchronization reference model, shows details of various specification approaches and applies the reference model to compare existing prominent approaches as case studies.**

I. Introduction

ADVANCED multimedia systems are characterized by the integrated computer-controlled generation, storage, communication, manipulation and presentation of independent time-dependent and time-independent media [30], [68]. The key issue which provides integration is the digital representation of any data and the synchronization of and between various kinds of media and data.

The word *synchronization* refers to time. *Synchronization* in multimedia systems refers to the temporal relations between media objects in a multimedia system. In a more general and widely used sense some authors use synchronization in multimedia systems as comprising content, spatial and temporal relations between media objects. We differentiate between time-dependent and time-independent media objects. A time-dependent media object is presented as a media stream. Temporal relations between consecutive units of the media stream exist. If the presentation durations of all units of a *time-dependent media object* are equal, it is called a *continuous media object*. For example, a video consists of a number of ordered frames; each of these frames has a fixed presentation duration. A *time-independent media object* is any kind of traditional media like text and images. The semantic of the respective content does not depend upon a presentation according to the time domain.

Synchronization between media objects comprises relations between time-dependent media objects and time-independent media objects. A daily example of synchronization between continuous media is the synchronization between the visual and acoustical information in television. In a multimedia system, the similar synchronization must be provided for audio and moving pictures. An example of temporal relations between time-dependent media and time-independent media is a slide show. The presentation of slides is synchronized with the commenting audio stream. To realize a slide show in a multimedia system, the presentation of graphics has to be synchronized with the appropriate units of an audio stream.

Synchronization is addressed and supported by many system components including the operating system, communication system, databases, documents and often even by applications. Hence, synchronization must be considered at several levels in a multimedia system.

The operating system and lower communication layers handle single media streams with the objective to avoid jitter during the presentation of the units of one media stream (e.g., [20], [57], [59], [61]). For example, users will be annoyed if an audio presentation is interrupted by pauses or if clicks result in short gaps in the presentation of the audio clip.

On top of this level is located the run-time support for the synchronization of multiple media streams (e.g., [4], [5], [18], [35]). The objective at this level is to maintain the temporal relations between various streams. In particular the skew between the streams must be restricted. For example, users will be annoyed if they notice that the movements of the lips of a speaker do not correspond to the presented audio.

The next level holds the run-time support for the synchronization between time-dependent and time-independent media as well as the handling of user interactions (e.g., [10], [45], [46], [52]). The objective is to start and stop the presentation of the time-independent media within a tolerable time interval, if some previously defined points of the presentation of a time-dependent media object are reached. The audience of a slide show is annoyed if a slide is presented before the audio comment introduces a new picture. A short delay after the start of the introducing comment is tolerable or even useful.

The *temporal relations* between the media objects must be specified. The relations may be specified implicitly during capturing of the media objects, if the goal of a presentation is to present the media in the same way as they were originally captured. This is the case of audio/video recording and playback.

The temporal relations may also be specified explicitly in the case of presentations that are composed of independently captured or otherwise created media objects (e.g., [8], [16], [33]). In the slide show example, a presentation designer selects the appropriate slides, creates an audio object and defines the units of the audio presentation stream where the slides have to be presented. Also, the user interactivity may be part of a presentation and the temporal relations between media

Manuscript received August 12, 1994; revised June 10, 1995.

The authors are with IBM European Networking Center, Vangerowstr. 18, D-69115 Heidelberg (e-mail: gblakowski@vnet.ibm.com and steinmetz @vnet.ibm.com).

Publisher Item Identifier: S 0733-8716(96)00242-9.

objects and user interactions must be specified. The tools that are used to specify the temporal relations are located on top of the previous levels.

In recent years, in nearly every multimedia workshop and conference, many synchronization-related contributions have been provided. Most of the contributions address only issues of one or a subset of the levels or regard synchronization only from a specific viewpoint and they are partly overlapping.

The objective of this contribution is to provide an integral view to the area of multimedia synchronization. Therefore, we focus on a consistent definition of synchronization-related terms, synchronization requirements, the synchronization specification, synchronization between media objects and the synchronization related to structuring of multimedia systems. An emphasis is also put on the synchronization in a distributed environment that introduces additional complexity but is very important for client/server architectures and future teleservices like access to information bases using an information highway.

Descriptions of low-level technical support for media synchronization, like Earliest Deadline First (EDF) or rate monotonic scheduling in the operating system, isochronous transport services and support for single media streams, are not part of this contribution.

In Section II, the basic terms of synchronization are defined. Subsequently, in Section III, the requirements for synchronization resulting from user perception of multimedia presentations are described. A synchronization reference model is presented in Section IV that allows the structuring of the levels of synchronization and the classification of existing synchronization systems. Section V provides an overview for synchronization specification methods. Some prominent and representative systems are presented and classified according to the synchronization reference model in Section VI. A summary and outlook are given in the last section.

II. Notion of Synchronization

A. Multimedia Systems

Several definitions for the terms multimedia application and multimedia system are described in the literature. Three criteria for the classification of a system as a multimedia system can be distinguished: the number of media, the types of supported media, and the degree of media integration.

The most simple criterion is the number of media used in an application. Using only this criterion, even a document processing application that supports text and graphics can be regarded as a multimedia system [31].

The types of supported media are an additional criterion [30]. In this case, we distinguish between time-dependent and time-independent media. A time-independent media object is usually presented using one presentation unit. An example is a bitmap graphic. Time-dependent media objects are presented by a sequence of presentation units. An example is a motion picture sequence without audio (i.e., a video sequence) presented frame after frame. Because the integration of time-dependent media objects is a new and essential aspect in information processing, some authors define a multimedia system as a system that supports the processing of more than one medium with at least one time-dependent medium [13].

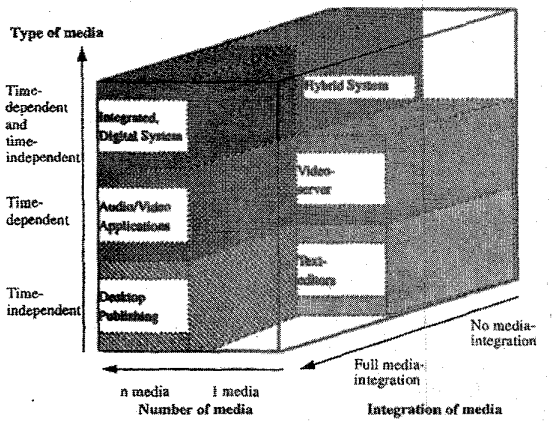

Fig. 1. Classification of media use in multimedia systems.

The degree of media integration is the third criterion [30]. In this case, integration means that the different types of media remain independent but can be processed and presented together.

Combining all three criteria, we propose the following *definition of a multimedia system:* a system or application that supports the integrated processing of several media types with at least one time-dependent medium.

Fig. 1 classifies applications according to the three criteria. The arrows indicate the increasing degree of multimedia capability for each criterion.

Integrated digital systems can support all types of media, and due to digital processing, they may provide a high degree of media integration. Systems that handle time-dependent analog media objects and time-independent digital media objects are called *hybrid systems* [32], [66]. The disadvantage of hybrid systems is that they are restricted with regard to the integration of time-dependent and time-independent media, because, for example, audio and video are stored on different devices than time-independent media objects and multimedia workstations must comprise both types of devices. The same applies to the interconnection between workstations. Audio/Video-applications that implement functionalities of consumer devices, like video recorders, often do not support the integration of audio and video and time-independent media objects. In addition, they often do not support the separate handling of audio and video media objects. Single time-dependent media objects are often supported by audio and video servers. Traditional desktop-publishing systems are examples of integrated processing of time-independent media objects.

B. Basic Synchronization Issues

Integrated media processing is an important characteristic of a multimedia system. The main reasons for these integration demands are the inherent dependencies between the information coded in the media objects. These dependencies must be reflected in the integrated processing including storage,

manipulation, communication, capturing and, in particular, the presentation of the media objects.

The word *synchronization* refers to time. As mentioned, some authors use synchronization in multimedia systems in a more general and widely used sense as comprising content, spatial and temporal relations between media objects.

1) Content Relations: Content relations define a dependency of media objects on some data.

An example of a content relation is the dependency between a filled spreadsheet and a graphic that represents the data listed in the spreadsheet. In this case, the same data are represented in two different ways. Another example is two graphics that are based on the same data but show different interpretations of the data.

For integrated multimedia documents, it is useful to express these relations explicitly to enable an automated update of different views of the same data. In this case, only the data are edited and for the views, the kind of dependencies of the data and the presentation rules are defined. All views of the data are generated automatically and cannot be edited directly. An update of the data triggers an update of the related views. This technique is, for example, used in database systems and may also be used for the different media of a multimedia system.

In general, the implementation of content relations in multimedia systems is based on the use of common data structures or object interfaces that are used to present objects using different media.

2) Spatial Relations: The *spatial relations* that are usually known as layout relationships define the space used for the presentation of a media object on an output device at a certain point of time in a multimedia presentation. If the output device is two-dimensional (e.g., monitor or paper), the layout specifies the two-dimensional area to be used.

In desktop-publishing applications, this is usually expressed using *layout frames*. A layout frame is placed and a content is assigned to this frame. The positioning of a layout frame in a document may be fixed to a position in a document, to a position on a page or it may be relative to the positioning of other frames.

The concept of frames can also be used to specify where the presentation units of a time-dependent media object are placed. For example, video frames may be positioned using layout frames. In window-oriented systems, a frame or group of frames may be represented by a window. A window may be resized, moved, iconized, etc. and gives the user additional manipulative freedom to adapt the presentation to the requirements.

Experimental three-dimensional output devices like holographic experiments and three-dimensional projection allow the user to create three-dimensional presentations. In usual window systems, the third dimension is only expressed in terms of overlapping windows. Stereo audio output devices also have layout that defines the positioning of an audio source in a presentation. This is, for example, used in audio and video conferences to give a participant the impression of a seat ordering [49] that is related to the placement of pictures or videos of the other conference participants. This gives the user a more natural communication impression, makes it easier to follow a discussion and therefore, increases the user's acceptance.

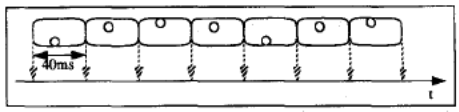

Fig. 2. Intra-object synchronization between frames of a video sequence showing a jumping ball.

3) Temporal Relations: Temporal relations define the temporal dependencies between media objects. They are of interest whenever time-dependent media objects exist.

An example of a temporal relation is the relation between a video and an audio object that are recorded during a concert. If these objects are presented, the temporal relation during the presentations of the two media objects must correspond to the temporal relation at the time of recording.

These time relations are what we will understand to be synchronization in multimedia systems.

4) Comment: All three types of synchronization relations are important for integrated digital multimedia systems and are meanwhile subject to standardization efforts like MHEG [52] and Hypermedia/Time-based Structuring Language (HyTime) [36].

Content and spatial relations are well-known from publishing and integrated application systems with databases, spreadsheets, graphical tools and word processing systems. The key aspect in multimedia systems is the temporal relations derived from the integration of time-dependent media objects. Therefore, the rest of this contribution addresses temporal relations only.

C. Intra- and Inter-Object Synchronization

We distinguish between time relations within the units of one time-dependent media object itself and time relations between media objects. This separation helps to clarify the mechanisms supporting these types of relations, which are often very different:

- Intra-object synchronization: intra-object synchronization refers to the time relation between various presentation units of one time-dependent media object. An example is the time relation between the single frames of a video sequence. For a video with a rate of 25 frames per second, each of the frames must be displayed for 40 ms. Fig. 2 shows this for a video sequence presenting a bouncing ball.
- Inter-object synchronization: inter-object synchronization refers to the synchronization between media objects. Fig. 3 shows an example of the time relations of a multimedia synchronization that starts with an audio/video sequence, followed by several pictures and an animation that is commented by an audio sequence.

1) Time-Dependent Presentation Units: Time-dependent media objects usually consist of a sequence of information units. Such information units are known as *Logical Data Units (LDU's).*

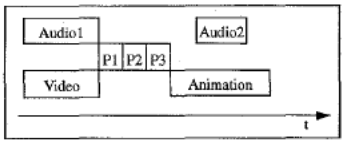

Fig. 3. Inter-object synchronization example that shows temporal relations in a multimedia presentation including audio, video, animation, and picture objects.

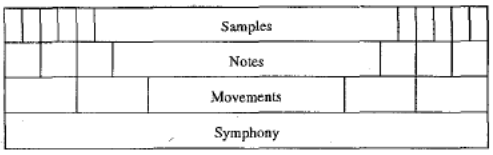

Fig. 4. The LDU hierarchy.

In many cases, several granularity levels of LDU's in a media object exist. An example is the symphony "The Bear" by Joseph Haydn (Fig. 4). It consists of four movements: *vivace assai*, *allegretto*, *minuet*, and *finale*. Each of these movements is an independent, self-contained part of the composition. It consists of sequences of notes for different instruments. In a digital system, each note is a sequence of sample values. In the case of CD-quality with PCM coding without compression, a sample rate of 44 100 Hz with two channels and 16-bit resolution per channel is used. On a CD, these sample values are combined to blocks of 1/75 s duration.

The level of granularity used is application-dependent. It is possible to look at the whole symphony, the movements, the notes, the samples or the combined samples as LDU's. The selection of the LDU's depends on the operations that should be performed on a media object. For simple presentation operations like "play," the whole symphony or the movements are the useful LDU's. For applying instrument-based playing techniques, the notes as smallest description units in the musical area are the useful granularity. For the signal processing task, the operations are based on samples or blocks of samples.

Another example is an uncompressed video object that is divided into scenes and frames. The frames can be partitioned in areas of $16*16$ pixels. Each pixel consists of luminance and chrominance values. All these units are candidates for LDU units.

In a video sequence coded in the MPEG [38] format, redundancies within subsequent frames may be used to reduce the amount of digital data used to represent the media object (inter-frame compression). In this case, a sequence of frames that are inter-frame compressed can be regarded as an LDU.

The levels of granularity imply a hierarchical decomposition of media objects. Often there are two kinds of hierarchies. The first is a content hierarchy that is implied by the content of the media object. This is the hierarchy of symphony, movement and notes in the symphony example. The second is the coding hierarchy based on the data encoding. For the symphony example, the hierarchy may be a media object representing a movement, that is divided into blocks of samples. The samples are the lowest level of the coding hierarchy.

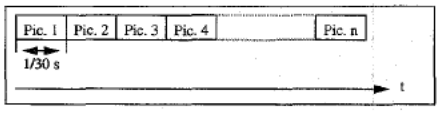

Fig. 5. Example of video LDU's.

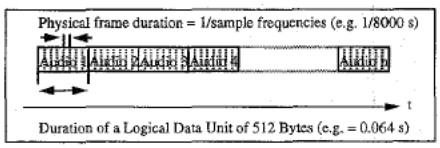

Fig. 6. Example of audio LDU's.

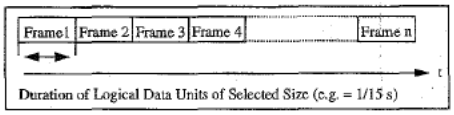

Fig. 7. LDU size selected by user.

In addition, LDU's can be classified into closed and open LDU's. *Closed LDU's* have a predictable duration. Examples are LDU's that are parts of stored media objects of continuous media like audio and video, or stored media objects with a fixed duration. The duration of *open LDU's* is not predictable before the execution of the presentation. Open LDU's typically represent input from a live source, for example, a camera or a microphone, or media objects that include a user interaction.

2) Classification of Logical Data Units: For digital video, often the frames are selected as LDU's. For example, for a video with 30 pictures per second, each LDU is a closed LDU with a duration of 1/30 s (Fig. 5).

In the case of the basic physical units being too small to handle, often LDU's are selected that block the samples into units of a fixed duration. A typical example is an audio stream where the physical unit duration is very small, therefore, LDU's are formed comprising 512 samples. In the example shown in Fig. 6, one sample is coded with one Byte, and hence, each block contains 512 Bytes.

Captured media objects usually have a natural basic duration of an LDU. In computer-generated media objects, the duration of LDU's may be selected by the user. An example of these user-defined LDU durations is the frames of an animation sequence. For the presentation of a two-second animation sequence, 30 to 60 pictures may be generated depending on the necessary quality. Thus, the LDU duration depends on the selected picture rate (Fig. 7).

Streams are more complex when the LDU's vary in duration. An example is the recording of events at a graphical user interface to replay a user interaction. In this case, an LDU is an event with a duration lasting until the next event. The duration of LDU's depends on the user interaction and varies accordingly (Fig. 8).

Open LDU's of unpredictable duration are given in the case that the LDU has no inherent duration. An example of an open LDU (i.e., an LDU with no inherent duration) is a user interaction in which the duration of the interaction is not known in advance (Fig. 9).

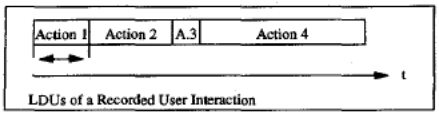

Fig. 8. LDU's of varying duration.

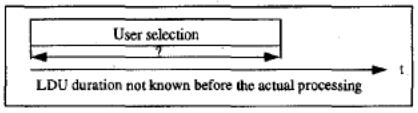

Fig. 9. An open LDU representing a user interaction.

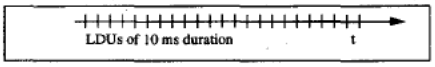

Fig. 10. LDU's of a timer.

