[StMe92a] *Ralf Steinmetz, Thomas Meyer*; **Modelling Distributed Multimedia Applications;** International Workshop on Advanced Communications and Applications for High Speed Networks (IWACA), S.337-349, München, März 1992.

Int. Workshop on Adv. Comm. and Appl. for High Speed Networks, March 16-19, 1992, Munich, Germany

Modelling Distributed Multimedia Applications

Ralf Steinmetz[‡], *Thomas Meyer*[†]

[‡]IBM European Networking Center,
Tiergartenstr. 8, 6900 Heidelberg, P.O.Box 10 30 68, Germany
Tel: +49 6221 404280, FAX: +49 6221 404450
e-mail: steinmet at dhdibm1.bitnet

[†]Lehrstuhl für Praktische Informatik IV
University of Mannheim
Seminargeb. A5, 6800 Mannheim, Germany
e-mail: mey at dhdibm1.bitnet

Abstract

It is often believed that with the mere knowledge of the kind of media, *e.g.* "speech", the system and specifically the communication requirements of a distributed multimedia application are automatically fixed. In general this assumption is wrong. As a matter of fact, certain technology aspects including the kind of media, coding, and compression techniques are of primary relevance. However the specific application's functionality imposes further requirements.

In this paper we present a new model for distributed multimedia applications which is used for the description of such applications. It allows for the retrieval of some communication requirements, and it can be used to structure the described application. This model was created within and used for the development of an audio/video distribution service as application of the HeiTS (IBM ENC Heidelberg High-Speed Transport System) project in Heidelberg.

1. Introduction

Today digital networks such as ISDN which integrate different services are available. However applications with added values are still rare. The reason is not just a mere mistake or misunderstanding of the users' need, but from a network architecture layered point of view, technology at lower levels seems to be ahead: The data rate of networks is larger than cost-effective adapter technology can consume as a whole. In [29] even LAN and MAN classes of several gigabites/s up to terabites/s are anticipated. The situation is even more dramatic at higher software layers where some services may be executed by the main CPU of the workstation. On the other hand operating systems are still not well-suited for audio and video data transfer [16]. To overcome this well-known application dilemma several company specific and externally funded projects especially within ESPRIT, RACE and DELTA currently deal with applications for high-speed networks.

A frequently raised question is the following: What is the use of "high-speed"? Our answer is that in addition to a fast file transfer, the integrated communication of audio and video is provided as the added value (see [32]). We understand multimedia according to the following definition [16; 35; 39]: A multimedia system is characterized by a computer-controlled, integrated processing, storage, presentation, generation, manipulation, and communication of independent information of time dependent (continuous) and time independent (discrete) media.

Today's interest of system architects and developers in multimedia applications and interfaces is twofold:

- Application developers want to abstract from the available physical dependencies and the multitude of multimedia devices [38]. Key issues are the treatment of audio and video in existing and novel environments, new user interface paradigms, and the description of distributed multimedia applications.
- From the description of the application, system — and communication system — developers want to retrieve the requirements for the underlying components. In a detailed analysis of application scenarios and classification schemes such as [3; 6; 15; 31] we found out that services are enumerated and communication requirements extracted. It is a fact that the telephone service requires 64 kbps, a one way end-to-end delay of 600 ms and isochronous data transfer mode. Does the same apply, *e.g.*, for a telephone with variable bit coding as part of a joint editing session? Posing such a kind of question, we felt that some parameters heavily depend on technology aspects and others on the application's context.

With this work we want to present a first step towards a solution for both above mentioned areas: The model describes applications, some of their communication requirements, and allows for the retrieval of some communication requirements, and can be used to structure the described application using object-oriented techniques. This model was created within and used for the development of an audio/video distribution service as application of the HeiTS (IBM ENC Heidelberg High-Speed Transport System) project [17; 14].

Previous work showed the dependency of the communication requirements from the kind of media [40], CCITT service class [1], and from applications within different service classes [42]. Our work can be regarded as a follow-on.

In the following section we present a short discussion about media and the state of the art concerning the derivation of communication requirements. We developed a model with five levels of abstractions which are subsequently described. In a final section, some aspects of our actual implementation are presented.

2. Some Communication Requirements Depend on the Application's Context

In this section we focus on some communication oriented system requirements: Within the **traffic mode** we distinguish the isochronous, the synchronous and the asynchronous mode. Each packet communicated using the isochronous traffic mode must be delivered within a well defined time interval. The synchronous mode assures packet delivery before a specified end-to-end delay. The asynchronous mode may only assure an average throughput for many packets. In order to cover multimedia aspects, we address the communication parameters going beyond the OSI specifications. Particularly the **throughput** in terms of a guarantee is of central importance for audio and video. Some application rely also on a guaranteed **end-to-end delay** (transit or round-trip) and request certain error handling (**reliability**) [12]. For multimedia additionally **synchronization** of various data streams is important and must be taken into account [22; 34; 37]. The consideration of various data streams together imposes also new requirements towards call managements [15]. High-speed networks open the potential for "computer-supported cooperative work" applications applying multicast techniques at different levels [10; 30].

Let media be the following set:

$$M := \{text,\ graphic,\ image,\ video,\ audio,\ ...\}$$

At certain levels of abstraction media appears as a combination of other kinds of media, TV-data is a combined audio/video data stream.
With respect to the time domain, multimedia systems deal with two categories of media [16; 35; 39]:

- **Discrete** or **time independent media** such as text, graphic and image are cosidered to be the traditionally known media from the computer science perspective. Processing of discrete media has no real-time demands. In general, this kind of media leads to the use of reliable transmission and a certain average throughput.
- **Continuous** or **time dependent media** like audio and video include real-time requirements. An audio packet arriving too late is of no use in a conversation supported by a telephone application and can be discarded. Other time-critical data relates to, *e.g.*, a shared pointer in a workstation conference. The main communication requirements cover guarantees of throughput and end-to-end delay. For uncompressed continuous media communication often errors may be tolerated.

In a first approximation it is useful to derive some values for the different communication requirements as described above, done *e.g.* in [18; 40] and shown in Figure 1. In [20] the pure media-oriented view is mixed with some services distinguishing, *e.g.*, audio, voice, electronic mail and facsimile.

