MULTIMEDIA: DATA STRUCTURES, DOCUMENTS

by

J-Y. Dauneau and P. Varaiya

Memorandum No. UCB/ERL M92/31

11 February 1992

MULTIMEDIA: DATA STRUCTURES, DOCUMENTS

by

J-Y. Dauneau and P. Varaiya

Memorandum No. UCB/ERL M92/31

11 February 1992

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

MultiMedia : Data Structures, Documents

J-Y. Dauneau, UCB[1] /ENSICA[2] P. Varaiya, UCB
M. Diaz, LAAS du CNRS[3] P. Senac, LAAS du CNRS/ENSICA

February 11, 1992

[1]University of California at Berkeley, Department of Computer Science, Berkeley, CA 94720.

[2]Ecole Nationale Superieure d'Ingenieurs de Constructions Aeronautiques, 1 Av. Leon Blum, 31056 Toulouse Cedex, France.

[3]Laboratoire d'Automatique et d'Analyse des Systemes, 7 Av. du Colonel Roche, 31077 Toulouse Cedex, France.

Contents

List of Figures

List of Tables

Introduction

With the availability of dedicated and more and more powerful architectures, it is now possible to access and manipulate high-quality analog and digitized video and sound signals through information systems. This ability definitely extends the range of potential applications to up to now closed to data processing fields. For example, when computers were introduced at school, the use of computer science for learning purposes has lead to austere and non efficient applications. The ultimate result is now a large number of "computerphobic" people now called upon to painfully integrate what should be natural aid to work in any sector. Multimedia must take the end user into account from the beginning as a major requirement.

Multimedia is itself a "multi-meaning" word depending on the context in which it is considered. Chapter 1 tries to make it clear about this composite term. Among the potential applications, few prototypes have been developed, some of which are presented in chapter 2 by focusing on their data structures. The wide range of multimedia applications implies numerous requirements in terms of data structures. Chapter 3 makes the synthesis of the most important of these requirements. After choosing an independent approach of these, we focus on synchronization problems that lead to the definition of a complex data structure in chapter 4. A model for multimedia distributed systems is first described and the notions involved in the definition itself are redefined for clarification. A basic justification is then presented.

Lastly, a brief survey of the Xwindow system is given in chapter 5 putting the emphasis on some aspects that make the currently used version[1] not fully adequate for multimedia.

[1] X11R4.

Chapter 1

What is Multimedia ?

1.1 Introduction

The word "multi-media" appeared in the sixties (in the spoken language) after the large diffusion of means of information (radio, television, advertisement) when, in western countries, one could rather freely get and use them. The term "mass-media" is the origin from which comes "media" and then multi-media and hyper-media. This chapter discusses what "multimedia" is or could be, and what a multimedia application deals with . Thus, it is totally informal.

As many authors, we could just have used the term "multimedia", but trying to define it, even if it seems to be a waste of time, seems a way to point out many aspects of future information systems.

1.2 Dictionary approach

Three sources are used[1]: the ISO Multimedia and Hypermedia Coding Expert Group [JTC90b], The Oxford English Dictionary [Oxf89] and the Robert French dictionary [Rob90] (current dictionary).

- **Medium :**

  1. "a means by which information is perceived, expressed and transmitted"
     (MHEG definition proposal)
  2. "5.a. An intermediate agency, means, instrument or channel. Also intermediation instrumentality: in phrase *by* or *through the medium* of. spec. of newspapers, radio, television, etc., as vehicles of mass communication. [...] (see *media*)."
     Media: "Newspapers, radio, television, etc., collectively, as as vehicles of mass communication. Freq. *attrib.* or as *adj.*
     (Oxford dictionary)
  3. "any massive support for information diffusion"
     (Robert dictionary)

  According to The Oxford dictionary, the word appears in the twenties, probably after *mass media.*

[1]This section is written for purists who anyway would have asked for it and can easily be skipped if you have ever heard about media.

- **Multimedia :**

  1. "the property of handling several types of representation media" (ISO - NB: this definition has not a real meaning without the following ISO definitions)
  2. "Designating or pertaining to a form of artistic, educational, or commercial communication in which more than one medium is used."
  3. "relative to several media" (current dictionary)

  For each of our sources, "multimedia" is an adjective and so should not be used alone. According to The Oxford dictionary, the word appears in the sixties.

1.3 Authors' definition

Scientific publications on multimedia are numerous, but in few of them the authors precisely define this term. Actually, the object of the study is usually implicit, but authors don't want to give restrictive definitions. Some definitions have been provided by the scientific literature, and by the ISO Multimedia and Hypermedia coding Experts Group (MHEG).

1.3.1 Scientific literature

Many authors define 'multimedia' as "any combination of text, graphics, sound and video so as to build an animation". This definition seems to be rather adequate currently because the type of information[2] listed is the one we commonly use. But it is easy to imagine that we don't know what kind of information will become available in the future. A new definition would be needed each time we want to handle something new.

Some authors give a general definition of multimedia and make it clearer by giving examples of multimedia applications. Thus, in [Naf90], 'multimedia' is defined as "a new buzzword describing different kinds of information represented in digital form". The fuzzy aspect remains in the terms "kind of information". In [MSS90], 'multimedia' is defined relatively to the user: "[...], the interchanged information must be available in a form receiptable to humans, or in a form suitable to automated processing. Information in

[2]We use the phrase "type of information" loosely here. This part appeals to the reader's intuition and, as stated in the introduction, is informal.

a form receiptable to humans is generally associated today within the term *multimedia*". In [KNS90], the definition of *multimedia* is given by its functional characteristics, i.e. "to store and retrieve documents composed of text, images, and audio".

Other authors don't try to define the word itself but something else it qualifies, such as *multimedia application*. In [HSS90], the latter "refers to computer network applications with widely varying data traffic characteristics". Sometimes, the phrase 'multimedia communications' is defined. Thus, in [GYN90] and [Ste90], this is the field referred to "the representation, storage, retrieval, and dissemination of machine-processable information expressed in multiple media, such as text, voice, graphics, images, audio, and video". This definition, which is the most complete found in the literature, takes the functional aspects as well as the typed character of the handled information into account.

Note that in many articles on the subject it is assumed that you know what *multimedia* means. In technical dictionaries, the notion is either ignored or worse defined than in any current dictionary. Eventually, one can say as in [She90] "there is a revolution afoot. It's called *multimedia* ..."

1.3.2 Some definitions

In this section, we try to define several types of media according to the physical nature of the underlying devices and to the entity operating on/with these media. Similar propositions can be found in [JTC90b].