TABLE I
TYPES OF LDU'S

	LDU duration defined during capturing	LDU duration defined by the user
Fixed LDU duration	Audio, Video	Animation, Timer
Variable, unknown LDU duration	Recorded interaction	User interaction

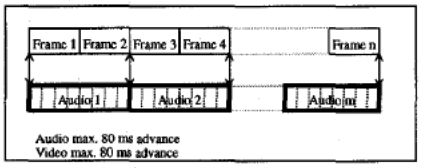

Fig. 11. LDU view of lip synchronization.

Timers can be regarded as streams of empty LDU's with a fixed duration (Fig. 10).

Table I gives an overview of the types of LDU's discussed above.

3) Further Examples: The following three examples show synchronization based on LDU's:

1) Lip synchronization demands tight coupling of audio and video streams. Synchronization can be specified by defining a maximal skew between the two media streams (Fig. 11).
2) A slide show with audio commentary demands that the change of slides be temporally related to the audio commentary (Fig. 12).
3) The following example shown in Fig. 13 will be used in Section V to demonstrate synchronization specification methods.

   A lip synchronized audio/video sequence (Audio1 and Video) is followed by the replay of a recorded user interaction (RI), a slide sequence (P1–P3) and an animation (Animation), which is partially commented using an audio sequence (Audio2). Starting the animation presentation, a multiple choice question is presented to the user (Interaction). If the user has made a selection, a final picture (P4) is shown.

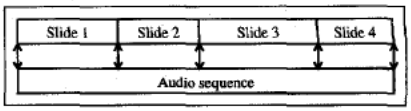

Fig. 12. LDU view of a slide show.

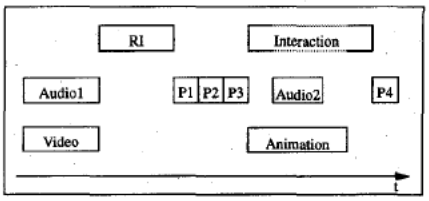

Fig. 13. Synchronization example.

D. Live and Synthetic Synchronization

The live and synthetic synchronization distinction refers to the type of the determination of temporal relations. In the case of live synchronization, the goal of the synchronization is to exactly reproduce at a presentation the temporal relations as they existed during the capturing process. In the case of synthetic synchronization, the temporal relations are artificially specified [40].

The following example shows aspects of *live synchronization:*

> Two persons located at different sites of a company discuss a new product. Therefore, they use a video conference application for person-to-person discussion. In addition, they share a blackboard where they can display parts of the product and they can point with their mouse pointers to details of these parts and discuss some issues like: "This part is designed to ..."

This example covers two live synchronization aspects: video conference demands lip synchronization of the audio and video, and the movement of the mouse pointer must be synchronized to the corresponding explanation given in the video conference.

An example of *synthetic synchronization* is a learning environment of a city realized by the Bank Street College of Education, New York [60]:

> A learner may perform a virtual voyage (surrogate travel) to an ancient Mayan city. Using a joystick, the learner walks through the jungle and explores the Mayan ruins. At the same time, he hears the sounds from the jungle. He can also take a closer look at the nature in the environment and can "visit" a video museum to get further information.

In the case of synthetic synchronization, temporal relations have been assigned to media objects that were created independently of each other. The synthetic synchronization is often used in presentation and retrieval-based systems with stored data objects that are arranged to provide new combined multimedia objects. A media object may be part of several

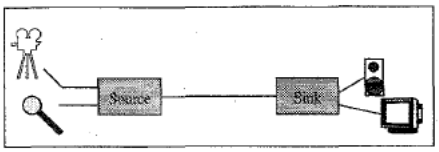

Fig. 14. Live synchronization without intermediate long-term storage.

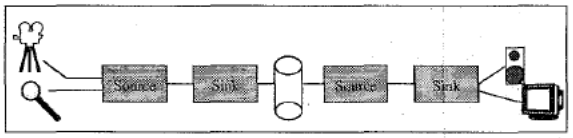

Fig. 15. Live synchronization with intermediate long-term storage and delayed presentation.

multimedia objects. For example, the same video clip about Germany may be part of a multimedia object that presents the countries of the European Union, as well as of a multimedia object that presents the countries qualified for the soccer world cup. Media objects of a multimedia object may be stored/located at different servers.

For synthetic synchronization, it is necessary to use a model for the specification and manipulation of temporal synchronization conditions and operations. Common examples [39] of such operations are:

- Presenting media streams in parallel.
- Presenting media streams one after the other (serial).
- Presenting media stream independent of each other.

1) Live Synchronization: A typical application of live synchronization is conversational services. In the scope of a source/sink scenario, at the source, volatile data streams (i.e., data being captured from the environment) are created which are presented at the sink (Fig. 14). The common context of several streams on the source site must be preserved at the sink. The source may be comprised of acoustic and optical sensors, as well as media conversion units. The connection offers a data path between source and sink. The sink presents the units to the user. A source and sink may be located at different sites.

A possible manipulation by the sink is to adapt the presentation to the available resources. This may be, for example, a change of resolution or a lower frame rate. To reduce resource usage, it is preferable that such adaptations be already performed at the source, in particular if the source and sink are distributed and connected by a network.

Another type of live synchronization is shown in Fig. 15 and includes storage which holds the encoded data. The presentation goal is the same as before, but the capturing and presentation are decoupled. In this case, it is possible to manipulate the presentation of the media. The presentation speed may be changed, and random access is possible (which is not possible in the scenario shown in Fig. 14).

In summary, we must emphasize that the primary demand of live synchronization is to present data according to the temporal relations which existed during the capturing process of the media objects.

2) Synthetic Synchronization: The emphasis of synthetic synchronization is to support flexible synchronization relations between media. In synthetic synchronization, two phases can be distinguished:

- In the specification phase, temporal relations between the media objects are explicitly defined.
- In the presentation phase, a run-time system presents data in a synchronized mode.

The following example shows this for the creation of a multimedia presentation:

> Four audio messages are recorded that relate to parts of an engine. An animation sequence shows a slow 360 degree rotation of the engine. With a software tool (e.g., a synchronization editor), time relations between the animation and matching audio sequences are specified. The media objects, along with the synchronization specification can be used by a presentation tool that executes the synchronized presentation.

Media objects that are stored in a live synchronization scenario can also be included in a synthetic synchronization playback.

Another variation of synthetic synchronization is the synchronization specification at run-time, such as:

> In a railway time-table information system, a user specifies his demands. An automatically-generated audio sequence presents this information to the user. During the presentation, a video sequence is displayed that shows how to go to the departure gateways and how to proceed at the destination. The synchronization between the generated audio and video is specified and performed at run-time.

E. Comment

In the case of live synchronization, the synchronization specification is implicitly defined during capturing. In the case of synthetic synchronization, the specification is done explicitly. If stored media objects are presented, presentation manipulations like changing the presentation speed and direction and direct access to a part of the object are possible. Adapting the presentation quality to the user demands or the capacity of the underlying system resources is possible in both cases.

User interaction in live synchronization includes only the interaction during capturing. Synthetic synchronization can include user interactions at run-time, for example, for navigation.

III. Presentation Requirements

For delivering multimedia data correctly at the user interface, synchronization is essential. It is not possible to provide an objective measurement for synchronization from the viewpoint of subjective human perception. As human perception varies from person to person, only heuristic criteria can determine whether a stream presentation is correct or not. In this section, results of some extensive experiments are presented that are related to human perception of synchronization between different media.

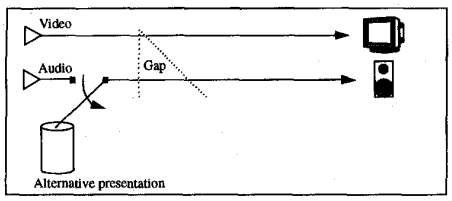

Fig. 16. The gap problem, restricted blocking.

Presentation requirements comprise, for intra-object synchronization, the accuracy concerning delays in the presentation of LDU's and, for inter-object synchronization, the accuracy in the parallel presentation of media objects.

For intra-object synchronization, we try to avoid any jitter in consecutive LDU's. Just as processes can wait for each other, using the method of blocking (e.g., in CSP—Calculus of Sequential Programming), a data stream of time-independent LDU's can also be stopped.

There is a different situation when looking at sequences of audio and moving pictures:

- What does the blocking of a stream of moving pictures mean for the connected output device?
- Should and can the last picture of a stream be shown during the blocking?
- Should, in the case of speech or music, a previous part be repeated during the blocking?
- How long can such a gap, as shown in Fig. 16, exist?

This situation has become known as the *gap problem* [67]. In the case of moving pictures, existing systems are solving the problem by simply switching the output device to dark or white, or by showing the last moving picture as a still picture. A practical solution must regard the time factor. It is significant, whether the duration of such a gap is a couple of milliseconds, a couple of seconds or even a couple of minutes. Only the actual application (and not the system) is able to select the best solution. Therefore, alternatives must be available that are selected independently of the expected blocking time. The concept of alternative presentations is indicated in Fig. 16. In this example, it is shown that in the case that the gap between the late video and audio exceeds a predefined threshold, the audio presentation is switched to an alternative presentation. In the case that the gap is shorter, the audio presentation may be stopped until the gap is closed. In general, in the case of blocking, alternative single pictures, sequences of pictures or audio signals can be presented, or previously used presentation units can be simply repeated. This method of process blocking, pertaining to streams of audio and video, is known as *restricted blocking*.

Restricted blocking uses as a means for resynchronization the repeated presentation of the last sample(s), or an alternative presentation. Another possibility is the *re-sampling* of a stream. The basic idea of re-sampling is to speed up or slow down streams for the purpose of synchronization. We distinguish off-line and on-line re-sampling. *Off-line re-sampling* is used after the capturing of media streams with independent devices. An example is a concert which is captured with two independent audio and video devices. If these devices, like many real-world devices, have insufficient accurate crystal clocks, the theoretic playback duration according to the sample rate of the stored audio and video sequences may differ. Before the execution of the presentation, it is possible to re-sample them to the same theoretic playback duration. *On-line re-sampling* is used during a presentation in the case that, at run-time, a gap between media streams occurs.

Methods for re-sampling are to re-define the playback rate, to duplicate, to interpolate or to skip samples or to re-calculate the whole sequence. The human perception of the re-sampling depends strongly on the media. Video sequences can be re-sampled by adding or deleting single frames in a stream, as is done in NTSC/PAL conversions. If the output device supports different playback rates, the playback rate can be directly adjusted.

Audio streams are more complex. A user will be annoyed by simply duplicated or deleted blocks of audio. Also, changes in the playback rate can easily be noticed by the user, especially in the case of music playback because the frequency is changing. The same is true for simple interpolation of samples. Algorithms exist that can stretch or widen an audio sequence without this frequency change.

The *drop and add* algorithms are based on the idea of insertion or deletion of parts of audio blocks as mentioned above. Several techniques to reduce the user annoyance are known that have to be selected and parameterized depending on the material. One possibility in the case of lengthening is to use fading techniques to avoid knacks. The duration of the fading may not be too long because this leads to glitches. In the case of reduction of blocks by skipping parts, important parts of the audio hearing like attack phase of a base drum or snare get lost. Despite of its disadvantages the drop and add method is important because it may be performed in real-time and is usable for on-line re-sampling.

Algorithms that provide better results, e.g., by using transformations, exist but they do not support real-time demands and are only suitable for off-line re-sampling.

For inter-object synchronization, more detailed results of studies in the lip synchronization and pointer synchronization areas are described in the following section to make clear the importance of user perception aspects for presentation accuracy. A summary of requirements for other types of synchronization are included in Section V-A "Quality of Service."

A. Lip Synchronization Requirements

Lip synchronization refers to the temporal relationship between an audio and video stream for the particular case of humans speaking. The time difference between related audio and video LDU's is known as the *skew*. Streams which are perfectly "in sync" have no skew, i.e., 0 ms. Experiments at the IBM European Networking Center [69] measured skews that were perceived as "out of sync." In these experiments, users often mentioned that something was wrong with the synchronization, but this did not disturb their feeling for

the quality of the presentation. Therefore, the experimenters additionally evaluated the tolerance of the users by asking if the data out of sync affected the quality of the presentation. In discussions with experts that work with audio and video, the experimenters came to realize that generally, subjects responded to or remembered particular parts of the clips, therefore the experimenters observed a wide range of skews (up to 240 ms). To get accurate and good skew tolerance levels, the experimenters selected a speaker in a TV news environment in a head and shoulder shot. In this orientation, the viewer is not disturbed by background information and the viewer should be attracted by the gesture, eyes, and lip movement of the speaker.

The main results are:

- The "in sync" region spans a skew between −80 ms (audio behind video) and +80 ms (audio ahead of video). In this zone, most of the test candidates did not detect the synchronization error. Very few people said that if there was an error it affected their notion of the quality of the video. Additionally, some results indicated that the perfect "in sync" clip was classified "out of sync." The experimenter's conclusion is that lip synchronization can be tolerated within these limits.
- The "out of sync" areas span beyond a skew of −160 ms and +160 ms. Nearly everyone detected these errors and were dissatisfied with the clips. Data delivered with such a skew was in general not acceptable. Additionally, often a distraction occurred; the viewer/listener became more attracted by this "out of sync" effect than by the content itself.
- In the "transient" area where audio was ahead of video, the closer the speaker was, the easier errors were detected and described as disturbing. The same applied to the overall resolution, the better the resolution was, the more obvious the lip synchronization errors became.
- A second "transient" area, where video was ahead of audio, is characterized by a similar behavior as above as long as the skew values are near the in sync area. One interesting effect emerged, namely that video ahead of audio could be tolerated better than the opposing case. As above, the closer the speaker, the more obvious the skew.

B. Pointer Synchronization Requirements

In a computer-supported co-operative work (CSCW) environment, cameras and microphones are usually attached to the users' workstations. In the next experiment, the experimenters looked at a business report that contained some data with accompanying graphics [69]. All participants had a window with these graphics on their desktop where a shared pointer was used in the discussion. Using this pointer, speakers pointed out individual elements of the graphics which may have been relevant to the discussion taking place. This obviously required synchronization of the audio and remote telepointer.

Using the same judgement technique as in their first experiments, the "in sync" area related to audio ahead of pointing is 750 ms and for pointing ahead of audio it is 500 ms. This zone allows for a clear definition of the "in sync" behavior, regardless of the content.

The "out of sync" area spans a skew beyond −1000 ms and +1250 ms. At this point, the test candidates began to mention that the skew made the attempted synchronization worthless and became distracted unless the speaker slowed down or moved the pointer more slowly. From the user interface perspective, this is not acceptable. Quite clearly, the practice of pointing to one location on the technical figure while discussing another is virtually impossible.

In the "transient" area, the experimenters found that many test candidates noticed the "out of sync" effect but it was not mentioned as annoying. This is certainly different from "lip sync" where the user was more sensitive to the skew and, without question, found it annoying.

IV. A Reference Model for Multimedia Synchronization

A reference model is needed to understand the various requirements for multimedia synchronization, identify and structure run-time mechanisms that support the execution of the synchronization, identify interfaces between run-time mechanisms, and compare system solutions for multimedia synchronization systems.

To this end, we first describe existing classification and structuring methods. Then, a four-layer reference model is presented and used for the classification of multimedia synchronization systems in our case studies. As many multimedia synchronization mechanisms operate in a networked environment, we also discuss special synchronization issues in a distributed environment and their relation to the reference model.

A. Existing Classification Approaches

An overall classification was introduced by Little and Ghafoor [40]. They identified a physical level, system level and human level, but gave no detailed description or classification criteria. Other classification schemes distinguish between intrastream (fine-grain) synchronization and interstream (coarse-grain) synchronization, or between live and synthetic synchronization [40], [70].

The model of Gibbs, Breiteneder, and Tschichritzis [26] maps a synchronized multimedia object to an uninterpreted byte stream. The multimedia objects consist of derived media objects comprised of rearranged media sequences, e.g., scenes from a complete video. The parts of the media sequences are themselves part of an uninterpreted byte stream.

Ehley, Furth, and Ilyas [23] classify intermedia synchronization techniques that are used to control jitter between media streams according to the type and location of the synchronization control. They distinguish between a distributed control based on protocols, distribution based on servers and distribution on nodes without server structure. For local synchronization control, they distinguish control on several layers and the use of local servers.

These classification schemes seem to be orthogonal, and each one of them only captures some specific aspects. They do

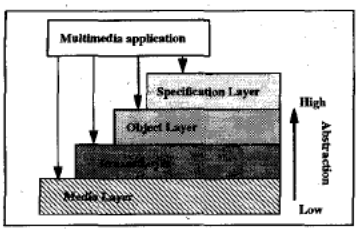

Fig. 17. Four-layer reference model.

not fulfill the above-stated requirements of the synchronization reference model.

An improved three-layer classification scheme has been proposed by Meyer, Effelsberg, and Steinmetz [51]. The layers are: the media layer for intrastream synchronization of time-dependent media, the stream layer for interstream synchronization of media streams, the object layer for the presentation, including the presentation of time-independent media objects and the specification of complex multistream multimedia presentations. At each layer, typical objects and operations are identified. Each layer can be accessed directly by the application or indirectly through higher layers. This approach fulfills the demands of a reference model approach and we will enhance and interpret it appropriately in the following.