	Voice	Video		Data
		Uncompressed	Compressed	
Information rate	16~64Kb/s	~100Mb/s	~1.5Mb/s	0.2~10Mb/s
Packet delay	~250ms	~250ms	~250ms	~1s
Fluctuation of packet delay	10ms-order	10ms-order	1ms-order	1ms-order
Packet loss	10^{-2}	10^{-2}	10^{-11}	10^{-11}

Figure 1. First approximation: Some communication requirements in relation with the medium [40]

With the knowledge of the medium and its requirements at the level discussed so far, it is often mentioned that audio or video in a conversational service necessarily requires isochronous data transfer mode. In general this is wrong! In order to avoid glitches (in the case of audio) a maximal transit end-to-end delay (from source device to sink device) must not be exceeded. If data arrives too early and sufficient buffering space is available, a constant data delivery rate at the sink can be guaranteed. Isochronous communication reduces buffer requirements and therefore is cost effective especially for high-quality video signals and longer end-to-end delays. Resuming, the synchronous mode is required, whereas an isochronous mode is helpful and often convenient.

As discussed in [1; 2] such a first approximation does often not apply to all applications to the same extend. With respect to the tolerable end-to-end delay, *e.g.*, it is completely different if voice is used in a conversational or retrieval service. Applications can be grouped in categories according to the CCITT classes of service [6; 7] (see Figure 15). Similarities within this groups can be used to derive common requirements [1].

However applications within the same service class may have very different requirements such as the high-speed telefax and a telephone service. The great diversity of CCITT services and application within the individual CCITT service classes leads to a great variability of communication requirements. Therefore three groups of performance criteria can be distinguished [42]: Delay sensitive, loss sensitive, and delay and loss sensitive. This classification does not cover the criteria we want to discuss and it is very coarse. A service has to be treated within its context; coding and compression has to be taken into account. High-speed networks capable of communicating multimedia information support the trend to more interactive distributed applications enabling capabilities of different service classes within the same application. Therefore it is difficult to classify applications such as distributed tutoring or joint editing according to the CCITT scheme.

In the following we present a model with five levels of abstraction which takes into account the application's context, the system structure, and the used technology. At the most abstract level, basic functions describing the generation and consumption of different kinds of information are identified. These functions are combined to types of functional units. At the next level kinds of media are related with them. An application combines several functional units and defines the context. At the placement process locations are associated. In the following section we will start with the description of the basic functions.

3. Identification of Basic Functions

For the analysis of applications and the derivation of their communication requirements at the most abstract level we have identified four disjoint *basic functions*: Two related to data generation/capture and the other two dealing with data delivery/presentation/storage.

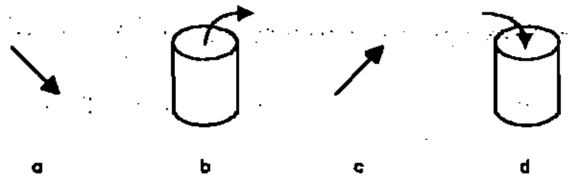


Figure 2. **Basic functions:** Persistent and nonpersistent information generation, presentation and storage

The information capture and/or generation shown in Figure 2a has the fundamental property of handling unique (not recoverable) information: This information can not be directly influenced by the application process with operations like "stop" or "replay". Typically this information is derived from the surrounding of the computer as it may be the voice "generated" by a video-conference user. But also processes within the computer generating deliberately unpredictable results such as within some simulation and animation processes belong to this category of **generation of nonpersistent information**. From the operating system point of view the information is read from a device like a camera, microphone, keyboard or process. The system reading the information must guarantee the required throughput for the data transfer. The flow control is determined by the process of nonpersistent information generation and the available buffer space. Even with additional knowledge about coding and compression techniques used, still both, synchronous and isochronous data transfer may be used as long as the peak data rate for time dependent media is assured. At this level of the model, the data delivery function is not determined, *i.e.*, the choice of synchronous or isochronous traffic mode depends on the buffering capabilities and real-time behavior of the system.

The second type of input data denotes information which is "predictable" as shown in Figure 2b. This information allows for operations like "stop/go" or "replay", because the process of data generation can be started again or the whole information is stored somewhere. Data processed and presented by retrieval systems such as videotex is typically stored on large file servers and comprise this basic function dealing with the **generation of persistent information**. In multimedia environments persistent information is often retrieved from optical memories.

Most applications include the **presentation** of persistent and nonpersistent information (see Figure 2c). This presentation covers the various media including audio and video. A video-telephone service has four presentation functions, for each user audio and video delivery respectively. As data presentation for audio and video must be "continuous" the information transfer towards or through the presentation component has to allow for a guaranteed throughput according to the quality, coding and compression agreed upon.

As shown in Figure 2d the fourth basic function relates to the **storage** of data. Only long-time data storage is considered and not any buffering within the data paths. If an application uses this function the primary requirement on the system is reliability. If the information is stored it may be used later on for further processing by other applications and should therefore be, in general, free of errors. Nonpersistent information is converted to persistent information by the process of storage.

All basic functions can, but must not, be associated with specific devices such as cameras or disks: We do not operate on device level. The functions describe the behavior and the characteristics of the information generation and consumption. Persistent p and nonpersistent $\neg p$ information generations are the **inputs** I whereas presentation and storage are the **outputs** O of the information flow:

$$I := \{p, \neg p\} \qquad O := \{pr, st\}$$

All four basic functions are media specific after the process of composition to functional units. If many different kinds of media or various distinct functions with the same **kind of media are part** of the application, the same basic function appears many times.

At this lowest level of the model we are not able to identify any type of requirements with respect to, *e.g.*, the end-to-end delay of the information. This requirement does not only depend on the media and basic function, additional knowledge is required.

4. Composition of Functional Units

Using the previously introduced basic functions now we are able to combine them towards **functional units** *Fu*. Such a functional unit comprises exactly one source and exactly one sink:[1] Sources may consist of one or two types of input, while sinks have one or two types of output. The composition of basic functions to functional units introduces a data flow of the medium/media associated with these functions. If an application requires many similar basic functions or a bidirectional flow of information, at the configuration and placement process (see the following sections) the application is modelled by assembling various functional units. This also means that various devices may be involved.

With the functional units a set of applicable *operation* are associated. In the model these operations apply to the functional unit as a whole. In our prototype of a distributed multimedia application we use an object-oriented approach. Unit types are implemented as classes, the corresponding operations are the class specific methods.