1. Perception medium = what is perceived by the end user. This can relate to one or several of the five human senses.

   examples:
   - auditory perception : the user perceives noise, music, sound signals,...
   - visual perception : the user perceives text, drawings, moving scenes,...

   NB: this definition is subjective, as it takes non measurable criteria as a basis. Users won't call still images in the same way : they can be called graphics, images, pictures and so on.

2. Presentation medium = physical device used to display and/or acquire some information to /from the user.

examples:

- input: keyboard, mouse, microphone, camera, ...
- output: screen, printer, loudspeaker, ...

3. Representation medium = coded form of the information. This includes the stored form of the information.

examples:

- text : ASCII, EBCDIC, ...
- audio : MIDI, CCITT G.711, ...
- still pictures : Fax group 3, ...

4. Communication medium = physical device used for data transmission and/or storage.

examples:

transmission:

- twisted pair, coaxial cable, optical fiber, radio link, ...

storage:

- electronic memory, floppy disk, hard disk, optical disk, magnetic tape, ...

NB: - Similar definitions have been proposed by the ISO Multimedia and Hypermedia coding Experts Group. The distinction between storage and transmission media is made at the physical level. The "form of the information as it is stored" is included in the definition of "representation medium". For us, storage and transmission media should be disjoint from a certain level: transmission only deals with the lowest layers of protocols.

Many recent technical dictionaries have been consulted for that paper (seven of them). Only one proposed something else than "combination of text, graphics, sound and video". than giving an elaborated definition. It must be noticed that current dictionaries give better definitions than the expensive technical dictionaries.

Application	Object	Typical Size	Total
Geography	Multispectral Scan Image (corrected)	3,548 b × 2,983 b × 6 b/pixel	64 Mb
	Return-Beam Vidicon (corrected)	5,322 b × 5,322 b × 6 b/pixel	170 Mb
Medicine	Digital Chest X-Ray	1,024 b × 1,024 b × 12 b/pixel	13 Mb
	Emission Computed Tomography	128 b × 128 b × 16 b/pixel	260 kb
	Nuclear Medicine	256 b × 256 b × 16 b/pixel	1 Mb
	Nuclear Magnetic Resonance	512 b × 512 b × 16 b/pixel	4.2 Mb
	Ultrasound	512 b × 512 b × 8 b/pixel	2.1 Mb
	X-Ray Computed Tomography	512 b × 512 b × 14 b/pixel	3.7 Mb
Videotelephony	Medium Resolution (b/w)	512 b × 400 b × 8 b/pixel	1.6 Mb
Stills	Medium Resolution (color)	512 b × 400 b × 24 b/pixel	4.9 Mb
Motion Video	High Resolution (color)	1,024 b × 1,024 b × 24 b/pixel	25 Mb
(30 frame/s)	5 s Medium Resolution (b/w)	1.6 Mb/f × 30 f/s × 5 s	240 Mb
	5 s Medium Resolution (color)	4.9 Mb/f × 30 f/s × 5 s	735 Mb
	5 s High Resolution (color)	25 Mb/f × 30 f/s × 5 s	3.8 Gb
Office	VT100 ASCII Text Screen	80 c/l × 24 l × 8 b/c	16 kb
Automation	8.5in.×11in. Page ASCII Text(Courier)	66 c/l x 55 l x 8 b/c	29 kb
	Scanned 8.5in×11in. Page (b/w)	8.5in×11in.×(300 px/in.)^2×8 b/px	67 Mb
	Scanned 8.5in×11in. Page (color)	8.5in×11in.×(300 px/in.)^2×24b/px	200 Mb
	5 s Telephone Quality Audio	7,000 S/s × 5s × 8 b/S	280 kb
	5 s Stereo CD Quality Audio	44 kS/s × 2ch × 5s × 16 b/S	7 Mb
	5 s LPC Coded Voice	2.4 kb/s × 5 s	12 kb

Table 1.1: Typical uncompressed objects sizes

In fact, the four notions behind the definitions given here are aspects of multimedia. A loose definition of multimedia could be "relative to at least two perception, representation, presentation or communication media".

1.4 Current needed performances

In that part, the goal is only to get an idea of the current required performances in multimedia and the ability of the current technology to provide them.

Table 1.1 from [LG90a] presents the typical size of uncompressed objects in bits as they are needed in typical applications. The size ranges from 16 kb for a typical VT100 Text screen and 12 kb for coded voice up to 3.8 Gb for 5 seconds of high resolution color video. Note that currently PCs and equivalent handle text and still images. For a classical VGA color screen 640 × 480 pixels with 8 plans (using a color map technic), one image needs 640 × 480 × 8 = 2.46 Mbits, and it is possible to store almost 140

QOS	Maximum delay (s)	Maximum delay jitter (ms)	Average throughput (Mbits/s)	Acceptable bit error rate	Acceptable packet error rate
Voice	0.25	10	0.064	$< 10^{-1}$	$< 10^{-1}$
Video (TV quality)	0.25	10	100	10^{-2}	$< 10^{-3}$
Compressed Video	0.25	1	2–10	10^{-6}	$< 10^{-9}$
Data (file transfer)	1	—	2–100	0	0
Real-time data	0.001–1	—	< 10	0	0
Image	1	—	2–10	10^{-4}	$< 10^{-9}$

Table 1.2: Traffic streams: characteristic parameter values

	FDDI		DQDB	B-ISDN	
	I	II		STM	ATM
Standardizer	ANSI X3T9	ANSI X3T9	IEEE 802.6	CCITT SG XVIII	CCITT SG XVIII
Medium	Fiber	Fiber	Fiber	Fiber	Fiber
Structure	2 rings	2 rings	2 buses	star(bi-direct)	star(bi-direct)
Frames per second	—	8000 (125 μs per frame)	8000 (125 μs per frame)	8000 (125 μs per frame)	Open
Data rate	100 Mbits/s (per ring)	100 Mbits/s (per ring)	$\approx$150 Mbits/s (per bus)	$\approx$150 Mbits/s	$\approx$150 Mbits/s
Isochronous access	—	Reserved slots	Reserved slots	Always	Hybrid?

Table 1.3: High speed networks: features and standards

uncompressed images on a 40 Mb hard disk[3]. For typical workstations, (the tendency of the market being to bring PCs performances up to workstations) we have $1024 \times 1024 \times 8 = 8.3$ Mbits. Compared to the 3.8 Gb of 5 seconds of high resolution color video, there is a ratio of roughly 450 to 500.