B. The Synchronization Reference Model

A four-layer synchronization reference model is shown in Fig. 17. Each layer implements synchronization mechanisms which are provided by an appropriate interface. These interfaces can be used to specify and/or enforce the temporal relationships. Each interface defines services, i.e., offering the user a means to define his/her requirements. Each interface can be used by an application directly, or by the next higher layer to implement an interface. Higher layers offer higher programming and quality of service (QoS) abstractions.

For each layer, typical objects and operations on these objects are described in the following sections. The semantics of the objects and operations are the main criteria for assigning them to one of the layers.

Detailed programming examples derived from a real interface provided by a real product, prototype or standard demonstrate how synchronization can be achieved through each layer. The scenario for the programming example is to display subtitles at predefined times during the playout of a digital movie.

1) Media Layer: At the *media layer,* an application operates on a single continuous media stream, which is treated as a sequence of LDU's.

The abstraction offered at this layer is a device-independent interface with operations like *read(devicehandle, LDU)* and *write(device-handle, LDU).* Systems such as ActionMedia/II audio-video kernel [34] or SunSPARC audio device [71] provide the corresponding interfaces.

To set up a continuous media stream using the abstractions offered by the media layer, an application executes a process for each stream in the manner shown in the following example:

```
window = open
  (''Videodevice'');  // Create a video output window
movie open
  (''File'');         // Open the video file
while (not
 eof(movie)) {        // Loop
  read(movie, &ldu); // Read LDU
  if (ldu.time=20)    // Start the presentation
  print
  (''Subtitle 1'');  // of synchronized subtitles
  else if
  (ldu.time =26)
  print(''Subtitle 2'');
  write(window, ldu);} // Present LDU
close(window);        // Close window
close(movie);         // Close file
```

The process reads and writes LDU's in a loop as long as data are available. Synchronous playout of a subtitle is achieved by polling the timestamps of the LDU's for a certain value.

Using this layer, the application itself is responsible for the intrastream synchronization by using flow-control mechanisms between a producing and a consuming device [62]. If multiple streams run in parallel, the sharing of resources may affect their real-time requirements. Usually, a resource reservation and management scheme allows for guaranteeing intrastream synchronization [73]. The operating system schedules the corresponding process in real-time [56]. In distributed systems, the networking components are taken into account [2], [25]. In the special case of lip synchronization, the interstream synchronization can be provided easily, where simultaneous audio and video frames are interleaved within the same LDU (e.g., ActionMedia-II's audio-video support system [34] and the MPEG data stream [38]). Finally, the synchronous playback of time-independent media objects and user interactions are tasks to be performed by the application.

Media layer implementations can be classified into simple implementations and implementations that provide access to interleaved media streams.

2) Stream Layer: The *stream layer* operates on continuous media streams, as well as on groups of media streams. In a group, all streams are presented in parallel by using mechanisms for interstream synchronization.

The abstraction offered by the stream layer is the notion of streams with timing parameters concerning the QoS for intrastream synchronization in a stream and interstream synchronization between streams of a group.

Continuous media is seen on top of the stream layer as a data flow with implicit time constraints; individual LDU's are not visible. The streams are executed in a real-time environment (RTE), where all processing is constrained by well-defined time specifications [29]. On the other hand, the applications themselves that are using the stream layer services are executed in a non real-time environment (NRTE), where the processing of events is controlled by the operating system scheduling policies.

Typical operations invoked by an application to manage streams and groups from the NRTE are: *start(stream), stop(stream), create_group(list_of_streams), start(group)* and *stop(group).* The interaction with time-independent media objects and user interactions is performed by the attachment

of events to the continuous media streams (e.g., *setcuepoint(stream/group, at, event)*). Such an event is sent to the application whenever the stream reaches the specified point during playback. Using this layer, the application is furthermore in charge of any time-independent media object and user interaction processing. This leads to different application interfaces for continuous media and for time-independent media and user interactions.

The Sync/Stream Subsystem of IBM's MultiMedia Presentation Manager (MMPM) for OS/2 provides a set of services which can be used to implement data streaming and synchronization. This subsystem, which can be understood as the RTE, is comprised of the Sync/Stream Manager and several stream handlers [35]. Stream handlers are responsible for controlling the continuous data flow in real-time. The Sync/Stream Manager provides a resource management and controls the registration and activities of all stream handlers.

The following programming example for the use of the stream layer uses the string command interface provided by MMPM:

```
open digitalvideo
 alias ex            // Create video descriptor
load ex video.avs    // Assign file to video descriptor
setcuepoint ex
 at 20 return 1      // Define event 1 for subtitle 1
setcuepoint ex
 at 26 return 2      //Define event 2 for subtitle 2
setcuepoint ex on    // Activate cuepoint events
play ex              // Start playing
switch readevent() {// Event handling
  case 1:display
  (''Subtitle 1'')   // If event 1 show subtitle 1
  case 2:display
  (''Subtitle 2'')   // If event 2 show subtitle 2
 }
```

In MMPM/2, interstream synchronization for synchronized playback of multiple streams within a group is achieved by a master/slave algorithm, where one stream (the master) controls the behavior of one or more subordinate streams (the slaves). The skip/pause algorithm introduced in [6] gives a detailed discussion of the implementation of such a behavior. The synchronization mechanism in ACME [5], as well as the Orchestration Service [18], support stream layer abstractions for distributed multimedia systems.

The stream layer abstraction was derived from the abstraction normally provided by the integration of analog media in the computer system. In the Muse and Pygmalion systems of MIT's Project Athena [32] or in the DiME [70] system, continuous media were routed over separated channels through the computer. The connected devices could be controlled by sending commands via the RS-232C interface to start and stop the media streams. In such systems, live synchronization between various continuous media streams is directly performed by the dedicated processing devices.

An application using the stream layer is responsible for starting, stopping and grouping the streams and for the definition of the required QoS in terms of timing parameters supported by the stream layer. It is also responsible for the synchronization with time-independent media objects.

Stream layer implementations can be classified according to their support for distribution, the types of guarantees that they provide and the types of supported streams (analog and/or digital).

3) Object Layer: The *object layer* operates on all types of media and hides the differences between time-independent and time-dependent media.

The abstraction offered to the application is that of a complete, synchronized presentation. This layer takes a synchronization specification as input and is responsible for the correct schedule of the overall presentation. From our understanding, the abstractions are similar to the "object model" presented in [67].

The task of this layer is to close the gap between the needs for the execution of a synchronized presentation and the stream-oriented services. The functions located at the object layer are to compute and execute complete presentation schedules that include the presentation of the noncontinuous media objects and the calls to the stream layer. Furthermore, the object layer is responsible for initiating preparative actions that are necessary for achieving a correctly synchronized presentation. The object layer does not handle the interstream and intrastream synchronization. For these purposes, it uses the services of the stream layer.

An example of interfacing to this layer is an MHEG specification [52]. The scope of the MHEG standard is the coded representation of final form multimedia and hypermedia information objects. In the following, we give a rudimentary example of how our scenario might be coded in the MHEG standard (using a simple notation to demonstrate the essentials of our reference model):

```
Composite {                  //Composite object
start-up link                //How to start presentation
viewer start-up
viewer-list                  //Virtual views on
  Viewer1:                   //component objects
   reference to
   Component1
  Viewer2:
   reference to
   Component2
  Viewer3:
   reference to
   Component3
Component1                   //Component objects
   reference to content      //of the composite
  ''movie.avs''
Component2
  reference to content
  ''Subtitle1''
Component3
  reference to content
  ''Subtitle2''
Link1                        //Temporal relations
  ''when timestone status
  of Viewer1 becomes 20
  then start Viewer2''
Link2
  ''when timestone status
  of Viewer1 becomes 26
  then start Viewer3''
}
```

A possible implementation of the object layer is an MHEG run-time system, the *MHEG engine*. The MHEG engine evaluates the status of the objects and performs operations (actions) like prepare, run, stop or destroy on these objects. In the case of time-dependent media objects, the run operation may be

mapped to the initiation of a media stream on the stream layer. In the case of a time-independent media object, this call directly demands the object to be presented. Prepare times are necessary, for example, to allow the stream layer to build up a stream connection, or in the case of time-independent media objects, to prefetch the presentation, e.g., to adapt the picture color maps to the maps of the output device. The preparation is started by the prepare action.

Object layer implementations can be classified according to distribution capabilities and the type of presentation schedule computation. It can be determined whether the implementation calculates a schedule and, if it calculates one, whether the schedule is computed before the presentation (compile-time computation) or at run-time of the presentation (run-time computation). Concerning distribution, implementations may be local and may support distribution based on a server structure or full distribution without restriction.

The task of the application using the object layer is to provide a synchronization specification.

4) Specification Layer: The *specification layer* is an open *layer*. It does not offer an explicit interface. This layer contains applications and tools that allow one to create synchronization specifications. Examples of such tools are synchronization editors, multimedia document editors and authoring systems. Also located at the specification layer are tools for converting specifications to an object layer format. An example of such a conversion tool is a multimedia document formatter that produces an MHEG specification as proposed by Markey [50].

For example, the synchronization editor of the MODE system [8] may be used to specify the synchronization example. It offers a graphical interface to select the video and text objects to use, to preview the video, to select suitable points where the subtitles have to be shown, to specify the temporal relations of these points to the subtitle and to store the synchronization specification.

The specification layer is also responsible for mapping QoS requirements of the user level to the qualities offered at the object layer interface.

Synchronization specification methods can be classified into the following main categories:

- Interval-based specifications, which allow the specification of temporal relations between the time intervals of the presentations of media objects.
- Axes-based specifications, which relate presentation events to axes that are shared by the objects of the presentation.
- Control flow-based specifications, in which at given synchronization points, the flow of the presentations is synchronized.
- Event-based specifications, in which events in the presentation of media trigger presentation actions.

C. Synchronization in a Distributed Environment

Synchronization in a distributed environment is more complex than in a local environment. This is mainly caused by the distributed storage of synchronization information and the different locations of the media objects involved in the presentation. The communication between the storage and presentation site introduces additional delays and jitter. Often, we also encounter multi-party communication patterns.

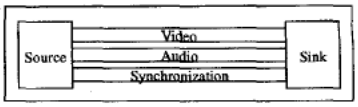

Fig. 18. Use of a separate synchronization channel.

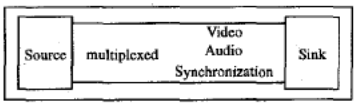

Fig. 19. Multiplexed media and synchronization channels.

1) Transport of the Synchronization Specification: At the sink node, the presentation component needs to have the synchronization specification at the moment an object is to be displayed. We distinguish between three main approaches for the delivery of the synchronization information to the sink:

- Delivery of the complete synchronization information before the start of the presentation: This approach is often used in the case of synthetic synchronization. Typically, the application at the sink node accesses the object layer interface with the specification or a reference to the specification as a parameter. The implementation of this approach is simple and it also allows easy handling in the case of several source nodes for the media objects. The disadvantage is the delay caused by the transport of the synchronization specification before the presentation, especially if it is stored on another node. The transport of the synchronization specification is a duty of a component located at the object layer or above.
- Use of an additional synchronization channel: This approach, shown in Fig. 18, is mainly useful in the case of one source node. It is used and is preferable in the case of live synchronization when all the synchronization information is not known in advance. No additional delays are caused by this method. A disadvantage is that an additional communication channel is needed that may cause errors due to delay or loss of synchronization specification units. It is often forgotten that the information on the synchronization channel must be decoded at the time the respective object is to be displayed, i.e., data communication at this channel must obey certain time behavior. Also, the case of multiple source nodes for synchronized media objects is difficult to handle. The synchronization channel must be handled by the object layer and possibly supported by the stream layer if the synchronization channel is to be defined as a stream.
- Multiplexed data streams: The advantage of multiplexing data streams on one communication channel (Fig. 19) is that the related synchronization information is delivered together with the media units. No additional synchronization channel is necessary and no additional delay is caused by this approach. An important problem regarding multiplexed media and synchronization information is the

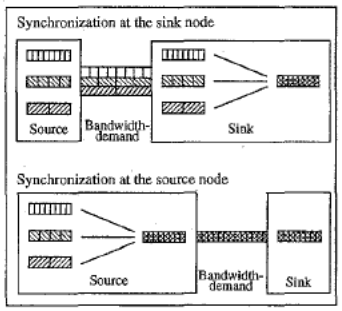

Fig. 20. Combining objects to reduce communication resource demands.

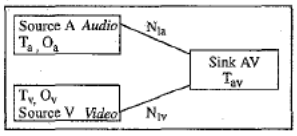

Fig. 21. Clock offsets in a distributed environment.

difficulty of selecting an appropriate QoS which matches the requirements of all involved medias, e.g., reliability is dominated by the most stringent media objects. This method is also difficult to use for multiple source nodes. It must be supported by the stream layer. The use of multiplexed data streams may be implied by coding standards like MPEG. MPEG defines a bitstream that combines video, audio and the related synchronization information. Hence, this type of bitstream can be regarded as one medium on the stream layer and for the synchronization with other media, the other approaches can also be chosen.

2) Location of Synchronization Operations: In some cases it is possible to synchronize media objects by combining the objects into a new media object. This approach may be used to reduce communication resource demands, as shown in Fig. 20. In this case, an animation and two bitmaps that must overlay a video sequence are already merged at the source node to become a new video object to reduce bandwidth demands.

The mixing of objects, including time-independent media objects, must be supported by the object layer. The mixing of media streams, like mixing audio channels, must be supported by the stream layer.

3) Clock Synchronization: In distributed systems, the synchronization accuracy between the clocks of the source and sink nodes must be considered. Many synchronization schemes demand knowledge about the timing relations. This knowledge is the basis for global timer-based synchronization schemes, as well as for schemes that demand that operations on distributed nodes are timely and coordinated to ensure, on one hand, in-time delivery and on the other hand, that operations are not performed too early to avoid a buffer overflow.

This problem is especially important for the synchronization in the case of multiple sources (Fig. 21). If a synchronized audio-video presentation should start at time T_{av} at the sink node, the audio transmission of Source A must start at $T_a = T_{av} - N_{la} - O_a$, with N_{la} as the known net delay and O_a as the offset of the clock of node A with respect to the clock of the sink node. For source node B, the start time of the video transmission is $T_v = T_{av} - N_{lv} - O_v$.

The offsets O_a and O_v are not known. The resulting problem of delivery to the sink in time can be solved if the maximal possible values for O_a and O_v are known. It is possible to allocate buffer capacities at the sink and to start the transmission of the audio and video in advance to guarantee that the required media units are available. Because the necessary buffer capacity at the sink node depends on the possible offset, and we must assume limited buffer capacity, it is necessary to limit the maximal offset. This can be achieved with clock synchronization protocols like the Network Time Protocol [54] that allows the synchronization of the clocks with an accuracy in the range of 10 ms. With the use of public broadcast timer signals, submillisecond accuracies are practical [55]. This accuracy is suitable for global timer synchronization and for distributed operation scheduling.

The in-time delivery of LDU's of a stream is a task of the stream layer that must handle the clock offsets.The object layer is responsible for in-time delivery of time-independent media objects.

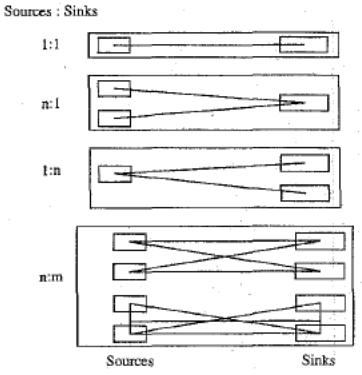

Fig. 22. Multiple communication relations.

4) Multiple Communication Relations: Possible communication patterns are shown in Fig. 22. Patterns with multiple sinks demand that at run-time, multicast and broadcast mechanisms be used to reduce resource requirements, in particular network resources. Also inefficient multiple executions of the same operation at different sinks should be avoided. The multicasting of streams is the task of the stream layer. Efficient planning of operation execution in the different communication patterns is a responsibility of the object layer.

5) Multi-Step Synchronization: Synchronization in a distributed environment is typically a multi-step process. During all steps of the process, the synchronization must be maintained in a way that enables the sink to perform the final synchronization. The steps of the process are:

- Synchronization during object acquisition, e.g., during digitizing video frames.
- Synchronization of retrieval, e.g., synchronized access to frames of a stored video.
- Synchronization during delivery of the LDU's to the network, e.g., delivering the frames of a video to the transport service interface.

TABLE II
OVERVIEW OF THE SYNCHRONIZATION REFERENCE MODEL LAYERS

Layer	Interface Abstraction	Tasks
Specification	• The tools performing the tasks of this layer have interfaces, the layer itself has no upper interface	• Editing • Formatting • Mapping user-oriented QoS to the QoS abstraction at the object layer
Object	• Synchronization Specification • Objects that hide types of enclosed media • Media-oriented QoS (In terms of acceptable skew and acceptable jitter)	• Plan and coordinate presentation scheduling • Initiate presentation of time-dependent media objects by the stream layer • Initiate presentation of time-independent media objects • Initiate presentation preparation actions
Stream	• Streams and groups of streams. • Guarantees for intrastream synchronization • Guarantees for interstream synchronization of streams in a group	• Resource reservation and scheduling of LDU processing
Media	• Device independent access to LDUs • Guarantees for single LDU processing	• File and device access

TABLE III
CLASSIFICATION OF METHODS AND MECHANISMS AT THE SYNCHRONIZATION REFERENCE MODEL LAYERS

Layer	Classification items
Specification	Synchronization specification method • Interval-based synchronization • Axes-based synchronization • Control flow-based synchronization • Event-based synchronization
	Type of tool • Textual specification tool • Graphical specification tool • Converter
Object	Type of distribution • Local • Distributed, based on servers • Distributed without server usage
	Type of schedule computation • No computation • Compile-time computation • Run-time computation
Stream	Type of distribution • Local • Distributed
	Type of guarantees for stream QoS • No guarantees for QoS, best effort • Guarantees for QoS by resource reservations
Media	Type of accessible data • Single medium data • Interleaved, complex data

- Synchronization during the transport, e.g., by isochronous protocols.
- Synchronization at the sink, i.e., synchronized delivery to the output devices.
- Synchronization within the output device.