Functional Units with a Single Input and a Single Output Function

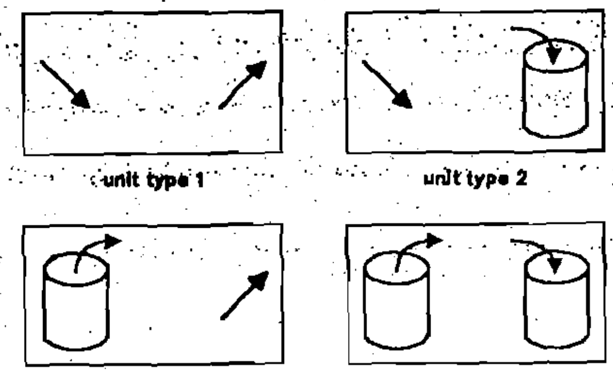


Figure 3. Elementary functional units: $ut1 := (\{\neg p\},\{pr\})$; $ut2 := (\{\neg p\},\{st\})$; $ut3 := (\{p\},\{pr\})$; $ut4 := (\{p\},\{st\})$

As shown in Figure 3 the **functional unit type 1** represents the process of capture and presentation of nonpersistent information. A typical example is a person-to-person audio information transfer application where the speaker talks at the source into a microphone generating information in real time while a second person listens at the sink. If continuous media is involved in the information flow within this unit type 1 then throughput guarantees are necessary, synchronous or isochronous mode is required. For discrete media without any relationship to other information (synchronization) the asynchronous traffic mode is usually convenient. Mandatory operations for this unit type are "activate" (initiates the actual data transfer) and "deactivate".

Within the **functional unit type 2** nonpersistent information is "converted" to persistent stored information. The user usually tolerates no or only very few errors in stored information (in contrast with the presentation of such information). The only requirement is that the system must be able to capture, transmit and store the information as fast as it is demanded by the information capture process. This highly depends on the media: It is, *e.g.*, easier to store the strokes of a keyboard than to capture, digitize and store video in HDTV quality.

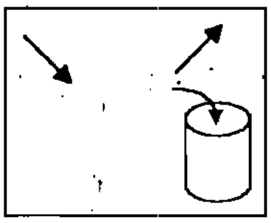
The **functional unit type 3** in Figure 3 can typically be used for the modeling of a retrieval service like videotex where an user located at the sink may interactively control the source. The control may explicitly be modelled by operations such as "start", "stop" and "rewind". As discussed at the previous functional unit type, the main requirement is to extract, transmit and present information as fast as it is required by the presentation process. In the case of continuous media the involved devices and communication paths must provide real-time capabilities.

The **functional unit type 4** in Figure 3 describes the typical part of a file transfer application. Communication requirements are primarily independent from the kind of media, there are no requirements related to delay or throughput. In most cases an application requires a file transfer for error free transmission of information. The preferred traffic mode is the asynchronous mode.

These four previously introduced functional units are "elementary". They determine the essential system and communication requirements for the remaining five possible functional units.

Functional Units with a Single Input and Two Output Functions

1 According to the German translations of "source" we choose the abbreviation Q (*Quelle*) and for the "sink" we use S (*Senke*).



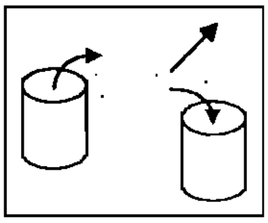


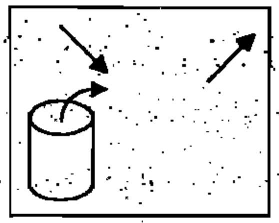
Figure 4. : $ut5 := (\{\neg p\},\{pr, st\})$; $ut6 := (\{p\},\{pr, st\})$

As shown in Figure 4 information presentation and storage is combined at the sink. These processes may occur simultaneously or sequentially.

Consider an application of listening radio and recording music on demand which is shown as **functional unit type 5**. Reliability is needed because of the storage function, real-time characteristics are required due to the continuous media presentation function.
The following formula does express that sources q and sinks s of the functional unit types 1 $ut1$ and 2 $ut2$ are the components of a unit type 5 $ut5$:
$ut5 := (ut1.q \cup ut2.q, ut1.s \cup ut2.s)$

Functional unit type 6 combines the functions of unit type 3 and 4 (see Figure 4). The communication requirement of unit type 3 are stronger than those of unit 4, *i.e.*, for persistent information generation, presentation of information is dominant compared with storage: Unit type 6 has the same requirements as unit type 3.
$ut6 := (ut3.q \cup ut4.q, ut3.s \cup ut4.s)$

Functional Units with Two Input and One Output Function



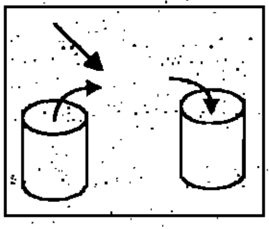


Figure 5. : $ut7 := (\{p, \neg p\},\{pr\})$; $ut8 := (\{p, \neg p\},\{st\})$

Figure 5 shows the functional units type 7 and 8 where persistent and nonpersistent information is generated together.

An example of the **functional unit type 7** is television broadcast. The source may be a news speaker commenting an audio/video sequence which is simultaneously or afterwards sent. At the sink the information is presented as one integrated data stream. At TV on demand a user may request certain video sequence from the persistent information generation function. The respective communication requirements are those from the functional unit type 1 and 2 together.
$ut7 := (ut1.q \cup ut3.q, ut1.s \cup ut3.s)$

The receiving side of a message handling system is an example for a **functional unit type 8** as shown in Figure 5. Apart from reliability the communication requirements are determined mainly by the functional unit type 3.
$ut8 := (ut2.q \cup ut4.q, ut2.s \cup ut4.s)$

Functional Unit with Two Input and Two Output Functions

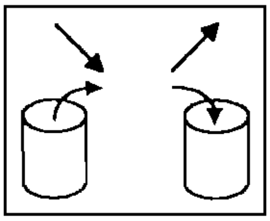


Figure 6. : $ut9 := (\{p, \neg p\},\{pr, st\})$

Figure 6 shows the **functional unit type 9** which can be used to describe a very simple text editing scenario. At the source the keystrokes represent the nonpersistent input and the document the persistent data. The display is the presentation basic function and the storage is the document itself:

$$ut9 := \binom{ut1.q \cup ut2.q \cup ut3.q \cup ut4.q,}{ut1.s \cup ut2.s \cup ut3.s \cup ut4.s}$$

5. Adherence of Media to Functional Units

At this level of abstraction, we bind functional units to the respective kind of media. Together with the kind of media additional information on the coding and compression used are added leading to various properties of the transferred data:

1. The first aspect relates to the **time intervals between** available valid information. We distinguish the following categories based on the moment where valid information becomes available:
   - If the time interval between adjacent events where the information becomes valid is constant with respect to jitter constraints, the delivery of the continuous media will be called *strongly periodic*. An example is PCM coded voice in a telephone system.
   - If the time intervals between adjacent events where the information becomes valid is given by a periodic function, the delivery of the continuous media will be called *weakly periodic*.
   - All other means of transfer for continuous media are called *aperiodic*. An example of an

aperiodic transfer is the coding of a moving pointer within a shared window: To transmit the current position and status of the buttons periodically blows up communication without much information transfer. The envisioned system will only transmit new information if the position or status has changed.