Table 1.2 from [HSS90] presents the performances needed for current traffic streams that are to be compared with the capacity of the currently defined high speed networks given in table 1.3 from the same author[4]. Even if these networks meet the data rate requirements for current data streams, other parameters will have to be considered and guarantees given on their values. These are transmission delay, delay jitter, throughput and error rates at different levels. This implies protocol definitions or adaptations considering resources allocation[5] (buffer, bandwidth, ...).

[3] $\simeq$ 4.6 seconds of video at 30 frame/s.

[4] Note the word *medium* used in table 1.3 to point out the physical transmission technology.

[5] these aspects are discussed later.

1.5 Approach of multimedia

In this section, we try to approach some multimedia implications in an informal way. This should help the novice multimedia reader to intuitively understand critical aspects of multimedia.

1.5.1 Intuitive approach of multimedia

For many people, multimedia vaguely means two or more media handled together, that is to say almost any combination of text, graphics, sound and video (as said before, this is the definition of many technical dictionaries in computer science !). In that context, the term of combination stands for sequencing and synchronization of many media segments and needs a clearer definition, too.

Multimedia has another aspect that can discourage the non "hardware specialists". Many interfaces need to be used, and Integrating many seldom compatible devices is usually problematical (see [GP91] for an extensive discussion of these problems).

1.5.2 Technical approach through examples

Several notions can be approached through the following examples. Such a method means that redundancy is expectable. Note that we are not talking only about multimedia applications, but also about one-medium applications.

1. Let's imagine an application with still images and sound, such as slides with music or spoken text, displayed together. The images must have a given order and the recorded sound sequences must be ordered according to the images. Here, sequencing is necessary (we need to define a total order between the images and the sound segments) as well as *synchronization* (a given sound sequence must begin when the related slide is displayed, and the next sound sequence must wait for the next slide to be displayed).

2. Let's consider an application that displays video on a computer screen. Images are stored and displayed by a computer not by a VCR (strictly, this is not a multimedia application). As video needs a given number of frames regularly displayed on the screen (30 frames/second for NTSC for example), we need *real-time functions.*

3. Multimedia applications will provide the user (let's consider several types of users : the application programmer, using the application for creation of multimedia documents, and the basic user, using the application programmer's production) with a new role : the director's role. His situation is similar to that of a film director. In other words, presently, interaction between the application user (programmer here) and information is synchronous: for example, when we see a film, we see the whole film, not the frame number 12445 and then the frame 32154. The user needs to access information in an asynchronous way, as the film director can see the rush number x to know if it should come after number y or not. This means that data should be accessed both in a linear and in a *non sequential* way.

Let's summarize the above considerations. So as to handle several media in a *multimedia application*, we need :

- synchronization, including sequencing
- real time
- non sequential access to data[6].

Note that sequencing is usually considered as an aspect of synchronization, but one may think that it's not.

Finally, we would like to make the distinction between two types of media, ie stored and acquired media. Some applications, such as hypermedia applications or multimedia mail systems, use stored media. For example, a mail system can retrieve the documents from a disk, build the mail and send it. Other applications, such as videotelephony or air traffic control, use data acquired as it is generated. We need these two types of media to be combinable.

1.6 Implications of multimedia

Multimedia means new devices and a new behavior for users and developers. At the present time, multimedia actually involves vision and audition, and other senses (tactile devices are under development —and some of them are even integrated in games— , but research is needed for smell and taste so

[6] this is called *random access* by MPEG [JTC90a] (see also [LeG91]).

as to reach the complete virtual world —note that an attempt was made for films with odor—).

The multimedia application programmer will have new synchronization tools:

- indexing tools
- sequencing tools
- linking tools
- ...

as well as new high level user interface building tools. He will work on new documents that may not be printable as any classical C program is.

We won't here describe the philosophical consequences of virtual worlds, nor make plans on what the future multimedia development platforms will be. Anyway, one can easily feel that it will upset our habits. The challenge is the definition of the single medium, handling the five senses.

1.7 Immediate requirements

The following items are consequences of the previous considerations. This is what should constitute the future work in the field of multimedia.

1. create new devices and new interfaces (a real standard is needed).
2. merge the devices, combine them so as to create multimedia systems
3. create multimedia data, either data structures and contents.

All these topics involve synchronization notions and other descriptive features that need to be clearly defined[7]. Theoretical tools are needed, too.

Multimedia will require the development of new supports at any level :

1. new hardware
2. new (or extended) operating systems with multiple system calls ability:
   - continuous data handling and storage
   - real time
   - ...

[7] see chapter 4 for an attempt.

3. new User Interface = something as a generalized X (X is currently designed for bitmap displays but is easily extensible. How far is it possible to extend it so as to handle the new devices[8]?).

4. new applications

5. new networks with time constraints and guarantees on communication quality of service

6. new storing devices for quicker access and larger storing space along with new controllers.

1.8 Conclusion

More than a complete and detailed definition of the term *multimedia* that is sometimes wrongly used as a noun, we have presented some implications of multimedia in what concerns hardware, applications and communications in general. Refer to [GP91] for an extensive presentation of intuitive aspects of multimedia in a non formal manner, insisting on current practical aspects.

Actually *multimedia* has so many implications that it almost reaches each research field in computer science. We have chosen to put, in this work, the emphasis on the data structures *multimedia* documents. Multimedia communication is another quite important field; it is also being developed at LAAS and UCB.

[8] see chapter *Xwindow for Multimedia.*

Chapter 2

Multimedia Applications

Application	*Media*	*Selected Functions*
Office Automation	Images,Text	Composition,Filing
Medical Information Systems	Video,Images,Text	Data Acquisition,Filing
Geography	Images,Graphics	Data Acqu.,Storage,Image Manip.
Education/Training	Audio,Video,Images,Text	Browsing,Interactivity
Command and Control	Audio,Images	Data Acqu.,Data Modification
Weather	Images,Numeric Data,Text	Data Acqu.,Simul.,Data Integrat.
Banking	Numeric Data,Text,Images	Image Archiving
Travel Agents	Audio,Video,Images,Text	Video Browsing
Advertising	Video,Images,Text	Image Composition Enhancement
Electronic Mail	Audio,Images,Text	Communication
Engineering,CAD/CAM	Numeric Data,Text,Images	Cooperative Work
Consumer Electronic Catalog	Audio,Video,Text	Video Browsing
Home Video Distribution	Audio,Video,Text	Video Browsing
Real Estate	Audio,Video,Images,Text	Video Browsing
Library	Images,Text	Database Browsing, Query
Legal Information Systems	Images,Text	Database Query
Tourist Information	Audio,Video,Text	Video Browsing
Newsprint Publication	Images,Text	Image,Text Composition
Dictionaries	Images,Text	Database Browsing, Query
Electronic Cooperation	Audio,Video,Text	Videoconf., Concurrency Control
Air Traffic Control	Audio,Text,Graphics	Concurrency Control

Table 2.1: Multimedia applications

2.1 Introduction

Text editing, pixmap editing, graphics editing have already been mixed to allow diverse operations on compound documents integrating text, images and graphics. The most achieved applications in that field are Computer Assisted Publishing programs with WYSIWYG[1] interactive screen output. Hypermedia applications provide the user with a way to link existing documents while conferencing multimedia applications allow the introduction and exchange of existing documents.