6) Manipulation of the Presentation: The support of functions like pause, forward and backward with different presentation speeds, direct access, stop and repeat is difficult in a distributed environment. The necessary information must be distributed in the environment. Objects that have already been prepared in advance for the presentation must be deleted. Network connections may be subject to change or must be rebuilt. Therefore, delays in the execution of these manipulation functions are difficult to avoid.

7) Consequences for Synchronization in a Distributed Environment: To achieve synchronization in a distributed environment, many decisions must be made. The first decision is the selection of the type of transport for the synchronization specification. In run-time, decisions must be taken concerning the location of the synchronization operations, handling of the offsets of the clocks and the handling of multicast and broadcast mechanisms. In particular, coherent planning of the steps in the synchronization process, together with the necessary operations on the objects, e.g., decompression, must be done. In addition, presentation manipulation operations demand additional re-planning at run-time.

In general, the execution of synchronized distributed presentations is a complex planning problem. The resulting plan is often known as a schedule.

D. Aggregate Characteristics of the Synchronization Reference Model

The reference model allows for the structuring and classifying of synchronization systems. The identification of the interfaces and layers enables one to combine existing solutions into complete systems. Table II provides an overview of the interface abstractions and tasks of all layers of our reference model.

The classification of mechanisms and methods in the layers is summarized in Table III.

V. SYNCHRONIZATION SPECIFICATION

The synchronization specification of a multimedia object describes all *temporal dependencies* of the included objects in the multimedia object. It is produced using tools at the specification layer and is used at the interface to the object layer. Because the synchronization specification determines the whole presentation, it is a central issue in multimedia systems. In the following, requirements for synchronization specifications are described and specification methods are described and evaluated.

A synchronization specification should be comprised of:

- Intra-object synchronization specifications for the media objects of the presentation.
- QoS descriptions for intra-object synchronization.
- Inter-object synchronization specifications for media objects of the presentation.
- QoS descriptions for inter-object synchronization.

The synchronization specification is part of the description of a multimedia object. In addition, for a multimedia object, it may be described in which presentation form, respectively in which alternative presentation forms, a media object should be presented. For example, a text could be presented as text on the screen or as a generated audio sequence. A specification may allow only one of these or a selection of the presentation form at run-time.

TABLE IV
SOME QoS FOR THE PRESENTATION OF A MEDIA OBJECT

Media	Image (e.g. bitmap)	Video	Audio
Quality of Service	Color Depth	Color Depth	Lin. or log. sampling
	Resolution	Resolution	Sample Size
		Frame Rate	Sample Rate
		Jitter	Jitter
		Error Rate	Error Rate

In the case of live synchronization, the temporal relations are implicitly defined during capturing. QoS requirements for single media are defined before starting the capture.

In the case of synthetic synchronization, the specification must be created explicitly. Several synthetic synchronization specification methods have been described in the literature. The most important are classified, surveyed and evaluated in the following sections.

A. Quality of Service

Synchronization requirements can be expressed by a QoS specification. The necessary QoS depends on the media and application.

1) Quality of Service for a Media Object: The QoS specification for a media object includes the quality concerning single LDU's of a media object and the accuracy with which the temporal relations between the LDU's of this media object must be fulfilled if the media object is a time-dependent object.

Table IV shows some QoS parameters for a media object. The white boxes contain qualities that are independent of temporal relations. The light shaded boxes contain timing related qualities that are under the limited influence of the presentation system because the quality depends on the quality selected during capture. Usually, only quality degradation via the presentation system is possible. The dark shaded boxes contain qualities which are potentially under full control of the presentation environment.

2) Quality of Service of Two Related Media Objects: One QoS parameter can define the acceptable skew within the concerned data streams; namely, it defines the affordable synchronization boundaries. The notion of QoS is well established in communication systems. In the context of multimedia, it also applies to local systems.

If audio and video parts of a film are stored as different entries in a database, lip synchronization according to the above-mentioned results should be taken into account. In this context we want to introduce the notion of *presentation-and production-level synchronization*:

- Production-level synchronization refers to the QoS to be guaranteed prior to the presentation of the data at the user interface. It typically involves the recording of synchronized data for subsequent playback. The stored data should be captured and recorded with no skew at all, i.e., "in sync." This is in particular applicable if the file is stored in an interleaved format. At the participant's site, the actual incoming audiovisual data is "in sync" according to the defined lip synchronization boundaries. Assuming the data arrive with a skew of +80 ms, and if audio and video LDU's are transmitted as a single multiplexed stream over the same transport connection, then it will be displayed as apparently "in-sync." Should the data be stored on the hard disk and presented simultaneously at a local workstation and to a remote spectator, then for correct delivery, the QoS should be specified as being between −160 ms and 0 ms. At the remote viewer's station without this additional knowledge of the actual skew the outcome might be that by applying these boundaries twice, data are not "in sync." In general, any synchronized data which will be further processed should be synchronized according to a production-level quality, i.e., with no skew at all.
- The presentation requirements discussed in Section III identify presentation-level synchronization. This synchronization defines whatever is reasonable at the user interface. It does not take into account any further processing of the synchronized data; presentation-level synchronization focuses on the human perception of the synchronization. As shown in the above paragraph, by recording the actual skew as part of the control information, the required QoS for synchronization can be easily computed. The QoS values shown in Table V relate to presentation-level synchronization. Most of them result from exhaustive experiments and experiences, others are derived from literature as referenced in [69]. They can serve as a general guideline for any QoS specification. During the lip and pointer synchronization experiments (Section III), we learned that many factors influenced these results. We understand this whole set of QoS parameters as a first-order result to serve as a general guideline. However, these values may be relaxed depending on the actual content.

TABLE V
QUALITY OF SERVICE FOR SYNCHRONIZATION PURPOSES

Media		Mode, Application	quality of service
video	animation	correlated	+/- 120 ms
	audio	lip synchronization	+/- 80 ms
	image	overlay	+/- 240 ms
		non-overlay	+/- 500 ms
	text	overlay	+/- 240 ms
		non-overlay	+/- 500 ms
audio	animation	event correlation (e.g. dancing)	+/- 80 ms
	audio	tightly coupled (stereo)	+/- 11 µs
		loosely coupled (dialogue mode with various participants)	+/- 120 ms
		loosely coupled (e.g. background music)	+/- 500 ms
	image	tightly coupled (e.g. music with notes)	+/- 5 ms
		loosely coupled (e.g. slide show)	+/- 500 ms
	text	text annotation	+/- 240 ms
	pointer	audio related to the item to which the pointer shows	- 500 ms, + 750 ms

3) Quality of Service of Multiple Related Media Objects: So far, media synchronization has been evaluated as the relationship between two kinds of media or separate data streams. This is the canonical foundation of all types of media synchronization. In practice, we often encounter more than two related media streams; a sophisticated multimedia application

scenario incorporates the simultaneous handling of various streams. An example is a video conference where a window displays the actual speaker and the audio emerges from an attached pair of speakers.

Video and audio data are related by lip synchronization demands. Audio and the telepointer are related by the pointer synchronization demands. The relationship of video data and the telepointer is then yielded by a simple combination. In this example, we will define the following skews [69]:

```
max skew (video ahead_of audio) = 80 ms
max skew (audio ahead_of video) = 80 ms
max skew (audio ahead_of pointer) = 740 ms
max skew (pointer ahead_of audio) = 500 ms
```

leading to the skew

```
skew (video ahead_of pointer) = 820 ms
skew (pointer ahead_of video) = 580 ms
```

In general, these requirements can be derived easily by the accumulation of the canonical skew as shown in the above example. The information gathered by the aggregation of media is of interest for the user, as well as for the multimedia system which must provide service according to these values.

In some cases, too many specifications of a synchronization skew exist; for example, a language lesson that includes audio data in English and Spanish, as well as the related video sequences. The course builder enforces lip synchronization between video and audio regardless of the language (±80 ms). Additionally, the sentences need to be synchronized to switch from one language to the other (we chose a figure of 400 ms for this case). As lip synchronization is more demanding than the synchronization between the languages, this would lead to the following skew specification:

```
1. max skew (video ahead_of audio_english) = 80 ms
2. max skew (audio_english ahead_of video) = 80 ms
3. max skew (video ahead_of audio_spanish) = 80 ms
4. max skew (audio_spanish ahead_of video) = 80 ms
5. max skew (audio_english ahead_of audio_spanish) =
    400 ms
6. max skew (audio_spanish ahead_of audio_english) =
    400 ms
```

This specification consists of a set of related requirements which all need to be fulfilled, i.e., we must find "the greatest common denominator." For each canonical form, the derived skews are computed as follows:

```
1+2+3+4:
max skew (audio_english ahead_of audio_spanish) =
    160 ms
max skew (audio_spanish ahead_of audio_english) =
    160 ms

1+2+5+6:
max skew (video ahead_of audio_spanish) = 480 ms
max skew (audio_spanish ahead_of video) = 480 ms

3+4+5+6:
max skew (video ahead_of audio_english) = 480 ms
max skew (audio_english ahead_of video) = 480 ms
```

In the second step, the most stringent set of all requirements are selected:

```
1. max skew (video ahead_of audio_english) = 80 ms
2. max skew (audio_english ahead_of video) = 80 ms
3. max skew (video ahead_of audio_spanish) = 80 ms
4. max skew (audio_spanish ahead_of video) = 80 ms
5. max skew (audio_english ahead_of audio_spanish) =
    160 ms
6. max skew (audio_spanish ahead_of audio_english) =
    160 ms
```

In the following step, any set of synchronization requirements can be chosen from the above-derived calculations:

```
max skew (video ahead_of audio_english) = 80 ms
max skew (audio_english ahead_of video) = 80 ms
max skew (audio_english ahead_of audio_spanish) =
    160 ms
max skew (audio_spanish ahead_of audio_english) =
    160 ms
```

In summary, the above procedures allow us to solve two related problems:

- If the applications impose a set of related synchronization requirements on a multimedia system, we are now able to find the most stringent demands.
- If a set of individual synchronization requirements between various data streams is provided, we are now able to compute the required relationships between each individual pair of streams.

B. Multimedia Synchronization Specification Methods

For the complex specification of multiple object synchronization, including user interaction, sophisticated specification methods must be used. The following requirements should be fulfilled by such a specification method:

- The method shall support object consistency and maintenance of synchronization specifications. Media objects should be kept as one logical unit in the specification.
- The method should supply an abstraction of the contents of a media object that on one hand allows the specification of temporal relations that refer to a part of the media object, and on the other hand regards the media object as one logical unit.
- All types of synchronization relations should be easily described.
- The integration of time-dependent and time-independent, media objects must be supported.
- The definition of QoS requirements must be supported by the specification method. It should preferably be expressed in the method directly.
- Hierarchical levels of synchronization must be supported to enable the handling of large and complex synchronization scenarios.

In the following sections, specification methods are assessed according to the criteria described above.

C. Interval-Based Specifications

In the interval-based synchronization specification, the presentation duration of an object is called an interval. Two time intervals may be synchronized in 13 different types [3], [28]. Some of these types are invertible like "before" and "after". Fig. 23 shows a reduced set of seven noninvertible types according to [39]. A simple synchronization specification method for two media objects is to use these seven types.

The enhanced interval-based model [74] is based on interval relations. The basic interval relations have already been shown in Fig. 23. In the enhanced approach, 29 interval relations that are defined as disjunctions of the basic interval relations have been identified as relevant for multimedia presentations. To simplify the synchronization specification, ten operators have

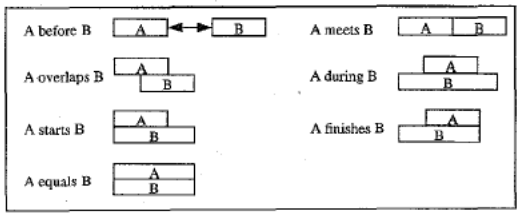

Fig. 23. Types of temporal relations between two objects.

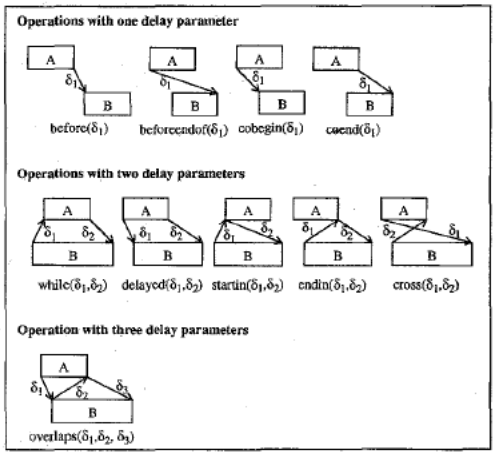

Fig. 24. Operations in the enhanced interval-based method.

been defined that can handle these interval relations. These operators are shown in Fig. 24. The duration of a presentation like A or B, as well as the delay δ_i, are subsets of $\Re_0^+$ because the duration of a presentation, as well as of a delay, may not be known in advance. In addition, for the operations *beforeendof, delayed, startin, endin, cross* and *overlaps* δ_i must not be $\{0\}$.

A slide show with slides Slide_i $(1 \leq i \leq n)$ and an audio object *Audio* can be specified in this model by:

```
Slide1 cobegin(0) Audio
Slidei before(0) Slidei+1 (1 ≤ i ≤ n − 1)
```

Lip synchronization between an audio object *Audio* and a video object *Video* is simply specified by:

```
Audio while(0,0) Video
```

The application example of Fig. 13 can be sketched as follows:

```
Audio1 while(0,0) Video
Audio1 before(0) RecordedInteraction
RecordedInteraction before(0) B1
P1 before(0) P2
P2 before(0) P3
P3 before(0) Interaction
P3 before(0) Animation
Animation while(2,5) Audio2
Interaction before(0) P4
```

This model allows the definition of a duration for time-dependent and time-independent media objects. This duration is used in the example to specify the duration of the presentation of the objects picture 1 to picture 3. The open duration of the user interaction can be specified by defining the duration as $\{\Re_0^+\}$.

The advantage of this model is that it is easy to handle open LDU's, and therefore user interaction. It is possible to specify additional indeterministic temporal relations by defining intervals for durations and delays. Disjunction of operators can be used for specifications of presentation relations like not parallel. Therefore, it is a very flexible model that allows the specification of presentations with many run-time presentation variations.

TABLE VI
ASSESSMENT OF THE ENHANCED INTERVAL-BASED SYNCHRONIZATION SPECIFICATION

Advantages	Disadvantages
Logical objects can be kept	Complex specification
Good abstraction for media content	Additional specifications for skew QoS necessary
Easy integration of time-independent objects	Direct specification of time relations between media objects, but not for subunits of the media objects
Easy integration of interactive objects	Resolving of indeterminism at runtime may lead to inconsistencies
Specification of indeterministic temporal relations supported	

The model does not include skew specifications. Despite the direct specification of time relations between media objects, it does not allow the specification of temporal relations directly between subunits of objects. Such relations must be defined indirectly by delay specifications, as shown in the while operation for the animation and audio in the application example, or by splitting the objects. The flexibility of specifiable presentations may lead to inconsistencies in run-time. For example, for two video objects A and B a *not* parallel relation has been defined. In run-time, A may be running and B may be coupled by a *before(0)* relation to the end of a user interaction. If this user interaction ends, video B must be started, but on the other hand, it may not be started because of the *not* parallel relation. It must be defined in the model how such inconsistencies must be handled in run-time or such potential inconsistencies must be detected before run-time and the specification must be rejected. Building of hierarchies is easily definable. The assessment of the enhanced interval-based method is summarized in Table VI.

D. Axes-Based Synchronization

In an axes-based specification, the presentation events, like the start and end of a presentation, are mapped to axes that are shared by the objects of the presentation.

1) Synchronization Based on a Global Timer: For synchronization based on a *global timer*, all single-medium objects are attached to a time axis that represents an abstraction of real-time. This specification method is used, for example, in the Athena Muse project [32], where synchronization is described by attaching all objects, independently of each other, to a time axis. Removing one object does not affect the synchronization of the other objects.

With modifications, this kind of specification is also used in the model of active media [72]. A world time is maintained, which is accessible to all objects. Each object can map this world time to its local time and moves along its local time axis. When the difference between world time and local time exceeds a given limit, resynchronization with world time is required. A time axis mechanism is also used in QuickTime [22].

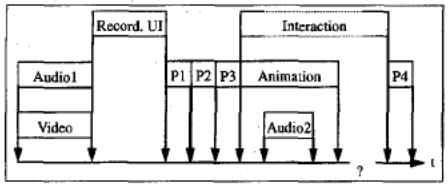

Fig. 25. Time axes synchronization specification example.