2. A second aspect refers to variations in the **amount of data** that becomes valid at these points in time.

   - If the amount of data within the transfer remains constant then the transfer will be called *strongly regular*. An example is the size of frames delivered by a camera or the sequence audio samples stored on a digital audio compact disc.
   - If the amount of data varies periodically in time then the transfer is called *weakly regular*. Some compression techniques for video make use of this approach: Larger packets are periodically devoted to a single compressed frame whereas smaller subsequent frames makes encode consecutive frame differences. Often the data packets are weakly periodic considering only their long-term mean data volume: According to the coding of moving pictures in MPEG I-frames are compressed single images whereas the compression of P- and B-frames takes into account sequences of images leading to smaller sizes [21]. A typical I:B:P ratio is 10:1:2. As no constant bit rate is delivered for any of those frames, only the long-term average can be described as weakly regular.
   - If the amount of data varies in time but not periodically then the data transfer is denoted as *irregular*. The processing of such data transfers is more sophisticated then the above mentioned alternatives.

For the estimation of the system requirements we use the notion of the respective functional units together with the kind of media.

Let us summarize:

- Nonpersistent information generation or presentation of time dependent media requires guaranteed throughput.
- Storage demands for reliability.
- In terms of throughput guarantees for discrete media, nonpersistent information "overrules" any persistent information requirements.
- Persistent information generation in combination with the presentation function and any additional basic function enables interactive operations between the sink and the source which influences the functional units (ut3, ut6, ut7, ut9) and needs to be modeled as a separate functional unit.

6. Configuration

From the applications' point of view the functional units are "building blocks" which describe at a very abstract level elementary functions. As multimedia applications usually operate with various media streams, we encompass the situation of combining various functional units in a **configuration** process.

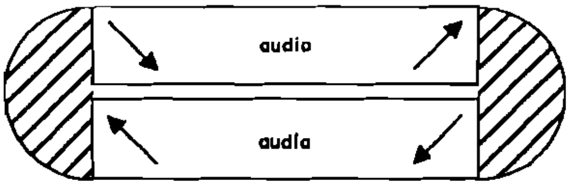


Figure 7. Example: Configuration of a telephone application

As a first example let us discuss the telephone service. Two functional units of type 1 are combined, namely the processes of nonpersistent information capture and presentation of the kind of medium "voice". Without additional information this combination allows for misleading interpretations such as an unidirectional data flow of an audio stereo signal. Therefore, it is additionally necessary to supply the knowledge of the **context** to the configuration process which in this special case relates the nonpersistent information capture of one functional unit to the presentation of the other respectively. The context is denoted in Figure 7 by the half circles. Such a context heavily depends upon the application itself and is not always determined by the system only. In terms of our example we are now able to sharpen the notion concerning the end-to-end delay. Since a person with his "processing" capabilities defines essentially this context, the question arises which one-way delay between listening and talking may be tolerated within a conversation. A value up to 600ms is acceptable [4] but lower values around 200ms are strongly encouraged [9; 41]. The essential feature of the context in the telephone application is the "audio loop" as the **temporal request-response behavior** of the user.

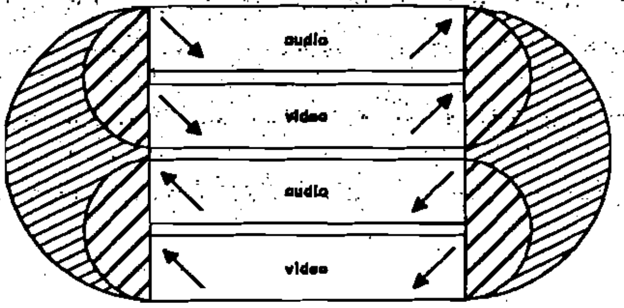


Figure 8. Example: Configuration of a video-phone application

In the case of a unidirectional stereo audio signal or combined audio and video information streams the contexts "ties" together both information streams. The video-phone configuration shown in Figure 8 demonstrates the situation of the respective context between audio and video. As the basic functions tied together are both of type input or output respectively the meaning of the context is in terms of a close temporal relationship, *i.e.* **synchronization**. The main feature of *live synchronization* is that synchronous input of data should result in synchronous output. Live synchronization should happen automatically; the application is not directly involved. Audio can played ahead of video for about 120 msec, and video can be displayed ahead of audio for about 240 msec. Both temporal skews will sometimes be noticed, but can easily be tolerated without any inconvenience by the user [24].

This live synchronization has to be distinguished from the case where data does not evolve on the fly, but is available by the basic function of persistent information generation [16]. In this case, which is called *synthetic synchronization* [22], users can freely order the various data entities in the time domain: The context defines a relationship between various basic storage functions. For this purpose, applications need to be able to express their synchronization requirements through corresponding language constructs. Notions like "present data entity A "simultaneously", "after", "independently" from data entity B" are needed [28]. These constructs apply either to the whole information entity or may refer to time or event stamps within the entity [37].
In general if two basic presentation functions are related by a context life synchronization occurs and if two storage functions a related synthetic synchronization takes place.

In the case of the video-phone application, the context additionally covers all basic functions at both ends of the video-phone application in Figure 8, which denotes the feedback characteristics of the user as discussed in the telephone example.

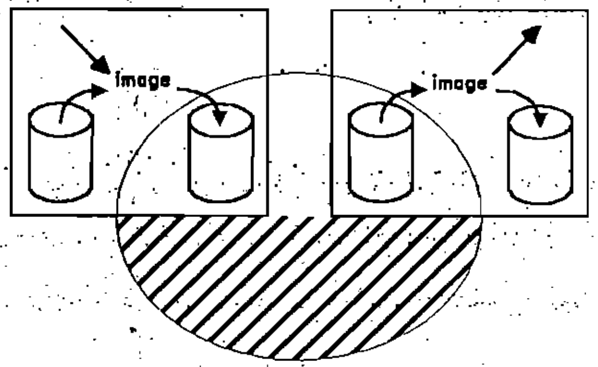


Figure 9. Example: Configuration of a messaging system application

The notion of the context is not restricted to nonpersistent information input and presentation, as shown in Figure 9 also the other types of basic functions may be involved. A message generated by the writer within a message handling process is deposited at the storage function drawn in the left functional unit. The receiver reads this message and acts according to the functional unit shown at the right. The whole message handling action is composed of both processes, where the storage and persistent information generation functions are the same, just separated in the time domain. This fact is shown by the context in Figure 9. The implication of this decoupling in the time domain relates to the end-to-end delay which is not critical any more.