Table 2.1 from [LG90a] presents the currently considered multimedia applications along with the media they use and some typical functions they provide.

[1] What You See Is What You Get.

2.2 Computer Assisted Publishing

Multimedia publishing application usually means creation of multimedia documents with static data[2] such as text, graphics, still images. Thus, the goal of such applications is to produce printable documents[3], i.e. with paper as final support.

An example of such applications is BALZAC described in [Naf90]. This multimedia editor was developed in the context of an ESPRIT project. This application is interactive with WYSIWYG[4] output, and handles sound along with text, graphics, images, mathematical expressions, tables and formulas, combining them so as to produce mixed documents called multimedia documents or compound documents.

The documents are structured at several levels :

1. each document is an instance of a document class. These classes can be either predefined or user created. A class is described by grammar rules and defines the logical structure of its instantiations. Typically, a tree is described, the leaves being blocs of either text, digitized image, structured graphic, decisional graphic, or mathematical formula. Figure 2.1 gives an example of a logical structure description, i.e. a class description.

2. each logical document class is associated to one or more physical description, a document being described with one of them. The physical descriptions are trees whose structure is independent from the logical structure, except for the leaves which are common elements of the logical and physical trees of a document. The physical structure is described in terms of a series of columns formed with blocs (the same blocs as in the logical tree).

3. each terminal element, i.e. bloc, contains pointers to attributes such as margin width, font size, underlining method, permission to hyphenate words, etc...

The formatter is integrated in the application. As the latter is interactive, the formatting is dynamically performed for the WYSIWYG editor. The characteristics of blocs are modified through editing functions dependent upon the nature of the constitutive material.

[2] see chapter *data structure definition*: class of a consistent data unit.
[3] see chapter one section 6.
[4] see introduction in this chapter.

report	::=	$document_title summary intro chapter+ conclusion
document_title	::=	BLOCKp
summary	::=	TABLE_OF title_intro title_chap+ title_conc title_sec
intro	::=	title_intro section*
title_intro	::=	BLOCKp
chapter	::=	title_chap section*
title_chap	::=	BLOCKp
conclusion	::=	title_conc section*
title_conc	::=	BLOCKp
section	::=	title_sec elt_sec
title_sec	::=	BLOCKp
elt_sec	::=	BLOCKpidgm

$: succession of documents, +: repeating element, *: optional or repetitive element
p = text, i = digitized image, d = structured graphic, g = decisional graphic,
m = mathematical formula.

Figure 2.1: Example of a class description

Such a way to describe a document (ie logical and physical structures) is widely used and other application descriptions with similar documents can be found in [HK86] and [TMMN86].

2.3 Electronic mail

From the first implementations of electronic mail protocols till now, many versions of electronic mail systems have been implemented. Between 1984 and 1988, CCITT X.400 (see [Sch89]) was defined along with ISO MOTIS (ISO 10021). The current mail handlers are single medium applications integrating many functions such as document composition (transfer, information about the arrival, diffusion of the mail, etc...). In [Tan90] such an application is modeled with the OSI network model by mapping it on the two upper layers: application and presentation.

2.3.1 Current mail handlers data structures

The current widely available mail handlers use a data structure defined in RFC 822 and the SMTP protocol[5]. RFC 822 considers only messages containing ASCII characters and nothing else (other character sets, facsimile, voice, video, ...)[6].

The data structure is very simple: a message is a file with a set of special fields at the beginning. Each field begins with a keyword followed by a colon and a value. Table 2.2 from [Tan90] gives a list of the fields of mail headers[7]. Some rules define how to handle large lines, spaces and format indications and everything between brackets is considered as comments. Thus, the data structure itself contains its description and so, modifying the data or the description are equivalent editing functions (a simple text editor is enough); this can be either very damaging or practical.

2.3.2 Multimedia mail systems

Several multimedia mail systems have been developed and some of them are publicly available such as Microsoft Mail which allows the exchange of bitmaps or BBN's mail system on Sun workstations. The NeXT mail system handles formatted text, images, audio and voice, and the Diamond message

[5]Simple Mail Transfer Protocol.

[6]other ARPANET standards handle some of them.

[7]The fields concerned with forwarded mail are not listed.

Field	*Description*
Sender	Address of the person who sent the message
To	Recipient's address
Received from	Where did the message come from
Received by	Who received the message
Received via	On which physical medium did it arrive
Received with	Which protocol did it use
From	Name of the person who sent the message
Reply-To	Address to reply to
Cc	Addresses to which copies are to be sent
Bcc	Addresses of blind copies
In-Reply-To	Message ID being replied to
References	Other messages cited
Subject	What the message is about
Keywords	Content descriptors
Date	When was the message sent
Message ID	Message identification
Comments	User defined
Encrypted	Index into encryption key table

Table 2.2: RFC 822 header fields

system (see [TFC+85]) can send some more media types. The Andrew message system (see [BERS89]) handles compound multimedia messages with hierarchical data structures, hypertext links and even some programs. Some research projects include video.

The above multimedia mail systems are incompatible with the existing SMTP based ones that are widely used. Thus, building such an application running along with the existing mail facilities sounds interesting. The MONTAGE system (see [Edw91]) is designed in such a way and a prototype has been built. The data structure is more complex than just a file. The authors make the distinction between *dynamic media* with which "the presented information changes with the time", and *static media* that "do not have this temporal component". They also make the assumption that it is useless to present simultaneously dynamic and static data, the user being unable to figure out to what paying attention. Thus, each message has a primary class, either dynamic or static, that defines the class of the media in the message. These media are called together the *primary component* and displayed in a single window. The primary component can have attachments that are submessages and may be of any medium class.