TABLE VII
ASSESSMENT OF THE TIME AXES SYNCHRONIZATION SPECIFICATION

Advantage	Disadvantages
Easy to understand	Objects of unknown duration cannot be integrated, extensions to the model are required
Support of hierarchies easy to realize	Skew QoS has to be specified indirectly by using the common time axis or additional QoS specifications have to be given

Advantage	Disadvantages
Easy to maintain because of the mutual independence of objects	
Good abstraction for media contents	
Integration of time-independent objects is easy	

Synchronizing objects by means of a time axis allows a very good abstraction from the internal structure of single-medium objects and nested multimedia objects. Defining the beginning of a subtitle presentation relative to a scene in a video stream requires no knowledge of the related video frames. Since synchronization can only be defined based on fixed points of time, problems arise if objects include LDU's of unpredictable duration.

Moreover, synchronization based on one common global timer may not be sufficient for expressing the synchronization relations between different presentation streams. Depending on the coherence of these presentation streams, synchronization based on a common time axis might be either too strong or too weak. A possible solution is to define an additional QoS for each pair of media streams.

The use of the global timer demands that the media streams are able to synchronize themselves to the global timer. This may be difficult for audio streams because of the re-sampling problems. Therefore, the audio stream is often used as the global timer, but this still causes difficulties if several audio streams must be synchronized.

Fig. 25 shows the specification of the application example. It can be seen that there is no natural solution to handle the unpredictable duration of a user interaction.

The assessment of the time axis method is summarized in Table VII.

2) Synchronization Based on Virtual Axes: Virtual time axes, as used in the project Athena [32] or the HyTime standard [36], are a generalization of the time axis approach. In this specification method, it is possible to specify coordinate systems with user-defined measurement units. A synchronization specification is executed according to these axes. It is also possible to use several virtual axes to create a virtual coordinate space. An example is a music description by notes as shown

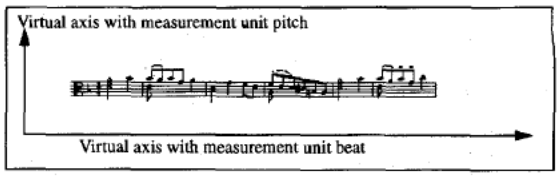

Fig. 26. Musical notes as an example of virtual axes.

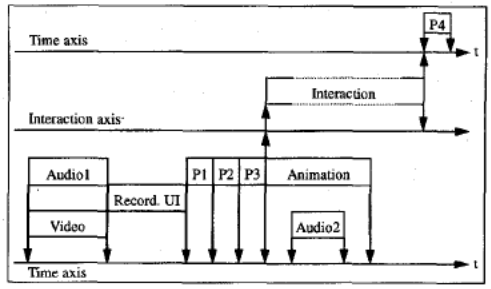

Fig. 27. Virtual time axes specification example.

TABLE VIII
ASSESSMENT OF THE VIRTUAL AXES SYNCHRONIZATION SPECIFICATION

Advantages	Disadvantages
Easy to understand	Skew QoS only indirectly or through additional specifications defined
Often specification according to the problem space possible	Specification may become complex with many axes
Good possibility for building hierarchies	Mapping of axes at runtime may be complex and time consuming
Easy to maintain, because objects are kept as units and mutually independent objects	
Good abstraction for media content	
Easy integration of time-independent media objects	
Interactive objects can be included using specialized axes.	

in Fig. 26. The tune frequency is defined by the position on the note lines. The sequence and duration is defined on the axis with the measurement unit beat.

The mapping of the virtual axes to real axes is done in run-time. In the example shown in Fig. 26, the pitch axis is mapped to the audio frequency and the beat axis is mapped to a timer.

The application example of Fig. 13 can be realized in this approach by two time axes and an interaction axis (Fig. 27). The latter should have interaction events as measurements units.

The assessment of the virtual axes method is summarized in Table VIII.

E. Control Flow-Based Specification

In *control flow-based specifications*, the flow of the concurrent presentation threads is synchronized in predefined points of the presentation.

1) Basic Hierarchical Specification: Hierarchical synchronization descriptions [1], [63] are based on two main synchronization operations: *serial synchronization* of actions and

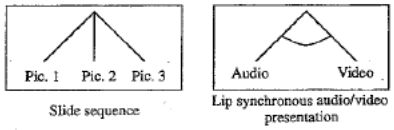

Fig. 28. Serial and parallel presentations.

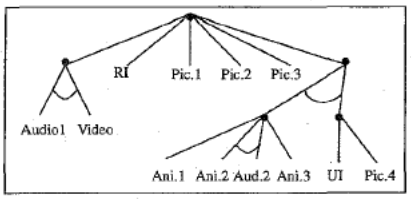

Fig. 29. Hierarchical specification example (RI = Recorded Interaction, Pic. = Picture, Aud. = Audio, Ani. = Animation, and UI = User Interaction).

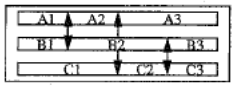

Fig. 30. Non-describable synchronization.

parallel synchronization of actions (Fig. 28). In a hierarchical synchronization specification, multimedia objects are regarded as a tree consisting of nodes which denote serial or parallel presentation of the outgoing subtrees.

An action can be either atomic or compound. An atomic action handles the presentation of either a single-media object, user input or delay. Compound actions are a combination of synchronization operators and atomic actions.

The introduction of a delay as a possible action [39] allows the modeling of further synchronization behavior like delays in serial presentations and delayed presentations of objects in a parallel synchronization.

Hierarchical structures are easy to handle and widely used. Restrictions from the hierarchical structure arise from the fact that each action can only be synchronized at its beginning or end. This means, for example, that the presentation of subtitles at parts of a video stream requires the video stream to be split into several consecutive components. This can also be seen in Fig. 29 for the synchronization specification of the animation and audio block in the example introduced in Section II-C3. The animation must be split into the parts Animation 1, Animation 2, and Animation 3 to be correctly synchronized with the audio block.

Accordingly, a synchronized multimedia object used as a component in another synchronization can no longer be regarded as an abstract unit if it has to be synchronized between the beginning and end of its presentation. That is to say, basic hierarchical specifications do not support adequate abstraction for the internal structure of multimedia objects. In addition, there are synchronization conditions which cannot be represented using hierarchical structures. For example, the three objects shown in Fig. 30 are presented in parallel, where any pair of objects is synchronized but always independently

TABLE IX
ASSESSMENT OF THE BASIC HIERARCHICAL SYNCHRONIZATION SPECIFICATION

Advantages	Disadvantages
Easy to understand	Additional description of skew QoS necessary
Natural support of hierarchies	For the presentation of time-independent media objects presentation durations must be added
Integration of interactive objects is easy	Splitting of media objects for synchronization purposes necessary
	No adequate abstractions for media object contents
	Some synchronization scenarios can not be described

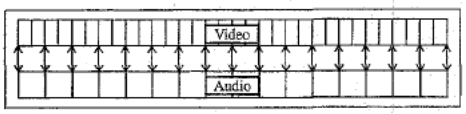

Fig. 31. Lip synchronization in the reference point synchronization model.

of the third object. To specify this synchronization, additional synchronization points must be used.

The assessment of the basic hierarchical method is summarized in Table IX.

2) Reference Points: In the case of synchronization via reference points [8], [67], time-dependent single-medium objects are regarded as sequences of LDU's. The start and stop times of the presentation of a media object, in addition to the start times of the subunits of time-dependent media objects, are called *reference points*. Synchronization between objects is defined by connecting reference points of media objects. A set of connected reference points is called a *synchronization point*. The presentation of the subunits that participate in the same synchronization point must be started or stopped when the synchronization point is reached. This approach to synchronization specifies temporal relations between objects without explicit reference to time.

Like synchronization based on a time axis, this description allows synchronization at any time during the presentation of an object; moreover, object presentations of unpredictable duration can be integrated easily. This type of specification is also very intuitive to use.

A drawback of reference point synchronization is that it requires mechanisms for detecting inconsistencies. In addition, synchronization based on reference points does not allow for specification of delays in a multimedia presentation. To solve this problem, Steinmetz [67] proposes time specifications which specify explicit real-time-based delays. The inclusion of timers also solves this problem. The specification based on a global timer can be regarded as a subset of the reference point synchronization: a timer according to Fig. 10 can be used as global timer and all objects refer only to this timer.

In a reference point synchronization specification, the coherence between data streams can be described by specifying a suitable set of synchronization points between the two data streams. A close lip synchronization with a maximal skew of ± 80 ms can be realized by setting a synchronization point, for example, every second frame of a video (Fig. 31). If no lip synchronization is required, it may be sufficient to set a synchronization point every 10 frames of the video. Therefore,

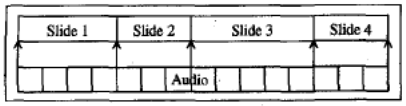

Fig. 32. Example of a slide show with an audio sequence in the reference point model.

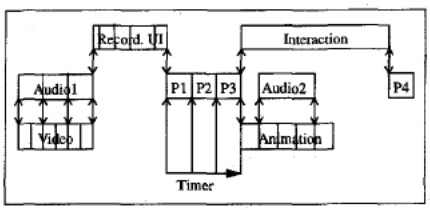

Fig. 33. Reference point synchronization specification example (with the integration of time-dependent and time-independent media objects, as well as closed and open LDU's).

TABLE X
ASSESSMENT OF THE REFERENCE POINT SYNCHRONIZATION SPECIFICATION

Advantages	Disadvantages
Easy to understand	More difficult to maintain because of the direct description of object relations
Natural description of temporal relations	Hierarchies sometimes complex
Easy integration of interactive objects	
Integrated description of skew QoS	
Easy integration of time-dependent objects	
Time axes-based synchronization can be regarded as special case of the reference point synchronization method	

the specification of the skew QoS is directly integrated into this specification method.

An example of the synchronized integration of time-dependent and time-independent media objects is shown in Fig. 32. Starting and stopping a slide presentation are initiated by reaching suitable LDU's in the audio presentation.

The application example of Fig. 13 can be completely specified with the reference point synchronization model (Fig. 33).

Hierarchies in the reference point synchronization method can be created by regarding a set of synchronized objects as one object, with the start of the first object and end of the last object as reference points. Virtual reference points for this presentation can be specified and mapped to the reference points within the hierarchy. The semantic of this mapping can become complex in the case that objects of unknown duration are included in the hierarchy. The assessment of the reference point method is summarized in Table X.

3) Timed Petri Nets: Another type of specification is based on *petri nets* [41], [43] that are extended with duration specifications at places, a kind of *timed petri net.*

The rules for a timed petri net are:

- A transition fires, if all input places contain a nonblocking token.
- If a transition fires, a token is removed from each input place and a token is added to each output place.
- A token that is added to a new place is blocked for the duration that is assigned to this place.

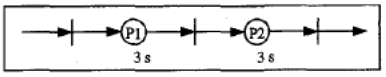

Fig. 34. Petri net specification of a slide show.

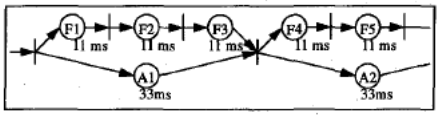

Fig. 35. Petri net lip synchronization (F = Video frame, A = Audio block).

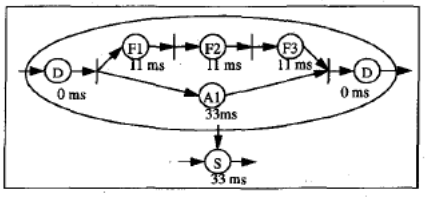

Fig. 36. Petri net hierarchy comprised of the synchronization of A1 and F1 to F3.

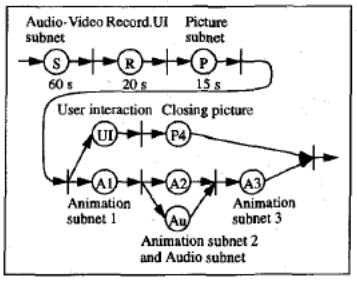

Fig. 37. Petri net specification example.

A slide show can be specified by assigning corresponding durations to the places (Fig. 34).

For time-dependent media objects, each place in the petri net represents an LDU. Lip synchronization can be modeled on the basis of connecting appropriate LDU's by transitions (Fig. 35).

It is also possible to combine a set of consecutive LDU's to one place as long as no inter-object synchronization exists between these LDU's and others. A hierarchy can be constructed by creating subnets that are assigned to a place. The duration of the longest path in the subnet is assigned to the place (Fig. 36).

The application example of Fig. 13 can be modeled as shown in Fig. 37. The subnets are not shown because they can be created by the straightforward use of the techniques described above.

Timed petri nets allow all kinds of synchronization specifications. The main drawbacks are the complex specifications and the insufficient abstraction of media object content because, much like the hierarchical specification, the media objects must be split into subobjects. The assessment of the timed petri net method is summarized in Table XI.

TABLE XI
ASSESSMENT OF THE PETRI NET SYNCHRONIZATION SPECIFICATION

Advantages	Disadvantages
Hierarchies can be created	Not easy to handle
Easy integration of time-independent objects	Complex specification
Easy integration of interactive objects	Splitting of media objects
Integrated skew QoS	Insufficient abstraction of media object content

TABLE XII
EVENT-BASED SPECIFICATION EXAMPLE

Event / Action	Start	Audio1.stop	Timer1. ready	...
Audio1	start			
Video	start			
Pic.1		start	stop	
Timer1		start(3)		
Pic.2			start	
...				

TABLE XIII
ASSESSMENT OF THE EVENT-BASED SYNCHRONIZATION SPECIFICATION

Advantages	Disadvantages
Easy integration of interactive objects	Not easy to handle
Easy extensible by new events	Complex specification
Flexible because any event can be specified	Hard to maintain
	Integration of time-dependent objects by using additional timers
	Separate descriptions of skew QoS necessary
	Difficult use of hierarchies

F. Event-Based Synchronization

In the case of event-based synchronization, presentation actions are initiated by synchronization events, e.g., as in HyTime and HyperODA [7]. Typical presentation actions are:

- Start a presentation.
- Stop a presentation.
- Prepare a presentation.

The events that initiate presentation actions may be external (e.g., generated by a timer) or internal to the presentation generated by a time-dependent media object that reaches a specific LDU.

Table XII sketches an event-based synchronization for parts of the application example of Fig. 13.

This type of specification is easily extended to new synchronization types. The major drawback is that this type of specification is difficult to handle in the case of realistic scenarios. The user is lost in this state transition type of synchronization specification, hence creation and maintenance becomes difficult. The assessment of the event-based method is summarized in Table XIII.

G. Scripts

A *script* in this context is a textual description of a synchronization scenario [33], [72]. Elements of scripts are activities and subscripts. Often, scripts become full programming languages extended by timing operations. Scripts may rely on different specification methods.

TABLE XIV
ASSESSMENT OF THE SCRIPT SYNCHRONIZATION SPECIFICATION

Advantages	Disadvantages
Good support for hierarchies	Not easy to handle
Logical objects can be kept	Complex specification
Easy integration of time-independent objects	Implicit usage of common timers necessary
Easy integration of interactive objects	Special constructs for skew QoS necessary
Easy extensible by new synchronization constructs	
Flexible because programmable	

A typical example is a script that is based on the basic hierarchical method and supports three main operations: serial presentation, parallel presentation and the repeated presentation of a media object.

The following example sketches a script for the application example from Fig. 13. >> denotes a serial presentation, & denotes a parallel presentation and $n*$ denotes a presentation repeated n times ([72]):

```
activity DigAudio                Audio(''video.au'');
activity SMP                     Video(''video.smp'');
activity XRecorder               Recorder(''window.rec'');
activity Picture                 Picture1(''picture1.jpeg'');
activity Picture                 Picture2(''picture2.jpeg'');
activity Picture                 Picture3(''picture3.jpeg'');
activity Picture                 Picture4(''picture4.jpeg'');
activity StartInteraction        Selection;
activity DigAudio                AniAudio(''animation.au'');
activity RTAnima                 Animation(''animation.ani'');

script Picture_sequence
    3 Pictures=                  Picture1.Duration(5) >>
                                 Picture2.Duration(5)>>
                                 Picture3.Duration(5);

script Lipsynch AV = Audio & Video;
script AniComment AA = Animation & AniAudio.Translate(2);
script Multimedia
  Application_example {
                                 AV >>
                                 Record. UI >>
                                 3Pictures >>
                                 ((Selection Picture4) & AA)
}
```

Scripts are very powerful because they represent full programming environments. A disadvantage is that this method is more procedural than declarative. The declarative approach seems to be more easier for the user to handle. The assessment of the script method (based on the basic hierarchical method) is summarized in Table XIV.

H. Comment

The presented synchronization specification methods have different specification capabilities and are different with respect to user-friendliness, but many of them just present different "views" of the same problem. The different specification capabilities restrict the mapping between specifications of different methods to the common subset.

The selection of an appropriate specification method depends on the targeted application and on the existing environment. As the temporal behavior of multimedia objects is only one part of a presentation, we must keep in mind the context as it may be an audio/video editor or an MHEG presentation tool.

The selected method must fit into the selected environment. There is no "best" or "worst" solution. For simple presentations without user interaction, the method based on a global timer seems to be appropriate. For complex structures with interaction, for example, the reference point model seems to be suitable.

In many cases, users will not directly specify the synchronization using a specific specification method. They will instead use a graphical authoring system that may produce specifications based on different methods. Experience shows that usually one of these specification methods underlies the construction of the user interface and therefore indirectly the advantages and disadvantages of the method reflect themselves at the user interface. In addition, many authoring systems allow the author to step out of the high-level graphical representation and to specify a complex synchronization directly at the lowest synchronization specification level, e.g., the textual level provided by the underlying method, which is not the best way to proceed.