As described in the previous examples, the context denotes the associations of functional units by relating

- several persistent information generation and/or several storage functions, or
- several nonpersistent information generation and/or several presentation functions.

At this level of the model we are able to analyze local and distributed application without any distinction. Especially with respect to future integrated multimedia system this approach makes sense: The transfer of audio and video data within a local system presents no problem as long as dedicated boards or/and a single-processing system is used. For a multi-processing system real-time requirements concerning the data transfer within the workstation arises in a similar way as video and audio is communicated over data networks. For sueh a data transfer IPC mechanisms are used. These mechanisms were not conceived and developed for the communication of time dependent media. It turns out that todays' IPC mechanisms are not appropriate!

7. Placement

In order to identify where communication over networks arise it is necessary to introduce the knowledge of locations of functions within a distributed system. This process of binding basic functions to locations is called **placement**. By the placement the sources and sinks of an applications has to be related with an element out of the possible set of locations.

Note, for our discussion aspects like late binding and object migration are considered implicitly by the knowledge of the application behavior. A model for such applications covers either all potential configurations and placements of the basic functions or a partial view at a certain moment.

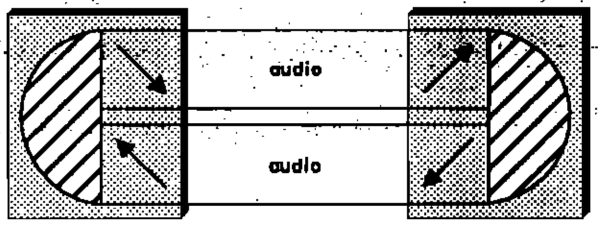


Figure 10. Conversational service: Placement of a telephone application

A conversational scrvice such as the telephone comprises at least two different users located at different places (see Figure 10). In the placement process we distinguish between functions with users located at the same place and a distributed systems with involve communication networks. This communication over networks is shown by the "empty space" between input and output functions within the functional units.

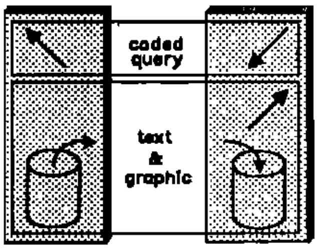


Figure 11. Retrieval service: Placement of a videotex application

The retrieval service is modelled by using the functional units of type 6 and 1 shown in Figure 4. The basic function of persistent information generation is located at a central server, whereas the initiator of the retrieval process is placed somewhere else. In Figure 11 these location describing attributes are denoted by the boxes drawn around the basic functions. It depends on the point of view if just one or many users should be considered. The interaction of the user with the central server is modeled as the functional unit type 1. Todays' distribution services allow for very few interactions with the user. High-speed networks providing short response times encourage applications to be more interactive such as the navigation through hypermedia documents [25; 26].

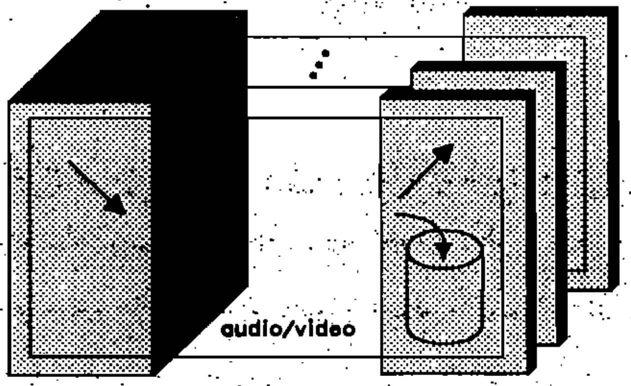


Figure 12. Distribution service: Placement of a **broadcasting** application

TV as a distribution service is sketched in Figure 12 where the various locations are denoted by different boxes. This communication structure of type broadcast (addressing all potential users) has separate requirements than multicast connections (addressing all users of a subgroup) and unicast connections [5]. A fundamental aspect of protocols for multicast relates to the the reply semantics [8; 19]. It has to be distinguished if the application does not want any response (which is the case in broadcast), or it expects one, some or replies of all member of the addressed multicast group members. Having identified a multicast communication structure the immediate questions concerns the reply semantics which can not be answered completely without additional knowledge about the application: Consider a full-motion video, multicast communication which is recorded at one station for further evaluation. If the information is highly compressed and application requires post processing of the data, reliability is highly encouraged. At least 1-reliability is needed, which expresses that the answer of the recording station must be returned. On the opposed side, *e.g.*, the recording of an uncompressed video signal of a talk does not impose severe reliability requirements and 0-reliability should be used. Multicast is not only an issue of networks, it has the same functionality in local systems and may therefore also be discussed as part of the configuration process as well.

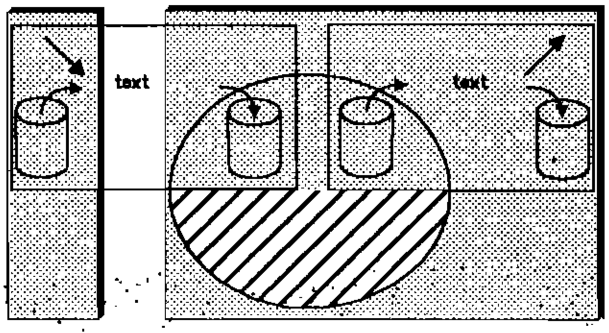


Figure 13. Messaging service: Placement of a text messaging application

Within a typical messaging system a note is generated at the sender and transmitted to the receiver. There the message is extracted from the mailbox, presented, stored and involved in further processing. By the placement the sender and the receiver are mapped onto the individual processes shown in Figure 9 leading to Figure 13.

After the placement, system internal communication can be distinguished from communication over networks. Nevertheless local and remote communication have the same or at least similar communication requirements to be guaranteed for multimedia applications.