To be also compatible with MOTIS systems, MONTAGE implements a subset of X.400 in the messages' headers. The messages have a header

```
Author:  Keith Edwards <keith@cc>
CreationDate:  18 Oct 90 10:18:52 PDT
Subject:  Notes from the meeting
ID: 2848.AA08665
TOCVersion:  1.0
Class:  static
Primary:  txt.2848.txt SimpText
Attachment:  snd.2848.snd CODEC88 27
Attachment:  txt.2848.txt SimpText 30
Attachment:  xbm.2848.xbm XBM 49
```

Figure 2.2: Table of contents in Montage (example)

and a body which is divided in chapters, the first of them being the *table of contents.* The chapters contain one-medium type data (or the table of contents for the first one), and are single files. The table of contents describes the structure presented above (figure 2.2 from [Edw91] gives an example). The messages contain binary parts and so are first compressed to compensate the ASCII translation effects of the Unix command uuencode[8].

Note that the user interface of the prototype is Xwindow based[9] because "it is a standard" and "it provides runtime customization mechanisms". The attachments are displayed as icons on the side of the main display window.

Eventually, a difficult problem of multimedia mail systems is the interchange formats incompatibility and the large number of existing and proposed standards for document interchange (such as ODA or HyTime), coding (CDA, PostScript, ...), exchange (X.400, SMTP, ...), etc...

2.4 Hypertext and hypermedia

Beyond simple character strings linking, hypermedia is considered by many authors as the necessary extension of hypertext toward images, sound, video and so forth. This remains based on the hypertext technics and experience.

Here, we do not present the history of hypertext systems. It can be found in [Nie90] where the ancestors of the current systems are described starting with Vannevar Bush's Memex in 1945. An even more complete review of the

[8] uuencoded files are quite resistant to compression.

[9] see chapter *Xwindow for multimedia.*

recent systems has been achieved in [Con87] where hypertext systems are classified according to their application area. Finally, in [Nie90], chapter 4 presents the application of hypertext in the following fields:

- computer applications: online documentation, user assistance (online help), software engineering (software management tools), operating systems (with system based hypertext services)
- business applications: technical manuals (repair, maintenance, ...), other manuals, dictionaries and reference books, auditing (management of auditing), trade shows, product catalogs, advertising
- intellectual applications: brainstorming support, journalism, research (idea generation and writing)
- educational applications: foreign languages, classics, museums
- entertainment and leisure: tour guides, libraries, interactive fiction.

For us, some of these applications are extensions of what is usually called hypertext, and the disagreements between hypertext supporters and multimedia partisans are closer to religion wars than technical challenges.

2.4.1 Hypertext: linking texts

The first example of hypertext is that[10]. A footnote in a text is one of the ends of a link between two texts. That's why hypertext is sometimes referred to as *the generalized footnote.*

But hypertext is more than that, and there is a general agreement on what it is not rather than what it is. The following list gives some of the necessary features an hypertext system must have. These are not the set of sufficient features that comprises some "look and feel" criterias along with user oriented facilities.

Thus, a hypertext system must comprise:

- the ability to create new documents, that is to say new nodes for the hyperdocument
- a distributed database of textual and graphical documents

[10] yes, a footnote. You have the choice of either reading the footnote or keeping reading the main text. The fact a footnote is marked as a superscript is a choice of two way link implementation.

- the ability to link the documents thus creating what is generally called a hyperdocument. This is a set of nodes, each of these containing one document and linked to other nodes

- a window based user interface managing a dynamic set of windows and icons. Each window displays one node. Each icon is either an iconified window or a pointer to another node inside a window. The latter contains a short textual field suggesting the content of the node it points at . Of course, the window manager provides classical windowing functions: open, close, iconify, move, resize, ...

- the ability to create links between documents, already linked to other documents or not, thus extending the hyperdocument link set

- the ability to browse the hyperdocument by following the links, by searching for a given document, by using a graphical network representation of the hyperdocument.

This list of features is not a definition, for, such a definition taking non-formal criterias into account, a system is considered as hypertext or not by a general agreement more than by fulfilling specific criterias.

In [Con87], several types of links between documents are defined, and, in [Tri83], more than 80 types of links are defined. The links can be *referential links* that have a source in a text and a destination which usually is a node of the hyperdocument. Some additional semantics can be defined. Thus, *organizational links* define a hierarchy among the linked nodes. A *keyword link* has a given word as the source and that word can appear several times in the text. All the imaginable types of links cannot be exhaustively part of a structure without dramatically increasing the complexity of the latter. Thus, in many systems, such as the Neptune system[11] (see [Con87]), links contain couples *(attribute, value)* that allow the easy extension of the basic linking structure[12].

Some systems make the distinction between different types of codes. Thus, it is possible to use information formatted in various ways in the same hyperdocument. It is also useful to define an application dependent set of semantics for attributes defining the type of node relatively to the meaning of the information: for instance, it is possible to define articles, notes, decisions, reports, etc...

[11]Tektronix Neptune is a hypertext designed as an opened and layered architecture.

[12]our structure in chapter 4 section 6 is extensible in that way.

In [Con87], the notion of composite node is also defined as a "mechanism for aggregating related information [...] Several related hypertext nodes are "glued" together and the collection is treated as a single node [...]".

Finally, some authors[13] use a specific hypertext vocabulary that seems to be more and more shared among hypertext adepts. The main of these notions are :

- *anchor*: selection in a text such as a fluorescent highlight in a book
- *navigational linking*: ability to connect anchors using persistent ties that enable them to move between references and documents
- *warm linking*: navigational linking allowing the exchange of data over the links
- *hot linking*: linking with automatic synchronization between anchors One anchor is the *master* and others the *instances*. When the master is modified, the instances are also modified
- *active anchor*: anchors from which animations are started
- *web*: collection of anchors and linking information among a set of documents.

2.4.2 Hypermedia: the extension

For us, hypermedia is seen as a subset of what we call multimedia approached through hypertext. The notion of *"active anchor"* that the hypermedia designers have to consider is an example of what multimedia takes into account as a basic problem.

2.5 Teleconferencing

Multimedia teleconferencing systems are applications that need special guarantees from the computing and communication environments. Thus, the information exchanged during conferences needs to be synchronized and some real time guarantee is necessary, imposing tight constraints on the underlying networks. But beyond that, teleconferencing raises other problems such as document sharing and conference chairmanship. The descriptions of some

[13] see [Mey89].

teleconferencing systems can be found in [Sak90], [LBH+90], [EAC+91] and [JS91].

In [Sak90], the development of a desk-to-desk conference system is presented. The author shows how a new LAN architecture and a new protocol were developed so as to solve the problems of real time communication and information sharing. The issues considered in the development of that system are:

- environment related issues:
  - making communications as close as possible to face-to-face interaction
  - group coordination
- data related issues:
  - interactive data exchange and handling
  - data compatibility with personal work support environment.