VI. Case Studies

Some interesting approaches to multimedia synchronization are described in this section and classified according to the reference model presented previously. In particular, we analyze synchronization aspects in standards of multimedia information exchange and the respective run-time environments and prototype multimedia systems which comprise several layers of the synchronization reference model.

A. MHEG

The envisaged international standard *Coded Representation of Multimedia and Hypermedia Objects* [52] will define the representation and encoding of final form multimedia and hypermedia information objects to be interchanged within or across applications and services. The standard is known as *MHEG* because it is developed by the Multimedia and Hypermedia information coding Expert Group.

MHEG aims to support multimedia services that use a huge amount of structured information. This data is stored at the workstation, on a digital interchange medium like a CD or are accessed via telecommunication services.

MHEG ensures that the multimedia objects are portable and remain accessible in evolving environments, and it tries to meet the requirements of applications with a need for a final form representation of information, for multimedia synchronization, for the support of interactivity with the user and for real-time interchange of composite objects. MHEG can be used as the object layer interface format.

1) MHEG Objects: MHEG uses an object-oriented approach to achieve active, autonomous, reusable objects. It focuses on the generic structure of the objects. *Content objects* encapsulate media objects that are part of the presentation. For coding the content objects, media specific standards like JPEG and MPEG are used. *Composite objects* act as a "container" to group a set of MHEG objects. *Action objects* are used to exchange sets of actions to be performed between objects. Examples of actions are *prepare* to set the object in a presentable state, *run* to start the presentation and *stop* to end the presentation. *Link objects* are used to specify spatial, temporal and conditional relations between MHEG objects. An MHEG link is directional and connects one *source object* with one or more *linked objects* (Fig. 38). The MHEG runtime environment (called the MHEG engine) monitors the status of the conditions and triggers the execution of actions associated with the link. A *script object* is an encapsulation of scripts which can be executed by software separate from the MHEG engine. Selection objects are used for user interactions. Descriptor objects provide additional information about exchanged objects that are necessary for processing resource negotiation. In addition, presentable and template objects are concepts to use a single object several times in one or several presentations.

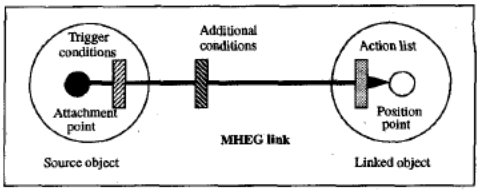

Fig. 38. MHEG link.

To support application specific requirements applications may extend MHEG classes by attributes of the objects, actions to be applied to the objects, conditions for triggering links, and media types. An application programming interface for the handling of these objects is not defined by MHEG. Instead mandatory and optional interface facilities are part of it. Mandatory facilities are basic actions (e.g., *run* and *stop* the presentation), optional facilities comprise creation and access to objects (e.g., *create object, set value* and *get value*).

2) Synchronization in MHEG: The generic space in *MHEG* provides a virtual coordinate system that is used to specify the layout and relation of content objects in space and time according to the virtual axes-based specification method. The generic space has one time axis of infinite length measured in *generic time units (GTU's)*. The MHEG run-time environment must map the GTU's to *physical time units (PTU's)*. If no mapping is specified, the default is one GTU mapped to one millisecond. Three spatial axes (X = latitude, Y = longitude, Z = altitude) are used in the generic space. Each axis is of finite length in an interval of $[-32768, +32767]$. Units are *generic space units (GSU's)*. Also, the MHEG engine must perform mapping from the virtual to the real coordinate space.

The presentation of content objects is based on the exchange of action objects sent to an object. Action objects can be combined to form an action list. Parallel action lists are executed in parallel. Each list is composed of a delay followed by delayed sequential actions that are processed serially by the MHEG engine, as shown in Fig. 39.

By using links it is possible to synchronize presentations based on events. Link conditions may be associated with an event. If the conditions associated with a link are fulfilled, the link is triggered and actions assigned to this link are performed.

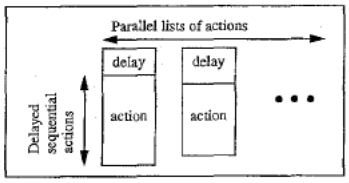

Fig. 39. Lists of actions.

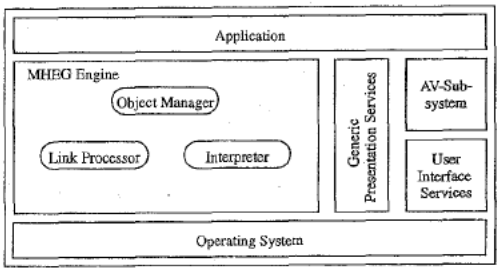

Fig. 40. Architecture of an MHEG engine.

3) MHEG Engine: At the European Networking Center in Heidelberg, an MHEG engine [27] has been developed. The MHEG engine is an implementation of the object layer. The architecture of the MHEG engine is shown in Fig. 40.

The *generic presentation services* of the engine provide abstractions from the presentation modules used to present the content objects. The *audio/video-subsystem* is a stream layer implementation. This component is responsible for the presentation of the continuous media streams, e.g., audio/video streams. The *user interface services* provide the presentation of time-independent media, like text and graphics, and the processing of user interactions, e.g., buttons and forms.

The MHEG engine receives the MHEG objects from the application. The *object manager* manages these objects in the run-time environment. The *interpreter* processes the action objects and events. It is responsible for initiating the preparation and presentation of the objects. The *link processor* monitors the states of objects and triggers links, if the trigger conditions of a link are fulfilled.

The run-time system communicates with the presentation services by events. The *user interface services* provide events that indicate user actions. The *audio/video-subsystem* provides events about the status of the presentation streams, like end of the presentation of a stream or reaching a cuepoint in a stream.

4) Summary: MHEG is a standardized exchange format that can be used as the interface format at the object layer. The synchronization is based on the virtual axes- and event-based methods. An MHEG engine represents the object layer run-time environment. The object layer implementation of the described engine is based on media servers. The audio/video-subsystem represents the stream layer. Fig. 41 shows the relation to the synchronization reference model.

Regarding distributed environments, the processing model of MHEG has the following drawback: the duration between the preparation and display action is coded in the MHEG object, but the duration depends on the run-time environment; therefore, this duration should be computed by the MHEG engine.

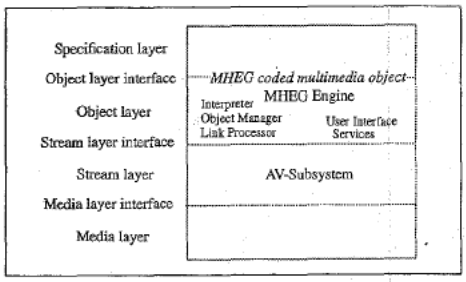

Fig. 41. Classification of MHEG and MHEG engine components according to the synchronization reference model.

B. HyTime

HyTime is an international standard (ISO/IEC 10744) [36] for the structured representation of hypermedia information. HyTime is an application of the *standardized general markup language (SGML)* [65].

SGML is designed for document exchange, whereby the document structure is of great importance, but the layout is a local matter. The logical structure is defined by markup commands that are inserted in the text. The markups divide the text into SGML elements. For each SGML document, a *data type definition (DTD)* exists which declares the element types of a document, the attributes of the elements and how the instances are hierarchically related. A typical use of SGML is the publishing industry where an author is responsible for the content and structure of the document, and the publisher is responsible for the layout. As the content of the document is not restricted by SGML, elements can be of type text, picture or other multimedia data.

HyTime defines how markup and DTD's can be used to describe the structure of hyperlinked time-based multimedia documents. HyTime does not define the format or encoding of elements. It provides the framework for defining the relationship between these elements.

HyTime supports addresses to identify a certain piece of information within an element, linking facilities to establish links between parts of elements and temporal and spatial alignment specifications to describe the relationships between media objects.

HyTime defines architectural forms that represent SGML element declaration templates with associated attributes. The semantic of these architectural forms is defined by HyTime. A HyTime application designer creates a HyTime-conforming DTD using the architectural forms he/she needs for the HyTime document. In the HyTime DTD each element type is associated with an architectural form by a special HyTime attribute.

The HyTime architectural forms are grouped into the following modules:

- The Base Module specifies the architectural forms that the document comprises.
- The Measurement Module is used to add dimensions, measurement and counting to the documents. Media ob-

jects in the document can be placed along with the dimensions.

- The Location Address Module provides the means to address locations in a document. The following addressing modes are supported:
  - Name Space Addressing Schema: Addressing to a name identifying a piece of information.
  - Coordinate Location Schema: Addressing by referring to an interval of a coordinate space if measuring along the coordinate space is possible. An example is to address to a part of an audio sequence.
  - Semantic Location Schema: Addressing by using application-specific constructs.
- The Scheduling Module places media objects into finite coordinate spaces (FCS's). These spaces are collections of application-defined axes. To add measures to the axes, the measurement module is needed. HyTime does not know the dimension of its media objects. So-called events are used for the presentation of media objects. An event is an encapsulation of a media object and comprises the layout specification related to an FCS. The events can be placed absolutely or relatively to other events within the FCS's.
- The Hyperlink Module enables building link connections between media objects. Endpoints can be defined using the location address, measurements and scheduling modules.
- The Rendition Module is used to specify how the events of a source FCS, that typically provides a generic presentation description, are transformed to a target FCS that is used for a particular presentation. During the mapping, presentation-related modifications are executed, e.g., changing the color representation, projection of the dimensions from the source to the target FCS or scaling of the presentation.

1) HyTime Engine: The task of a HyTime engine is to take the output of an SGML parser, to recognize architectural forms and to perform the HyTime-specific and application-independent processing. Typical subtasks of the HyTime engine are hyperlink resolution, object addressing, parsing of measures and schedules, and transformation of schedules and dimensions. The resulting information is then provided to the HyTime application.

The HyTime engine, HyOctane [12], developed at the University of Massachusetts at Lowell, has the following architecture: an SGML parser takes as input the application data type definition that is used for the document and the HyTime document instance. It stores the document object's markups and contents, as well as the application's DTD in the SGML layer of a database. The HyTime engine takes as input the information stored in the SGML layer of the database. It identifies the architectural forms, resolves addresses from the location address module, handles the functions of the scheduling module and performs the mapping specified in the rendition module. It stores the information about elements of the document that are instances of architectural forms in the HyTime layer of the database. The application layer of the database stores the objects and their attributes, as defined by the DTD. An application presenter gets the information it needs for the presentation of the database content, including the links between objects and the presentation coordinates to use for the presentation, from the database.

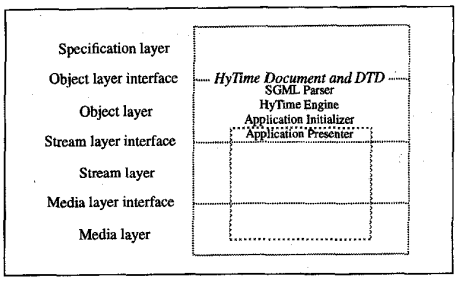

Fig. 42. Classification of HyTime and the HyTime engine according to the synchronization reference model.

2) Summary: HyTime is applicable to many application areas. It does not standardize content formats, encoding, document types or specific SGML DTD's. It provides a framework for addressing portions of hypermedia document contents and for the definition of linking, alignment, and synchronization. In the context of the synchronization reference model, a HyTime document, together with its DTD, can be used as input to the object layer. The synchronization is based on the virtual axes synchronization method. The SGML- and HyTime-related preprocessing is done by the HyTime engine in the object layer. The application presenter provides the other object layer and stream layer functionalities. Fig. 42 shows the relation to the synchronization reference model.

Other classification possibilities are to regard the database of the HyTime Engine as an object layer interface format or to use the database to generate an MHEG specification. In the latter case, the HyTime engine can be regarded as part of a format conversion tool at the specification layer.

C. Firefly System

The objective of the approach of Buchanan and Zellweger [16], [17] is to automatically generate consistent presentation schedules for interactive multimedia documents that comprise media objects of predictable behavior (like audio and video) and objects of unpredictable behavior (like user interactions). The generation algorithm is comprised of two phases. At the first phase, before execution of the presentation, high-level temporal specifications for a document are used to compute a presentation schedule, as far as possible without knowing the unpredictable durations. In the second phase during the presentation, the scheduling depending on unpredictable durations, is incorporated.

The specification of the temporal constraints distinguishes media-level specifications that describe the temporal behavior of individual media objects and document-level specifications that describe the temporal behavior of a complete multimedia document, in particular the temporal relations between single media objects. Media items are used for the media-level specification. They provide a reference to a media object and are used to describe the temporal behavior of this media object. A media item consists of:

- Events, which represent points in time during the presentation of a media object. They are comparable to a reference point.
- Durations, which specify the duration between two subsequent events in a media object. A duration is represented by a triple of values: minDuration, optDuration, and maxDuration. If the three values are equal, the presentation duration is fixed. If they specify an interval, the presentation is adjustable. No values are assigned for an unpredictable duration.
- Costs, which can be used as measurement for the degree of degradation in the case of stretching the presentation toward the maximal duration, and respectively shrinking it toward the minimum duration.

A document-level specification consists of:

- Media items, which are involved in the presentation.
- Temporal constraints, which are used to describe explicit temporal relations between events in one or more media items. Temporal constraints are classified into temporal equalities that describe a fixed temporal relation between two events (e.g., same time, one event 10 s before the other), and temporal inequalities that describe a temporal relation without a specified time (e.g., one event before the other, one event at least 10 s and at most 20 s before the other).
- Operations, which can be associated with an event and include nonaltering presentation-related operations, like increase volume of an audio presentation, and time-altering operations, like increase-playback-speed.
- Duration and costs, which can be described according to the media level. At the document level, this is used to describe a different behavior for several instances of one media item in a document.
- Unpredictable event control, which allows activation and deactivation of unpredictable events.

To support the development of temporal specifications, a graphical representation of the specification is supported. The synchronization specification method is a combination of reference point and interval-based synchronization.

The scheduler for the presentation that is located at the object layer is divided into two parts: the compile-time scheduler and the run-time scheduler. The compile-time scheduler constructs a main schedule that controls the parts of a document that are predictable, and auxiliary schedules that control the parts of the document that depend on unpredictable events. It is an example of compile-time schedule computation at the object layer.

The algorithm contains four parts:

- In the *obtaining durations and costs step*, the duration and costs for each media item are obtained. To do this, the media and document-level specifications for a media item are unified and time-altering operations are incorporated into the computation of the durations.
- In the *finding connected components step*, a union-find algorithm is used to find connected parts of a document. Two events are in the same connected component if they are related by a predictable duration or a temporal constraint. The connected components are called predictable, if there are no unpredictable events that trigger events of the component. Otherwise, they are called unpredictable.
- The *assigning times to events step* computes for each event in a connected component the time for that event with respect to the start time of the component. It uses a simplex algorithm with the durations and temporal constraints as constraints for the algorithm and the minimization of the costs as its objective function.
- In the *creating commands step*, the previous results are used to create the commands for the execution. A command includes a time when it must be executed, the media item to process, an associated event, the list of unpredictable events to be activated or deactivated and the operations to be executed. All commands of the predictable components are integrated in the main schedule. For each unpredictable component, a separate auxiliary schedule is constructed. To improve performance for a continuous media object with units of fixed durations, only the start of the complete media object and events that refer to other media objects are considered, not every single event within the media item. It is assumed that the stream presentation scheduling is done separately.

The run-time scheduler is an example of runtime schedule computation at the object layer and controls the document clock, the execution schedule and handles the unpredictable events. After the compile-time scheduler has produced the schedules, the run-time scheduler copies the main schedule into the execution schedule and starts the document clock. If the document clock reaches a time with an associated command, it initiates the command. If an activated unpredictable event occurs that triggers an unpredictable component, the run-time scheduler merges the corresponding schedule into the execution schedule taking the actual document time as start time for the first command in the schedule to merge. Because unpredictable components may be triggered several times, the run-time scheduler marks the instances in the execution schedule to be able to distinguish the commands for the different instances of an unpredictable schedule.

1) Summary: The Firefly system provides complete synchronization support. At the specification layer, an editor is provided. The temporal relations based on the reference point and interval-based specification methods are used at the object layer interface. The Scheduler provides compiletime and runtime computation of presentation schedules at the object layer. The schedule of streams is only initiated at the object layer, the execution is located at the stream layer. Fig. 43 shows the relation to the synchronization reference model.

The system provides well-organized scheduling planning and integration of unpredictable durations. Currently, the system does not consider media preparation durations, presentation restrictions by insufficient or missing local resources or delays introduced by networks.

D. MODE

The *multimedia objects in a distributed environment (MODE)* system [11], developed at the University of

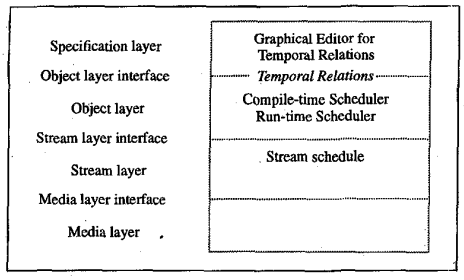

Fig. 43. Classification of the Firefly system according to the synchronization reference model.

Karlsruhe, is a comprehensive approach to network transparent synchronization specification and scheduling in heterogeneous distributed systems. The heart of MODE is a distributed multimedia presentation service which shares a customized multimedia object model, synchronization specifications and QoS requirements with a given application. It also shares knowledge about networks and workstations with a given run-time environment. The distributed service uses all this information for synchronization scheduling when the presentation of a compound multimedia object is requested from the application. Thereby, it adapts the QoS of the presentation to the available resources, taking into account a cost model and the QoS requirements given by the application.