8. Case Study: Implementation of an Audio/Video Distribution Application

Instead of discussing a complex application scenario showing the whole capabilities of our model, in the following we describe the use of this model for the HeiTS project. In HeiTS a prototype for an **audio/video distribution application** was developed which is currently being enhanced.

We successfully applied our model to the following areas:

1. The audio/video distribution application will be installed in a field trial demonstrating the effectiveness of the dissemination of information applying multimedia technology. This field trial takes place at an IBM branch office starting in April'92. It will help

us to learn more on the users' needs of such distributed multimedia applications. Together with some of the future users, we enhanced and changed the basic user relevant **design** issues of the application: Questions like "is there a central information-distribution-schedule" or, "should a user have the possibility to modify the schedule of a video clip transmission" were discussed using our model.

2. The **implementation** is based on a stream-handling run-time system allowing for the usage of continuous data streams (e.g. audio/video) within the application at various levels of abstraction. We choose the object-oriented approach to build a class hierarchy based on the model described in this paper.

Let us now elaborate on the relationship of the "design" and "implementation" to our model.

Design

At a first glance, an audio/video distribution application looks like a very simple application. A server station sends audio and video information to a group of remote listening clients. Such an application might immediately be compared with TV. The content as well as the distribution time is defined by a centralized party. Each viewer can decide for himself/herself if he/she wants to receive the distributed information, but he cannot influence the sending program. In terms of our model, we have a *functional unit type 1* from one server to many clients. The *application's context* between all clients describes that all clients receive the information at the same time. Using computer networks this might be very difficult to achieve, because the clients could be spread over several sub-networks.

The following question arises: "Why should anyone want to have a TV-application on computer systems?" One reason is to make efficient use of one integrated network technology for both, computer data and audio/video data. As another reason we believe, computer and multimedia stands for interactive communication between the user and the multimedia system. Thus the distribution service will evolve to an *information on demand* service. Distribution no longer means transmission but, centralized storage of information. In this scenario each client-server communication is a *functional unit type 3*. Furthermore, there exists no *application's context* between the clients. In theory the clients are not limited in subscribing a multitude of video clips from the server at the same time. Technically there are limitations due to a shortage of resources (e.g. simultaneous storage access, communication bandwidth).

Our application is located somewhere in between these two contrasting scenarios. We want to merge the following requirement: A central party is allowed to distribute information to a group of viewers (the management want to give a statement to their employees) and, the clients want to subscribe individually chosen information from a database. Using the model, the scenario looks as follows: If the server sends information, the *functional unit type 1* is valid for each client. This means that the client can decide to establish the connection (we say "it joins a group"). A viewer participating in a video distribution application wants to have information on the content (e.g. title, elapsing and remaining time). Therefore, he needs an additional *functional unit type 3* representing a transaction, where he can request this information. If no client listens and no time slot is reserved for a video distribution clip, then the client can select a clip out of the database. This is represented through a *functional unit type 3*. All clients which afterwards join the session can only take part at the current transmission, as it is. Again a *functional unit type 1* is used. The *application's context* between all active clients denotes, that they receive the same information, which is equivalent to send the information from the server to multiple targets at the same time. This does not guarantee the same receiving time at each client.

We also demonstrated with our model, that connection establishment can be expressed by different *functional unit types* in combination with the *application's context*. Currently we extend our transport system to support multiple target connections. Optimization is performed by exploiting multicast facilities of the LANs (IBM Token Ring) and our network layer protocol.

Implementation

The application runs on IBM Personal System/2 running the IBM Operating System/2 Version 1.3. The user interface as well as the stream-handling is implemented in C++. We make use of DVI technology: Our first prototype incorporated ActionMedia 750 and we are currently (November'91) integrating ActionMedia II.

The source code shown in this section demonstrates how the continuous data flow is implemented; any discussion on communication of discrete media is avoided (but is part of the prototype).

Figure 14 gives an overview of the stream-handling system used by an application. The stream-handling hides the data transfer between input- and output-devices from the application programmer. At the lower interface, the stream-handling is connected to the basic services. Currently they interface the OS/2 file system, the audio/video subsystem (ActionMedia II), and the transport system (HeiTS). Note, the basic services are provided by several libraries and device drivers which are used to implement source and sink objects (e.g. to read and to write digital, packetized data).

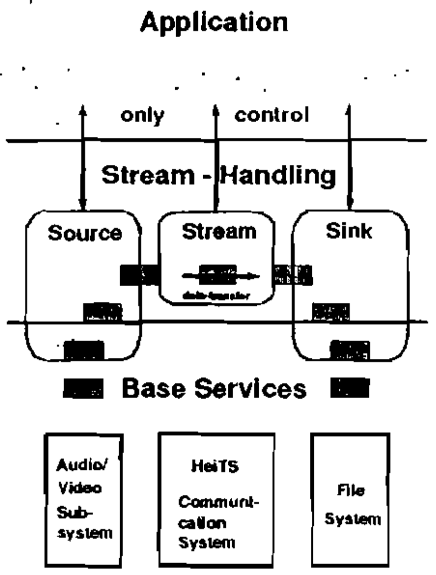


Figure 14. Stream-Handling: System structure

The following code fragment is an example on how an application invokes the stream-handling. Through this interface, the application is controlling streams, sources and sinks (connected to the streams) by invocation of the respective member functions.

Data packets do not "cross" the upper layer interface. Nevertheless, at the application's environment, it is possible to receive predefined messages. These messages are notifications (like communicating a time stamp to the application) or exceptions (allowing the application to react on errors occurring in a stream).

```
pStream   = new Stream()
pSrc      = new Source(pStream);
pSnk      = new Sink(pStream);
pSrc      ->Open();
pSnk      ->Open();
pSrc      ->Start();
pSnk      ->Start();


pSnk ->Stop();
pSrc ->Stop();
```

The stream-handling library itself supports a complete thread structure for continuous media streams. The following paradigm is used: A producer (source) writes buffers into a stream and a consumer (sink) reads buffers from a stream. A source and sink are interconnected, whenever they are writing and reading from the same stream. The definitions for the Source and Sink class are symmetric apart from: The member function Source::Read(Buffer *) and the member function Sink::Write(Buffer). In the corresponding base classes, the member functions Read and Write are not implemented (= 0) due to the following reason: The interfaces of these functions are demanded, but, media and device related aspects are implemented in the derived classes.

```
class Source
{
  protected:
    Stream  &Strm;

            void Thread ();

  public:
            Source      (Stream& s) {Strm = s;}

    virtual  SHRC Start  ();
    virtual  SHRC Stop   (void);
    virtual  SHRC Read   (Buffer *) = 0;
    virtual  SHRC Open   ()         = 0;
    virtual  SHRC Close  ()         = 0;
  }
```

A Connection between a source and a sink object located on different endsystems is also represented as sink at the producer and as source at the consumer. The class hierarchy give us the chance to model different input, output devices, and connections, having all the same interface for continuous data processing. Deriving new classes means to model the application specific requirements. A Connection class is also derived from a Source and Sink class. In our project we implement a special unidirectional and multiple target Connection class.