The system comprises a LAN with PCs featured with telephone controllers. The software was developed in C under MSDOS and has four main components:

- personal support: provides functions such as filing (storage and retrieval), document creation and editing for personal files, mail handling
- convening control: provides directory access control, telephone control and PC control
- conference control: provides the sharing facilities such as file access control, conferee or floor passing control, I/O control, protocol handling and multimedia handling. The latter provides functions for editing and displaying multimedia documents comprising text, graphics, images and handwritten information
- user interface.

The multimedia documents are hierarchically structured in *document / page / area* using a simplified subset from ODA (see [IS-86]). The structure gathers texts, graphics, facsimile images and digitized voice. However, the main aspect of the system is the design of a protocol allowing the distribution of handwritten (using writing pads or writing boards) and mouse-pointing

information, the floor control that allows continuity in presentation and consistency of shared screen information[14], the ability for any conferee to join and leave while the conference is in progress.

In [EAC+91], the authors of the Rapport system focus on the functions that must be associated to networks supporting conferencing systems:

- establishment and management of connections
- management of information exchange
- synchronization of exchanges.

This Rapport multimedia conferencing system is presented. Based on UNIX and Xwindow, the system uses an heterogeneous set of networks. The system does not take complex data structures into account but focuses on the common control of typed communication channels. Multicasting (replication of a single input signal and delivery to multiple destinations) and bridging (combination of multiple input signals to produce one or more output signals) are studied. Thus, the issue raised by this teleconferencing system does not imply the definition of complex multimedia data structures.

2.6 Video applications

The most wide-spread video application is video browsing that is sometimes called *interactive video* because it allows the user to select video sequences in a list. But the interactivity is, most of the time, non-existent during the display of a given sequence. Another kind of video application is that of the application presented in *appendix 1*, that, from raw data, generates video frames and displays them. Such applications can be called *video synthesis* applications.

But more elaborated applications are currently being developed that imply the definition of complex data structures to describe links between video sequences and other information. Such video applications gather several issues that are due to the high needed data rate and the tight synchronization with audio and other information. Thus, data compression is today a necessary condition for the use of video in multimedia systems.

The Digital News System developed by Apple Computer, Inc. for the EDUCOM'90 conference and presented in [HG91] was tested on participants

[14] this feature is called WYSIWIS for What You See Is What I See.

as an interactive news system. The end user is supposed to "drive" the news. A team of video editors, journalists, artists and HyperCard designers daily composed the news. Video contents were provided by CNN through a direct microwave video link and were stored on writable optical video disks with fast random access, frame-accurate positioning, high image quality variable rate and direct playback. The selection of sequences was done by selecting the SMPTE time codes[15], and, after composition, the NTSC analog video signal was digitized, compressed and translated into a Machintosh suitable playback format. For each video sequence, the sound track was also digitized and multiplexed in a single file. ASCII text and PostScript images were used, too. The images could be manipulated (text overlay, resizing, cropping, composition, ...).

The HyperCard environment allowed, from the composed video news magazine, to build interactive features. Once tested, the interactive version, roughly 35 Mbytes, was distributed on a hybrid network (two Ethernet LANs) via T-1 lines at 1.5 Mbit/s, and automatically loaded on 65 Machintoshes. The latters had 24 plans frame buffer and external speaker directly using the computer's 8-bits sound output. Each user could browse through the news, selecting the points of interest. Some hypertext links were used to access more detailed information on a given topic (speech transcripts, news analyses, charts, graphs, ...). Texts could be scrolled and video stopped at any time.

This multimedia system may be the kind of application that will be soon widely developed. It shows the needs for interactive mechanisms along with pilotable preorchestrated multimedia sequences. A further step of such a system will be to access large worldwide databases on any subject, allowing the user to investigate a particular aspect of a subject. For example, a political event in a foreign country can be the occasion to look at the political history of this country and at its geography to understand the underlying context. The information thus required will have to be remotely accessed and to be displayed in real-time. The needed data rates require operating systems and network services giving guarantees on their own performances, and efficient compression. Moreover, complex synchronized documents will have to be described so that these services should be able to support the composition, presentation, distribution of general multimedia documents.

[15]the Society of Motion Picture and Television Engineers gave its name to a standard code (*hr:mn:s:frame*) uniquely identifying a frame.

2.7 Conclusion

We have presented a heterogeneous and non-exhaustive list of multimedia related applications. Some of them use only what is sometimes called static media, but as soon as video and sound are integrated, many issues are raised such as synchronization and performances guarantees. To build viable multimedia systems, the definition of a generic and general multimedia data structure is necessary. A review of the information it should contain is given in next chapter.

Chapter 3

Data Structure Requirements

3.1 Introduction

Some attempts to define a multimedia data structure have been achieved by many authors. In [Edw91], the data structure is a two levels hierarchy realised through attachments. Synchronization is achieved on the main level by the definition of simultaneity. In [Lub91], an extension of ODA[1] is proposed by defining the notions of dynamic and static frames and blocks of information. In hypertext [Nie90], the structure is a collection of links that can be either mono-directional or bi-directional. The latter allows the user to link information but seldom allows to impose some other multimedia requirements (presentation, transmission, protection, ...), while the hierarchical data structure allows the definition of simultaneity in the presentation, but doesn't take the needs of links into account, except through a one-level attachments set.

Other authors develop models for multimedia data by extending concepts used in software engineering. Thus, in [KNS90], an object oriented approach is proposed by extending the concepts of class and meta-class. After the definition of their model, the authors show how it can solve problems such as versioning, archiving and database management. But their purpose is not to give solutions for synchronization in the presentation of multimedia objects.

In that chapter, we present in more details what, according to us, the main aspects of multimedia should be. This presentation is the basis for next chapter. For the latter, we have chosen some aspects to focus on, and then give possible directions to take the other aspects into account by extending our data structure.

3.2 Dimensions

A multimedia document has several aspects that we call dimensions to point out the fact that these aspects are at least partially independent. We first loosely define the notion of consistent data unit[2], then we present the dimensions developed in the next sections.

[1] Office Document Architecture [IS-86].

[2] a definition is given in chapter 4 section 'Basic definitions'.

3.2.1 Consistent Data Units - Documents

A first definition of Consistent Data Unit (CDU) is any unit of data that can be distributed over a multimedia system between applications and that can be either complex or simple. We however do not include multimedia documents (presented in chapter 4) that are complex data structures defining logical and synchronized relations between CDUs and other documents.

Thus, simple numeric data, encoded audio, compressed video, pixmaps, formatted text, ..., are CDUs. Moreover, a pixmap is a CDU itself or is a part of a larger CDU comprising many pixmaps and defining a video segment. Actually, the size of a CDU is most variable depending on the application and the description level it deals with. For example, a text editor handles CDUs that are characters, words, paragraphs, documents.