The MODE system contains the following synchronization-related components:

- The Synchronization Editor at the specification layer, which is used to create synchronization and layout specifications for multimedia presentations.
- The MODE Server Manager at the object layer, which coordinates the execution of the presentation service calls. This includes the coordination of the creation of units of presentation (presentation objects) out of basic units of information (information objects) and the transport of objects in a distributed environment.
- The Local Synchronizer, which receives locally the presentation objects and initiates their local presentation according to a synchronization specification.
- The Optimizer, part of the MODE Server Manager, which performs the planning of the distributed synchronization and chooses presentation qualities and presentation forms depending on user demands, network and workstation capabilities and presentation performance.

1) Synchronization Model: In the MODE system, a synchronization model based on synchronization at reference points is used [8]. This model is extended to cover handling of time intervals, objects of unpredictable duration and conditions which may be raised by the underlying distributed heterogeneous environment.

A synchronization specification created with the Synchronization Editor is stored in textual form. The syntax of this specification is defined in the context-free grammar of the Synchronization Description Language. This way, a synchronization specification can be used by MODE components, independent of their implementation language and environment.

MODE distinguishes between dynamic basic objects and static basic objects. A presentation of a dynamic basic object is composed of a sequence of presentation objects. This corresponds to a stream of LDU's. The index of each presentation object is a reference point. The presentation of a static basic object, that may be a time-independent media object, as well as an interactive object, has only two reference points, the beginning and the end of the presentation. The description of a reference point, together with the corresponding basic object, is called a synchronization element, denoted in the form BasicObject.ReferencePoint. Two or more synchronization elements can be combined into a synchronization point. An entire inter-object synchronization is defined by the list of all synchronization points.

A presentation quality can be specified for each basic object. It is described by a set of attributes comprising an attribute name, preferred value and value domain that describes all possible values for this attribute.

2) Local Synchronizer: The Local Synchronizer performs synchronized presentations according to the synchronization model introduced above. This comprises both intra-object and inter-object synchronization. For intra-object synchronization, a presentation thread is created which manages the presentation of a dynamic basic object. Threads with different priorities may be used to implement priorities of basic objects. All presentations of static basic objects are managed by a single thread.

Synchronization is performed by a signaling mechanism. Each presentation thread reaching a synchronization point sends a corresponding signal to all other presentation threads involved in the synchronization point. Having received such a signal, other presentation threads may perform acceleration actions, if necessary. After the dispatch of all signals, the presentation thread waits until it receives signals from all other participating threads of the synchronization point; meanwhile, it may perform a waiting action.

3) Planning and Execution of the Distributed Presentation: Before starting any presentation, the Optimizer is invoked. The Optimizer uses a heuristic search algorithm, taking into account the special conditions of a distributed environment like multiple steps of the synchronization in a distributed environment, multiple communication patterns, clock skews and several possible locations of synchronization operations into account. It uses information about the network, like the available bandwidth, service qualities and available resources at the workstation, as well as information about the processing demands for media objects. This information is provided to the Optimizer by environment and application media descriptions [10].

The planning result determines the achievable quality value for each presentation attribute according to both user demands and network and workstation resources. The result of the planning process is the MODE Flow Graph [9] that describes the times and nodes at which operations must be executed. The partitioned Flow Graph is delivered to the involved nodes and executed at run-time by the distributed MODE Server Manager.

4) Exceptions Caused by the Distributed Environment: The correct temporal execution of the plan depends on temporal

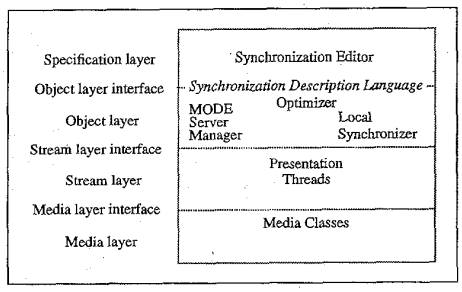

Fig. 44. Classification of the MODE system according to the synchronization reference model.

guarantees provided by the underlying environment. Therefore, MODE provides several guarantee levels. If the underlying distributed environment cannot give full guarantees, MODE considers the possible error conditions. Three types of actions are used to define a behavior in the case of exception conditions, which may be raised during a distributed synchronized presentation: 1) A waiting action can be carried out if a presentation of a dynamic basic object has reached a synchronization point and waits longer than a specified time at this synchronization point. Possible waiting actions are, for example, continuing presentation of the last presentation object ("freezing" a video, etc.), pausing or cancellation of the synchronization point. 2) When a presentation of a dynamic basic object has reached a synchronization point and waits for other objects to reach this point, acceleration actions represent an alternative to waiting actions. They move the delayed dynamic basic objects to this synchronization point in due time. Possible actions include temporarily increasing the presentation speed or skipping all objects in the presentation up to the synchronization point. 3) When a presentation object does not arrive in time, it is possible to skip the object and to present the next one.

Priorities may be used for basic objects to reflect their sensitivity to delays in their presentation. For example, audio objects will usually be assigned higher priorities than video objects because a user recognizes jitter in an audio stream earlier than jitter in a video stream. Presentations with higher priorities are preferred over objects with lower priorities in both presentation and synchronization.

5) Summary: MODE is a complete synchronization system especially designed to support synchronization in a distributed environment. MODE provides a Synchronization Editor at the specification layer. The output of the editor is used as reference point-based interface format between the specification and the object layer. The Optimizer is part of the object layer and performs a compile-time computation of the presentation schedule before the start of the presentation. The MODE Server Manager and the Synchronizer are also part of the object layer. The threads generated by them for the handling of dynamic media objects are part of the stream layer. Fig. 44 shows the relation to the synchronization reference model.

E. Multimedia Tele-Orchestra

At the University of Ottawa, the Multimedia Communication Research Laboratory (MCRLab) of Prof. Nicolas D. Georganas has developed a multimedia synchronization system known as *multimedia tele-orchestra.* This system is comprised of a sophisticated specification schema, the Time Flow Graph (TFG) [46], and an implementation of this synchronization in a distributed environment [47]. In contrast to many other specification methods, the TFG takes into account that temporal knowledge may often be relative, i.e., it cannot be described by exact time parameters. The authors call this a fuzzy scenario. In addition, the duration of presentation parts may be imprecise and not known in advance. Hence, neither the exact occurring time points nor the duration are required to specify synchronization.

The notion of intervals serves as a basis for the TFG. In [46] it is shown that all temporal relationships between intervals can be represented with TFG's. This leads to a partial sequential ordering which is used by the actual processing of synchronization at presentation time. With respect to our synchronization reference model, the TFG is an interval-based method located at the specification layer and also covers the interface between this layer and the object layer (see Fig. 45).

Based on the TFG, a distributed multimedia synchronization schema was developed and became known as the *synchronization controller for multimedia communication (SCMC)* [47]. As a key feature, it takes into account that data may be originated by different sources located at different places. SCMC is targeted to run over ATM networks. However, the same algorithms can be used to operate on top of other multimedia-capable network configurations like Ethernet 10 Base-T, 100 Base-T, and IsoEthernet.

In the tele-orchestration approach, a second component, the *temporal presentation controller (TPC),* is in charge of calculating a schedule with the earliest possible time to present objects at a remote computer. The result of the TPC, i.e., the respective schedule, is subsequently passed to the SCMC, which will actually control data processing to match the synchronization specification. In terms of the synchronization reference model, the SCMC makes use of individual LDU's. It does not rely on a stream. The SCMC provides to its user the capability to provide synchronization between individual data streams. Hence, the SCMC is located at the media layer, as well as the stream layer. The TPC maps time constraints defined by the TFG onto SCMC primitives. The TPC calculates local schedules, whereas the SCMC unifies all local schedules to an actual implementation of the demanded synchronization. Hence, the TPC is located in the object layer according to Fig. 45.

1) Summary: Tele-orchestra nicely covers the aspects of all layers of our synchronization reference model. Distribution is known and handled at the specification and stream layers. In [48], performance analysis results of this synchronization schema are presented.

F. Littles Framework

The main objective of this framework, currently integrated at the University of Boston [52] in a multimedia information system, is to support the retrieval and delivery of multimedia data. This system is comprised of methods for synchronization

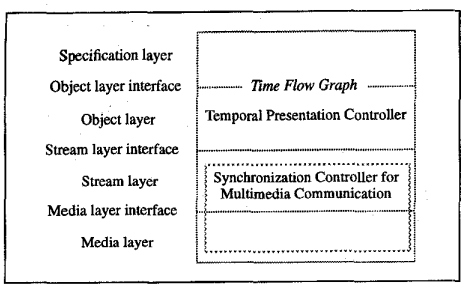

Fig. 45. Classification of the tele-orchestra system according to the synchronization reference model.

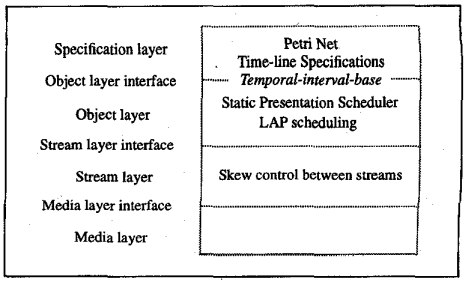

Fig. 46. Classification of Littles Framework according to the synchronization reference model.

specification, data representation, temporal access control and run-time intermedia synchronization. In particular, it provides mechanisms to overcome delays caused by storage, communication and computations on media objects. It also provides mechanisms for scalability and graceful degradation of multimedia services.

The specification of the synchronization is based on petri nets [39] and global timer-based specifications that are mapped to the *temporal-interval-base (TIB)* modeling approach. The temporal relations in this model include a start time for a data element, the duration of its presentation and the end time for it. Relative positioning is defined by delays between the start times of presentations.

Based on this specification, static and dynamic presentation scheduling is computed at the object layer. As an example of a simple planning algorithm in an environment with resource restriction, we present the static playback schedule computation algorithm [43], [44]. It assumes that the data elements are stored in a remote database. The data must be transported to the presentation workstation via a packet-switched network with restricted capacity. In a first step, the synchronization specification is used to compute the point of time for the start of the presentation (p_i) for each data unit. This is easily possible using the duration of the presentations (m_i). Using the start points of the presentations, it is necessary to compute the point of times to access the data units (q_i) from the database because they need a time (T_i) to be transported.

Let D_p be the constant propagation delay, D_t the delay proportional to the packet size (medium packet size/channel capacity) and D_v the variable load-dependent delay. Then, T_i is defined as $T_i = D_p + D_t + D_v$.

The following conditions must be fulfilled:

- $p_i \geq q_i + T_i$ (The data units must be available in time.)
- $q_{i-1} \leq q_i - T_{i-1} + D_p$ (Data should be accessed when previous sending of data is finished.)

The following algorithm is used to compute q_i:

```
q[m] = p[m] - T[m]                    // Start with last data unit.

for i = 0 to m - 2
  if q[m - i] < p[m - i - 1] - Dp      // Collision
    q[m - i - 1] =
    q[m - 1] - T[m - i - 1] + Dp       // Resolve collision
  else
    q[m - i - 1] =
    p[m - i - 1] - T[m - i - 1]        // No collision
  end
end
```

Because static scheduling does not consider dynamic changes in the environment, as well as commands from the user that alter, for example, the presentation speed, dynamic scheduling is introduced. The dynamic scheduling approach is called *limited a priori (LAP) scheduling*. It performs the scheduling and reservation of resources only for a short period of time. The multimedia presentation is split into components of similar resource usage. For these components, the schedules are computed and statistical resource reservation is used. Subsequently, the session scheduler executes the presentation of the components. In the case of user-initiated presentation manipulation operations or of load changes, the schedules are recalculated.

To support interstream synchronization, skew control mechanisms are supported. They are based on dropping and duplicating data units, in the case that a queue representing the stream processing reaches low or high threshold values.

The petri net and time-line specifications in the specification layer are mapped to a TIB specification as an object layer interface format that is a type of interval-based synchronization. The compile-time and run-time scheduling is located at the object layer. Additional skew control is provided at the stream layer.

1) Summary: The Littles framework represents a well-defined approach combining several layers. Its conception is concentrated on the retrieval of multimedia objects on one server and considers only a reduced set of distribution relevant parameters. Fig. 46 shows the relation to the synchronization reference model.

G. ACME

Abstractions for continuous media (ACME) [5] is an I/O server for continuous data streams in the stream layer. The server controls a set of physical devices. Users can define logical devices as abstractions from physical devices. A stream path is built by connecting input and output devices. The connection may be a real network connection. The stream consists of LDU's with an assigned time stamp.

A *logical time system (LTS)* synchronizes the I/O of logical devices. An LTS owns a clock that can be bound to the device which is most sensitive against delays, or it may be driven by a specified connection. Each LDU will be processed by a logical device, if the time stamp matches the LTS clock.

A blocking caused by a connection may occur. In this case, the connection's input device is blocked and must buffer more

and more units. The output device is starving, because it does not get enough LDU's. The blocking is resolved by skipping LDU's or by pausing the LTS in the case that a max skew value has been reached between time stamps of units and the LTS clock. The LTS is restarted if the time stamps of the logical data units are close to the paused LTS clock and an additional amount of data for the start-up phase of the resynchronization was received.

ACME offers a programming interface and provides support for media streams at the stream layer only.

H. Further Synchronization-Related Systems

Today, available multimedia extensions for operating systems, like Apple QuickTime [22], Microsoft Multimedia Extensions [53], and IBM Multimedia Presentation Manager/2 [35] contain synchronization mechanisms applied at the stream layer in the local domain. First networked systems like the IBM Ultimedia Server cover some synchronization issues in a distributed environment.

The Orchestration Service [18] provides a stream-oriented interface for synchronized playout of continuous media in a distributed environment. Nicolai [58], Little [42], Escobar [24], Shepherd [63], Ramanathan [62], and Anderson (as described in Section VI-G) have proposed techniques to control jitter among media streams in the stream layer. An evaluation and classification of these techniques is given in [23].

Based on the open distributed processing (ODP) standard [37], Coulson [19] has designed a multimedia support platform. The synchronization services are extensions of the computational model of ODP. The architecture comprises a synchronization management layer, a high level and a low level orchestration layer. The management layer offers services of the stream layer and event processing located at the object layer. The high level orchestration service supports groups of streams and the low level orchestration layer is responsible for single streams.

Stefani, Harzad, and Horn [64] have proposed the use of the synchronous language ESTEREL for the programming of multimedia synchronization. The language and its run-time environment provide support for fast event processing at the object layer.

At the University of Geneva [72], an object-oriented system with a global timer-based synchronization specification has been developed (see Section V-D1).

Bultermans framework [15] handles the problems of sharing network resources, synchronizing data coming from multiple sources and representation of data on heterogeneous hosts in a distributed environment. The components of the framework can manage the resources of the distributed environment for all active applications. The necessary information is provided by a specification of the applications resource and synchronization demands.

The Tactus system [21] includes a server for the synchronization of media at the sink node and an interface toolkit extended to support computation and controlling of streams and to deliver them to the presentation server. The scheduling is computed in advance to avoid delays at run-time. An introduced cut operation allows for low-latency reaction to user interactions by selecting between pre-computed schedules.

The use of traditional event-based user interface servers can be the reason for synchronization failures caused by the delay between calling the server to do a presentation and the actual presentation performance of the server. The time relations of demanded presentations get lost quite often. A proposal to handle this problem is to extend the window server to delay the execution of a presentation until a client-defined event occurs. This allows the reintroduction of the time relations between presentations in the server.

HyperODA [7] is a standardization activity that defines a multimedia document exchange format. It is an extension of the *open document architecture (ODA)* [14]. The extension of ODA to a multimedia and hypermedia document architecture requires new content architectures, e.g., for audio and video, and the definition of a model for the layout in time and integration with the layout in space. Intra-object synchronization is included in the content architectures, for example, in the audio content architecture. Inter-object synchronization is realized by event-based synchronization in prototypes. HyperODA is still in development.

I. Comment

A large number of synchronization supporting systems have been developed. Commercially available synchronization support is mainly restricted to the support of streams in local systems. Many research efforts are directed toward support of distributed environments, the development of presentation scheduling strategies and the integration of user interaction. The analysis of existing systems has confirmed that the synchronization reference model matches the structuring needs of multimedia synchronization systems.

VII. Summary and Outlook

A. Summary

In integrated multimedia systems, several aspects regarding synchronization must be considered. Unfortunately, the same term is used by many authors to denote different issues. In this contribution, synchronization-related terms are defined and the layers of synchronization processing in multimedia systems are classified in a synchronization reference model.

Intra-object synchronization is defined as the synchronization of the LDU's of one media object. Inter-object synchronization is the synchronization between the media objects.

For live synchronization, the synchronization specification directly results from the temporal relations during the capturing of the objects. In synthetic synchronization, the temporal relations between media objects are explicitly created.

A synchronization reference model is defined that classifies synchronization facilities and interfaces in layers and allows the identification and classification of media synchronization issues and approaches. The specification layer is comprised of tools for the creation and conversion of synchronization specifications. The object layer takes at its service interface

synchronization specifications as input. It plans and organizes the presentation, and initiates the presentation of time-independent media objects and of user interactions. For the presentation of continuous media, it uses the stream layer services. The stream layer interface supports abstractions of streams. It handles the intra-object synchronization and the synchronization between continuous media streams. The media layer interface hides access to the multimedia devices.