The following code fragment shows how a source-thread writes "produced data" into a stream.

```
void Source::Thread()
{
  while (!Stop)
    Strm << this;
  }
```

We do not distinguish between several sources and sinks using the *basic functions*. We directly take into consideration the composition of functional unit types, which also motivates the derivation of different stream classes.

A stream represents a pool of buffers where reading and writing is performed by a first-in-first-out strategy.

```
class Stream
{
  protected:
    Buffer    *QueueArray[];
    int       Widx, Ridx;
    Semaphore Sem;
    BOOL      Stop;

    virtual   void  Exception (Exp);

  public:
              Stream () {Widx = Ridx = 0;}

    virtual   void  operator << (Source *);
    virtual   void  operator >> (Sink *);
  }
```

The actual implementation only support streams connecting one source and one sink.[2] In order to implement streams representing functional unit types (5-9) with more than one source or one sink, we need to combine the streams into a group. A group maps several input streams into several output streams. Because source threads generate new buffers and the sink threads free them, a multiple copy operations takes place. Our Buffer Management System provides logical copy operations without copying the actual content of the buffers. Only multiple buffer descriptors are copied into the different output streams. A buffer is freed after the last descriptor is freed by a sink thread.
The group is also used to synchronize streams.

```
void Stream::operator << (Source *src)
{
  Buffer buf;
  Exp    exp;

  exp = src->Read(&buf);
  if (exp)
     Exception(exp);
  else
     QueueArray[Widx++] = buf;
  if (Widx == Ridx) SyncWait(Sem);
  }
```

Basic functions are the basis for *functional unit types* In the class hierarchy we consider them in creating a new stream and connection classes. The Source and Sink class help us supporting different devices, where media specific aspects are hidden by the Read/Write member functions. The Group represents many aspects of the *application context*. We do not explicitly distinguish between local and distributed applications. A local application connects a source device to a sink device using the same stream object.

The *placement* is expressed by connecting a Source object to a Connection-Sink object at the producer and, a corresponding Connection-Source is connected to the Sink object at the consumer.

In this implementation scope as a whole, our model is used to derive the class hierarchy of the application.

9. Conclusion

The description of applications at different levels allows for a classification of applications, hints for their implementation, and the derivation of some communication requirements associated with the individual levels of abstraction. In our model we distinguish:

- Level 1: **Basic functions** (four types),
- Level 2: **Functional units** (nine types),
- Level 3: **Adherence** of media
- Level 4: **Configuration** by additional context specification, and
- Level 5: **Placement**

As we have demonstrated, it is not possible to simply derive all media transfer requirements from the technology only, the application's context is essential: The toleration of errors within a voice stream depends *e.g.* heavily on the compression techniques applied. The tolerable end-to-end delay is mainly determined by the context of the application: E.g., in a one-way stereo playback of a concert larger delays are tolerable than in a telephone conversation.

From our first experiments in multimedia within the DiME project we learned that an object-oriented approach is very promising at the application programming interface [33] as well as for the upper-layer communication services [36]. The application prototype is being developed as a result of the presented model; it serves as demonstrator for HeiTS (our IBM ENC Heidelberg High-Speed Transport System).

In our model we assume that the kind of media is the same in the source and in the sink. [11; 13] demonstrates the necessity of media conversion. One example is the text to speech conversion for visually handicapped people [27]. Instead of binding the kind of media to the whole functional unit in the adherence process, we must allow for different kinds of media at the source and sink. We are currently enhancing the model towards media conversion and elaborating the implications of this supplement.

Analyzing the different communication services according to the CCITT classification with our model we encompass a nice correlation: Each type of service has very few different basic functions, *i.e.*, mostly two and sometimes three: A conversational service is built from functional units of type 1 and always has *e.g.* two different basic functions. Whereas most of local multimedia and especially hypermedia applications comprise all four basic functions. The reason for this is the cost and availability of appropriate communication systems. In LANs, applications also tend to comprise the four basic functions. In WANs the same behavior is anticipated: The evolution of high-speed multimedia applications reveals a convergence of different services towards integrated applications.

We would like to thank Bernd Schöner for initial discussions on the media dependency of communication requirements. With many valuable comments Dietmar Hehmann, Wolfgang Johannsen and Berthold Köhler contributed significantly to the quality of this paper.

[2] Assume, we want to synchronize the video and the audio data by interleaving the corresponding frames. All frames with the same time stamps are combined into the same buffer. For this application requirement it is sufficient to create only one source and one sink class which control the combined audio video data flow.

References

[1] *Heinrich Armbrüster.* **Applications of Future Broad-Band Services in the Office and Home.** IEEE Journal on Selected Areas in Communication, vol.4 no.4, Jul. 1986, pp. 429-437.

[2] *Heinrich Armbrüster.* **Weiterentwicklung der Telekommunikation: Universalnetz Breitband-ISDN.** ntz, vol.40, No.8, 1987, pp. 564-569.

[3] *Heinrich Armbrüster, Hans Jörg Rothamel;* **Breitbandanwendungen und -dienste: Qualitative und quantitative Anforderungen an künftige Netze;** ntz, vol. 43, no. 3, 1990, pp. 150-159.

[4] *Bell Telephone Labs.;* **Engineering and Operations in the Bell System;** AT&T Bell Labs, 1984.

[5] *K. P. Birman, T.A. Joseph;* **Exploiting Replication in Distributed Systems;** In: Distributed Systems, Editor: Sape Mullender, Addison-Wesley, New York, 1989, pp. 319-367.

[6] *The International Telegraph and Telephone Consultative Committee;* **Broadband Aspects of ISDN;** CCITT Study Group XVIII Recommendation I.121, Temporary Document 49, 1988.

[7] *International Telegraph and Telephone Consultative Committee;* **B-ISDN Service Aspects;** CCITT Study Group XVIII, Draft Recommendation I.211, Geneva, 23-25 May 1990.