In chapter 1, a table gives examples of CDUs along with their application field and their size[3].

3.2.2 Spatial dimension

This aspect of a multimedia document is *called logical* structure by many authors. The Office Document Architecture (see [IS-86]) makes the distinction between the logical and layout structures of a document.

The *spatial dimension* of a document is given by its components, the CDUs and subdocuments, and linked documents. At a given date, some of these components are presented to the user.

Another spatial aspect of documents is the *relative presentation* which takes the identity of the user for the presentation and editing processes into account.

3.2.3 Temporal dimension

The CDUs and linked documents that represent the spatial dimension of a document are ordered in time for their presentation. This is the *synchronization aspect* of the *temporal dimension.* Another aspect is the *lifetime* of a document. The latter can be built only for presentation purposes by an application without being stored or archived, but being suppressed once used. It also can be stored for later retrieval purposes.

Two types of synchronization are distinguished: basic and isochronous. The *isochronous synchronization* (see [Nic90]) or *continuous synchronization*

[3] table entitled 'typical uncompressed objects size'.

(see [Ste90]) is that which is needed between two streams such as sound and video. The classical problem of synchronization between a sound segment and a video segment in which somebody is speaking, thus requiring tight synchronization between the lips movements and the sound track, illustrates that notion. On the other hand, synchronization may only be needed at the beginning and the end of the presentation of some CDUs. This is what we call *basic synchronization.*

Figure 3.1 illustrates the notion of continuous synchronization.

3.2.4 Historical dimension

This dimension of a multimedia document deals with versioning. Many applications handle documents in different versions. Two kinds of versioning are distinguished:

- *time dependent versioning*, in which the document is considered at different dates
- *time independent versioning*, which is the document as it is presented to a given user. We call this relative presentation and consider it is part of the spatial dimension of a document. Next section addresses this issue.

In [KNS90], a scheme for versioning, for both time dependent and time independent versioning, is proposed for an object oriented approach of multimedia modeling.

3.3 Spatial dimension

The issue presented in this section is crucial for multimedia. Anyway, the next chapters do not address it because we will concentrate here on the temporal dimension. This implicitly assumes that it is possible to consider independently the spatial and temporal compositions of multimedia documents.

3.3.1 Presentation

The Spatial composition has been addressed by many authors and many systems have been developed only for that purpose. Thus, in Balzac ([Naf90]), composing functions are presented: alignment, spacing, mirroring, pivoting,

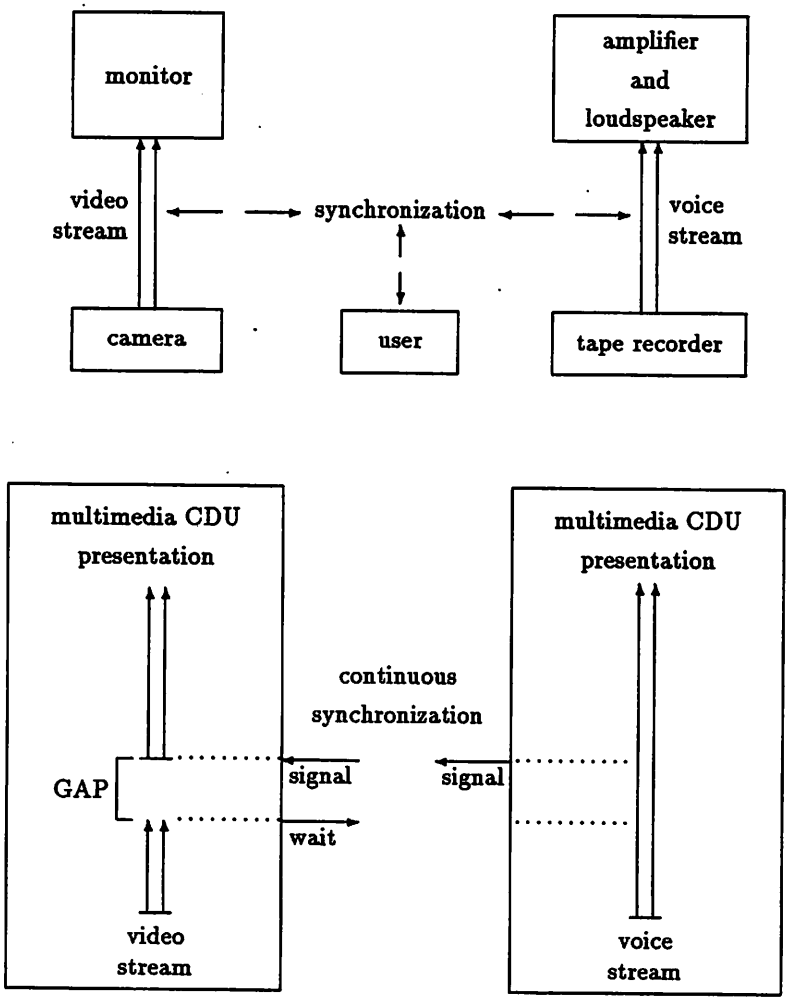


Figure 3.1: Continuous synchronization

transformations (zoom, polyline closure, torsion, rotation), manipulations in planes (cut, paste, ...), changing of display attributes such as line motif, positioning, recopying, erasing, filling, and so forth. If an overlay of several CDUs occurs, various operations can be performed for the presentation process. These are merging or composing operations such as raster operations.

The definition of media and multimedia documents must take that into account by providing a model for spatial composition. Standards have been developed for documents using static media[4] such as in the Office Document Architecture (see [IS-86]). In [MK88], too, an object oriented approach is given for spatial information presentation and representation.

3.3.2 Relative presentation

Beyond spatial presentation, the notion of *relative presentation* takes the identity of the user into account. It gives a relative view of a document to a given user according to his fields of interest and access restrictions. Thus, a given document has several external presentations.

Let's take an example: the multimedia document for a new industrial vehicle. The document comprises: a technical description, a development report, economical studies, user manuals, assembly directives, maintenance manuals, etc ... The production department is interested in the technical description, the results of the development report and the assembly directives. The finance department needs the economical studies and some parts of maintenance manuals for use and maintenance costs evaluation. The relative presentation scheme intends to provide the bases in the document itself for the ability to implement the relative presentation functions.

3.4 Temporal dimension

The temporal dimension is the new part of multimedia documents and it will be carefully considered here. In this section, we make the distinction between:

- *isochronous synchronization* it is a continuous temporal dependence relation between CDUs and documents that is a stream relation
- *basic synchronization* which doesn't require stream synchronization.

[4] see chapter 4 'Basic definitions'.