Several methods for the synthetic synchronization specification are discussed which have been developed over the last years. The enhanced interval-based method specifies relations between presentation intervals. The axes methods specify synchronization using a mapping of media objects to one or more common axes. The basic hierarchical method uses operations like parallel and serial to define relations between media objects. The reference point approach allows for specification by defining relations between the LDU's of media objects. Petri nets can be used to specify the flow of a presentation by using places with durations and attaching the start of presentation operations to the firing of transitions. The event-based method couples presentation operations to events. Scripts are a programming-oriented approach which make use of synchronization operations. All of these methods have been shown to provide different specification capabilities. Conversions or mappings of specifications between the different methods are possible, but often they are restricted to a common subset of the specification capabilities. In most cases, the user uses a graphical editor to specify the synchronization. The underlying specification methods usually reflect the user interface abstraction and the editors allow for direct access to these specification methods.

The required QoS of the temporal relationships to be presented to the viewer/listener is derived from the user's perception. Experiments have shown that a skew of more than $+/-80$ ms between an audio and a video stream is annoying, if lip synchronization is necessary. Further QoS requirements and a method for combining QoS requirements for multiple concurrent media have been presented.

A distributed execution environment causes many additional challenges because of the distribution of the synchronization specification and the respective media objects, the required communications and delays in the distributed environment and the demanded use of multi-party communication. Synchronization in the distributed environment is a multi-step synchronization process and planning problem.

The most well-known systems have been classified according to this synchronization reference model. This case study results in a comparison of the capabilities of the various approaches. On the other hand, it has proved the usefulness of the synchronization reference model.

B. Future Topics

The expansive development of multimedia applications demands that presentations can be executed on heterogeneous platforms. For that purpose, standards like MHEG, Java, and ScriptX are used that support exchangeable synchronization specifications. The success of multimedia exchange standards depends on the availability of run-time environments for the exchange formats. Which standard will be the most important in the future is still open. The availability of a standard format will also lead to the availability of supporting authoring systems.

The upcoming availability of multimedia teleservices demands an open distributed environment. For that purpose, an open stream mechanism and open object layer services are required. Initial work in the area of open streams in a heterogeneous environment has been done by the Interactive Multimedia Association, an industry-driven approach to open multimedia services. Additional efforts are required for the development of open object layer services.

VIII. Conclusion

In summary, we classified and compared the major approaches. Here, we focused on the demands, the various specification methods and basic run-time support concepts of the regarded approaches. Many more ideas, prototypes and products have implemented some kind of synchronization. However, it is still a matter of research to find out which are the most appropriate approaches for performing synchronization, especially in distributed environments.

Acknowledgment

The authors would like to thank C. Engler for his work in the synchronization experiments as well as T. Meyer-Boudnik and W. Effelsberg for their contributions and fruitful discussions in the reference model area.

References

[1] AFNOR Expert group, Multimedia Synchronization: Definitions and Model, Input Contribution on Time Variant Aspects and Synchronization in ODA-Extensions, ISO IEC JTC 1/SC 18/WG3, Feb. 1989.

[2] D. P. Anderson, R. G. Herrtwich, and C. Schaefer, "SRP: A resource reservation protocol for guaranteed-performance communication in the internet," Int. Computer Science Institute, Berkeley, CA, Tech. rep. TR-90-006, 1990.

[3] J. F. Allen, "Maintaining knowledge about temporal intervals," *Commun. ACM*, vol. 26, pp. 832–843, Nov. 1983.

[4] D. P. Anderson and P. Chan, "Toolkit support for multiuser audio/video applications," in *Network and Operating System Support for Digital Audio and Video, Second Int. Workshop, Heidelberg, Nov. 1991, Proc.*, R. G. Herrtwich, Ed. Berlin: Springer, 1992, pp. 230–241.

[5] D. P. Anderson and G. Homsy, "A continuous media I/O server and its synchronization mechanisms," *IEEE Computer*, vol. 24, pp. 51–57, Oct. 1991.

[6] D. P. Anderson and G. Homsy, "Synchronization policies and mechanisms in a continuous media I/O server," Int. Computer Science Institute, Berkeley, CA, Tech. rep. 91-003, 1991.

[7] W. Appelt, "HyperODA," ISO/IEC/JTC1/SC18/WG3.

[8] G. Blakowski, J. Hübel, U. Langrehr, and M. Mühlhaüser, "Tool support for the synchronisation and presentation of distributed multimedia," *Computer Commun.*, vol. 15, pp. 611–618, Dec. 1992.

[9] G. Blakowski, "The MODE-FLOW-GRAPH: A processing model for objects of distributed multimedia applications," in *Proc. Int. Symp. Communication*, 1994, pp. 646–649.

[10] ——, "High level services for distributed multimedia applications based on application media and environment descriptions," in *Proc. ACSC-15, 15th Australian Computer Science Conf.*, 1992; also *Australian Computer Science Communications*, vol. 14, pp. 93–109, Jan. 1992.

[11] ——, *Development and Runtime Support for Distributed Multimedia Applications* (in German). Aachen, Germany: Verlag Shaker, 1993.

[12] J. Budford, L. Rutledge, J. Rutledge, and C. Keskin, "HyOctane: a HyTime engine for an MMIS," *Multimedia Syst.*, vol. 1, no. 4, pp. 173–185, 1994.

[13] U. Bormann and C. Bormann, "Open processing of multimedia documents," (in German), *Informatik-Spektrum*, vol. 14, pp. 249–260, Oct. 1991.
[14] ——, "Standards for open document processing: Current state and future developments," *Computer Networks, ISDN Systems*, vol. 21, pp. 149–163, May 1991.
[15] D. C. A. Bulterman, "Specification and support of adaptable networked multimedia," *Multimedia Syst.*, vol. 1, no. 2, pp. 68–76, 1993.
[16] M. C. Buchanan and P. T. Zellweger, "Automatically generating consistent schedules for multimedia applications," *Multimedia Syst.*, vol. 1, no. 2, pp. 55–67, 1993.
[17] ——, "Automatic temporal layout mechanisms," in *Proc. 1st ACM Int. Multimedia Conf.*, Aug. 1–6, 1993.
[18] G. Coulson, F. Garcia, A. Campbell, and D. Hutchison, "Orchestration services for distributed multimedia synchronisation," in *Proc. 4th IFIP Int. Conf. High Performance Networking (HPN)*, 1992, pp. 14–18.
[19] G. Coulson, "Multimedia application support in open distributed processing," Ph.D. dissertation, Computing Department, Lancaster University, Apr. 1993.
[20] L. Delgrossi, R. G. Herrtwich, and F. O. Hoffmann, "An implementation of ST-II for the Heidelberg Transport System," *Internetworking Res. Experience*, vol. 5, 1994.
[21] R. D. Dannenberg, T. Neuemdorffer, J. M. Newcomer, D. Rubine, and D. B. Anderson, "Tactus: toolkit-level support for synchronized interactive multimedia," *Multimedia Syst.*, vol. 1, no. 2, pp. 77–86, 1993.
[22] D. L. Drucker and M. D. Murie, *QuickTime Handbook*. Hayden, 1992.
[23] L. Ehley, B. Furth, and M. Ilyas, "Evaluation of multimedia synchronization techniques," in *Proc. Int. Conf. Multimedia Computing and Systems*, 1994, pp. 110–119.
[24] J. Escobar, D. Deutsch, and C. Patridge, "Flow synchronization protocol," in *Proc. IEEE Globecom*, 1992, pp. 1381–1387.
[25] D. Ferrari, "Design and application of a delay jitter control scheme for packet-switching internetworks," in *Proc. 2nd Int. Workshop on Network and Operating System Support for Digital Audio and Video*, 1992, pp. 72–83.
[26] S. Gibbs, C. Breiteneder, and D. Tsichritzis, "Data modeling of time-based media," in *Visual Objects*. Geneve, Switzerland: Université de Genéve, Centre Universitaire d'Informatique, 1993, pp. 1–21.
[27] G. Grassel, "Object-oriented design and implementation of a MHEG runtime environment for the interactive presentation of multimedia documents in a distributed environment," (in German), Master's thesis, University of Mannheim, Mar. 1994.
[28] C. Hamblin, "Instants and intervals," in *Proc. 1st Conf. Int. Society for the Study of Time*, 1972, pp. 324–331.
[29] R. G. Herrtwich, "An architecture for multimedia data stream handling and its implication for multimedia transport service interfaces," in *3rd IEEE Workshop on Future Trends of Distributed Computing Systems*, 1992.
[30] R. Steinmetz and K. Nahrstedt, *Fundamentals in Multimedia Computing and Communications*. Englewood Cliffs, NJ: Prentice-Hall, 1995.
[31] R. Hunter, P. Kaijser, and F. Nielsen, "ODA: A document architecture for open systems," *Computer Commun.*, vol. 12, pp. 69–79, Apr. 1989.
[32] M. E. Hodges, R. M. Sasnett, and M. S. Ackerman, "Athena muse: A construction set for multi-media applications," *IEEE Software*, pp. 37–43, Jan. 1989.
[33] IBM Corporation, *Audio Visual Connection User's Guide and Authoring Language Reference, Version 1.05, IBM Form S15F-7134-02*, Aug. 1990.
[34] IBM Corporation, *ActionMedia/II—Technical Reference, Version 1.0*, 1992.
[35] IBM Corporation, *IBM Multimedia Presentation Manager Programming Reference and Programming Guide 1.0, IBM Form: S41G-2919-00 and S41G-2920-00*, Mar. 1992.
[36] ISO/IEC, "Hypermedia/time-based document structuring language (HyTime)," International Standard ISO/IEC IS10744, 1992.
[37] ISO/IEC, "Draft Recommendation X.903: Basic Reference Model of the Open Distributed Processing," N7055, ISO/IEC JTC1/SC21, 1992.
[38] ISO/IEC, "Information technology—coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s," International Standard ISO/IEC IS11172, 1993
[39] T. D. C. Little and A. Ghafoor, "Synchronization and storage models for multimedia objects," *IEEE J. Select. Areas Commun.*, vol. 8, pp. 413–426, Apr. 1990.
[40] ——, "Network considerations for distributed multimedia objects composition and communication," *IEEE Network Magazine*, vol. 4, pp. 32–49, Nov. 1990.
[41] ——, "Spatio-temporal composition of distributed multimedia objects for value added networks," *IEEE Computer*, vol. 24, pp. 42–50, Oct. 1991.
[42] ——, "Multimedia synchronization protocols for broadband integrated services," *IEEE J. Select. Areas Commun.*, vol. 8, pp. 1368–1382, Dec. 1991.
[43] ——, "Scheduling of bandwidth-constrained multimedia traffic," *Computer Communications*, vol. 15, pp. 381–387, July 1992.
[44] T. D. C. Little, "Protocols for bandwidth-constrained multimedia-traffic," in *Proc. 4th IEEE ComSoc Int. Workshop on Multimedia Communications*, 1992, pp. 150–159.
[45] ——, "A framework for synchronous delivery of time-dependent multimedia data," *Multimedia Syst.*, vol. 1, no. 2, pp. 87–94, 1993.
[46] L. Li, A. Karmouch, and N. Georganas, "Multimedia teleorchestra with independent sources: Part 1—temporal modeling of collaborative multimedia scenarios," *Multimedia Syst.*, vol. 1, no. 4, pp. 143–153, 1994.
[47] ——, "Multimedia teleorchestra with independent sources: Part 2—synchronization algorithms," *Multimedia Syst.*, vol. 1, no. 4, pp. 154–165, 1994.
[48] L. Li, L. Lamont, A. Karmouch, and N. Georganas, "A distributed synchronization control scheme in a group-oriented conferencing systems," in *Proc. 2nd Int. Conf., Broadband Islands, Athens, Greece*, 1993.
[49] L. Ludwig, N. Pincever, and M. Cohen, "Extending the notion of a window system to audio," *IEEE Computer*, vol. 23, pp. 66–72, Aug. 1988.
[50] B. D. Markey, "Emerging hypermedia standards—hypermedia marketplace prepares for HyTime and MHEG," in *Proc. USENIX-Conf. Multimedia—For Now and the Future*, 1991, pp. 59–74.
[51] T. Meyer, W. Effelsberg, and R. Steinmetz, "A taxonomy on multimedia synchronization," in *Proc. 4th Int. Workshop on Future Trends in Distributed Computing Systems*, 1993.
[52] ISO Multimedia and Hypermedia Information Coding Expert Group, ISO/IEC JTC1/ SC29/WG12, "Information technology—coded representation of multimedia and hypermedia information (MHEG), part 1: Base notation (ASN.1), committee draft ISO/ IEC CD 13522-1," June 1993.
[53] Microsoft Corporation, *Microsoft Windows Multimedia Authoring and Tools Guide, Microsoft Windows, Programmer's Reference Library*, Microsoft, Redmond, WA, 1991.
[54] D. L. Mills, "Internet time synchronization: The network time protocol," *IEEE Trans. Commun.*, vol. 38, no. 10, pp. 1482–1493, Oct. 1991.
[55] ——,"Precision synchronization of computer network clocks," *ACM Computer Commun. Rev.*, vol. 24, no. 2, pp. 28–43, Apr. 1983.
[56] A. Mauthe, W. Schulz, and R. Steinmetz, "Inside the heidelberg multimedia operating system support: Real–time processing of continuous media in OS/2," IBM European Networking Center, Heidelberg, Germany, Tech. Rep. 43.9214, 1992.
[57] K. Nahrstedt and R. Steinmetz, "Resource management in multimedia networked systems," Distributed Systems LAB79, University of Pennsylvania, Tech. Rep. MS-CIS-94-29, 1994.
[58] C. Nicolaou, "An architecture for real-time multimedia communication systems," *IEEE J. Select. Areas Commun.*, vol. 8, pp. 391–400, Apr. 1990.
[59] S. Oikawa and H. Tokuda, "User-level real-time threads: An approach toward high performance multimedia threads," in *Proc. 4th Int. Workshop on Network and Operating System Support for Digital Audio and Video*, 1993, pp. 61–71.
[60] L. Press, "Computer or teleputer?" *Commun. ACM*, vol. 33, pp. 29–36, Sept. 1990.
[61] C. Parris, H. Zhang, and D. Ferrari, "Dynamic management of guaranteed-performance multimedia connections," *Multimedia Syst.*, vol. 1, no. 6, 1994.
[62] S. Ramanathan and V. Rangan, "Feedback techniques for intra-media continuity and inter-media synchronization in distributed multimedia systems," *Computer J.*, vol. 36, no. 1, pp. 19–31, 1993.
[63] D. Shepherd and M. Salmony, "Extending OSI to support synchronisation required by multimedia applications," *Computer Commun.*, vol. 13, pp. 399–406, Sept. 1990.
[64] J.-B. Stefani, L. Hazard, and F. Horn, "Computational model for distributed multimedia applications based on a synchronous programming language," *Computer Commun.*, vol. 15, pp. 114–128, Mar. 1992.
[65] J. M. Smith, "Standard generalized markup language and related standards," *Computer Commun.*, vol. 12, pp. 80–83, Apr. 1989.
[66] R. Steinmetz, H. Schmutz, B. Schoner, and M. Wasmund, "Generic support for distributed multimedia applications," IBM European Networking Center, Heidelberg, Germany, Tech. Rep. 43.8910, Nov. 17 1989.
[67] R. Steinmetz, "Synchronization properties in multimedia systems," *IEEE J. Select. Areas Commun.*, vol. 8, pp. 401–412, Apr. 1990.

[68] ——, *Multimedia-Technology: Fundamentals.* (in German) Berlin: Springer-Verlag, 1993.
[69] ——, "Human perception of jitter and media synchronization," *IEEE J. Select. Areas Commun.*, this issue, pp. 61–72.
[70] R. Steinmetz and T. Meyer, "Multimedia synchronization techniques: Experiences based on different system structures," in *Proc. 4th IEEE ComSoc Int. Workshop on Multimedia Communications*, 1992, pp. 306–314.
[71] R. Terek and J. Pasquale, "Experiences with audio conferencing using the X Window System, UNIX, and TCP/IP," in *Proc. USENIX-Conf. Multimedia—For Now and the Future*, 1991, pp. 405–418.
[72] D. Tsichritzis, S. Gibbs, and L. Dami, "Active media," in *Object Composition,* D. Tsichritzis, Ed. Genéve, Switzerland: Université de Genéve, Centre Universitaire d'Informatique, 1991, pp. 115–132.
[73] C. Vogt, R. Herrtwich, and R. Nagarajan, "Heirat: The Heidelberg resource administration technique design philosophy and goal," IBM European Networking Center, Tech. Rep. 43.9307, 1992.
[74] T. Wahl and K. Rothermel, "Representing time in multimedia systems," in *Proc. Int. Conf. Multimedia Computing and Systems*, 1994, pp. 538–543.

Gerold Blakowski received the Diploma and Doctorate in informatics (computer science), from the University of Karlsruhe, Germany, in 1989 and 1993, respectively.

From 1989 to 1993, he was a research assistant at the Institute for Telematics, University of Karlsruhe, Germany, working in the area of distributed multimedia applications. Since 1993, he has been a visiting scientist at the Multimedia Technology Center of the IBM European Networking Center. His research interests currently include multimedia, distributed applications, and object-oriented technology.

Ralf P. Steinmetz (S'83–M'86–SM'93), for a photograph and biography, see p. 4 of this issue.