[8] *David R. Cheriton, Willy Zwaenepoel;* **Distributed Process Groups in the V-Kernel;** ACM Transaction on Computing Systems, vol. 3 no. 2, May 1985, pp. 77-107.

[9] *Thomas M. Chen, Jean Walrand, David G. Messerschmitt;* **Dynamic Priority Protocols for Packet Voice;** IEEE Journal on Seleeted Areas in Communications, vol.7, no.5, Jun. 1989, pp. 632-643.

[10] *Clarence A. Ellis, Simon J. Gibbs, Gail L. Rein;* **Groupware: Some Issues and Experiences;** Communication of the ACM, vol.34, no.1, January 1991, pp.38-58.

[11] *Warren E. Falconer, John A. Hooke;* **Telecommunication services in the Next Decade.** Proceedings of the IEEE, vol.74, no.9, Sept. 1986, pp.1246-1261.

[12] *Domenico Ferrari;* **Client Requirements for Real-Time Communication Services;** International Computer Science Institute, Technical Report 90-007, Berkeley, March 1990.

[13] *G. D. Flinchbaugh, P. L. Martinez, D. S. Rouse;* **Network Capabiltities in Support of Multimedia Applications;** IEEE Global Telecommunications Conference & Exhibition (GLOBECOM), Conference Record vol. 1, San Diego, Dec. 2-5, 1990, pp. 322-326.

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

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

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

[17] *Dietmar Hehmann, Ralf Guido Herrtwich, Ralf Steinmetz;* **Creating HeiTS: Objectives of the Heidelberg High-Speed Transport System;** IBM ENC Technical Report no.43.9102, 1991, and F&E Projekt-Bericht, GI'91, Darmstadt, 14.10-18.10 1991.

[18] *Dietmar Hehmann, Michael Salmony, Heinrich Stüttgen;* **Transport Services for Multimedia Applications;** Proceedings of the IFIP WG 6.1/WG 6.4 Workshop on Protocols for High Speed Networks, North Holland, 1989, 303-321

[19] *Larry Hughes;* **Multicast Response Handling Taxonomy;** Computer Communications, vol. 12 No 1, Feb. 1989, pp. 39-46.

[20] *Jaghdish C. Kohli, Dar S. Biring, Gasper L. Raya;* **Emerging Broadband Packet-Switch Tecnology in Integrated Information Networks;** IEEE Network, Nov. 1988, vol.2 no.6, pp. 37-51.

[21] *Didier Le Gall;* **MPEG: A Video Compression Standard for Multimedia Applications;** Communications of the ACM, vol.34, no.4, April 1991, pp.46-58.

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

[23] *Thomas Meyer;* **Anwendungsszenarien für verteilte Multimedia-Systeme basierend auf Hochgeschwindigkeitsnetzen;** Diploma Thesis, University of Mannheim, 1991.

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

[25] *Jakob Nielsen;* **the Art of Navigating through Hypertext;** Comm. of the ACM vol. 33 nr. 3, 1990, pp. 298-310.

[26] *Jakob Nielsen;* **Hypertext and Hypermedia;** Academic Press, 1990.

[27] *Laura R. Pate;* **Trends in Multimedia Applications and the Network Models to Support them;** IEEE Global Telecommunications Conference & Exhibition (GLOBECOM), Conference Record vol. 1, San Diego, Dec. 2-5, 1990, pp. 317-321.

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

[29] *Izhak Rubin, Joseph E. Baker;* **Media Access Control for High-Speed Local Area and Metropolitan Area Communication Networks;** Proc. of the IEEE, vol.78, no. 1, January 1990.

[30] *Sunil Sarin, Irene Greif;* **Computer-Based Real-Time Conferencing Systems;** IEEE Computer, vol. 18, no. 10, Oct. 1985, pp. 33-45.

[31] *Michael Salmony.* **The Potential of Broadband ISDN.** Computer Standards & Interfaces 8, 1988, pp. 15-21.

[32] *Gerd Schürmann, Uwe Holzmann-Kaiser;* **Distributed Multimedia Information Handling and Proc-**

essing; IEEE Network Magazine, vol.4 no.6, Nov. 1990, pp. 23-31.

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

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

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

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

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

[38] *Ralf Steinmetz, Christian Fritzsche;* **Abstractions for Continuous-Media Programming;** 2nd International Workshop on Network and Operating System Support for Digital Audio and Video, Heidelberg, Nov. 18-19, 1991.

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

[40] *Yasukazu Terada;* **High Speed, Broadband Communications and OSI;** North-Holland, Computer Standards & Interfaces 7, 1988, pp. 23-28.

[41] *Po-Choi Wong, Tak-Shing Peter Yum;* **An Integrated Sevices Token-Controlled Ring Network;** IEEE Journal on Selected Areas in Communications, vol.7 no.5, Jun. 1989, pp. 670-679.

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

Appendix

Service Classes	Type of Information	Examples of broadband services	Characteristics
Conversational Services	Moving pictures	Video telephony, Video-conference, Video-surveillance, Video transmission service	Bidirectional dialogue communication with real-time end-to-end information transfer from user to user/host. The flow of the user information may be bidirectional symmetric, bidirectional asymmetric or in specific cases unidirectional
	Sound	Multiple sound-programme signals	
	Data	High speed unrestricted digital information transmission service, High volume file transfer service, High speed teleaction	
	Document	High speed Telefax, High resolution image communication service, Document communication service	
Messaging services	Moving pictures and sound	Video mail service	Offers user-to-user communication between individual users via storage units with store-and-foreward, mailbox and/or message handling functions.
	Document	Document mail service	
Retrieval services	Text, data, graphics, sound, still images, moving pictures	Broadband videotex, Video retrieval service, High resolution image retrieval service, Document retrieval service, Data retrieval service	The user can retrieve information stored in information centers and in general for public use on an individual basis. This information will be sent on his demand.
Distribution services without user individual presentation control	Video	Existing quality TV distribution, Extended quality TV distribution, High definition TV distribution, Pay-TV	These service include broadband services. They provide continuous flow of information which is distributed from a central source to an unlimited number of authorized receivers. The user cannot control the start and order of the presentation.
	Text, graphics, still images	Document distribution service	
	Data	High speed unrestricted digital information distribution service	
	Moving pictures and sound	Video information distribution service	
Distribution services with user individual presentation control	Text, graphics, sound, still images	Full channel broadcast videography	Information is provided as a sequence of information entities with cyclical repetition. The user has individual access, controls start and order of presentation.

Figure 15. Broadband Services: Short form of the CCITT Recommendations I.121 [6]