3.4.1 Continuous synchronization

From the point of view of multimedia applications, some CDUs are acquired simultaneously with continuous synchronization, whether it is retrieved from a storage device or directly acquired from input devices. This kind of CDUs is for example video streams or two video streams and a sound stream (see fig. 3.1). The isochronous synchronization requires some guarantees on the transportation of data provided by underlying networks such as: end-to-end delay bounds, delay jitter. Possibly, some bounds on the error level are needed[5]. Such requirements imply buffering and buffering management (such as in [VF90]).

The issue of isochronous synchronization is addressed also in [Ste90] and [Nic90] in a more or less extensive manner.

3.4.2 Basic synchronization

Some CDUs and documents don't require continuous synchronization between each other but their presentation relies on the existence of synchronization points, mainly at the beginning and the end. We assume that complex basic synchronization relations can be described using elements of a consistent set of basic synchronization relations between two CDUs. To study this issue, a model is needed. Many of them have been developed as for example in [Lub72] or in [KG77], and in [LG90b].

In [Lub72], it is shown that the relationship between two simply synchronized CDUs is one of thirteen basic schemes. In [LG90b], this set is consistently reduced to seven basic synchronization schemes by considering inverse relations[6]. Thus, showing that a given model can be used for a formal description of these seven basic schemes is showing that this model is suitable for modeling basic synchronization between two CDUs and that this model is applicable to depict and represent relationships between the elements of a set of CDUs or documents.

Figure 3.2 from [LG90a] gives a list of these schemes and their formal description using an extended Petri net model developed in [LG90b].

[5] we say possibly because it seems that CDUs requiring time continuous synchronization are usually permissive in loss rate.

[6] for instance *after* is considered as $before^{-1}$.

These basic schemes will be used in chapter 4 to show the usefulness of the described data structure for multimedia.

3.5 Ten criteria and other considerations

Defining the needs of a data structure for multimedia would complete if quality criteria such as reusability, validity, extensibility and others were accounted for. The data structure definition intrinsically contains a balance between opposed considerations.

3.5.1 Ten criteria

The problem of describing a document is first to know at which level it should be described. It can contain almost everything, be so precise that the linked information inside is only a small part of the structure, depending on who and what will handle it. A balance has to be found between implementation criteria that imply solutions sometimes opposed to each other:

- *validity*: the structure must provide the bases for the multimedia functions that are defined
- *extensibility*: the structure must be extensible for future or misunderstood functions
- *compatibility*: the structure must be able to receive CDUs of different formats
- *trustworthiness*: the structure must contain consistent information on its own contents
- *reusability*: the structure is expected to evolve, existing software must be usable at least temporarily with the evolved structure
- *efficiency*: the structure must be able to receive hardware dependent information so as to take advantage of it
- *portability*: the structure must at least be able to give its identification, that is its coding rules
- *security*: the structure must contain mechanisms to protect itself or parts of itself

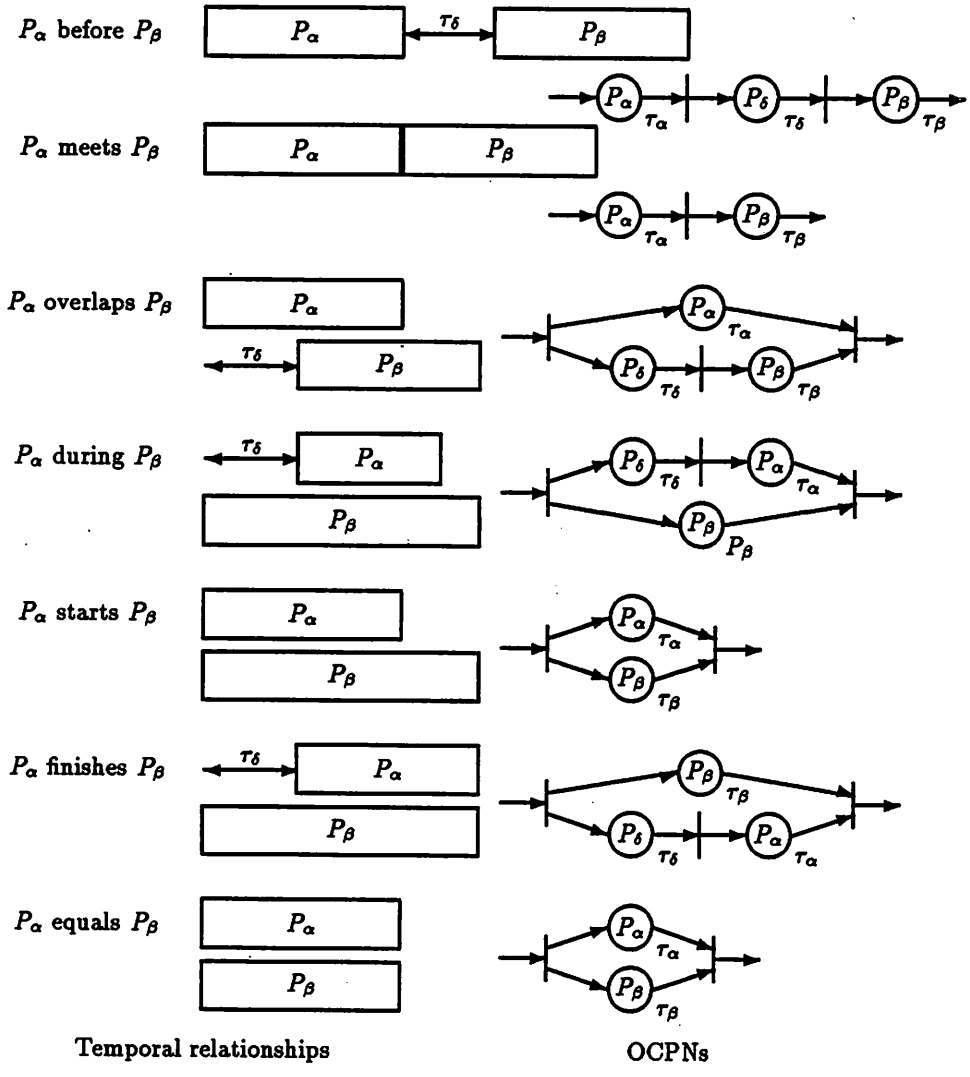


Figure 3.2: Seven basic synchronization schemes

- *verifiability*: the structure must be self-descriptive enough to be verified in some other way than presentation or editing
- *sharing ability*: the structure must be shared simultaneously and so provide bases for concurrent access mechanisms.

3.5.2 Data protection

Protecting information is a complex issue that can mainly be achieved in two ways:

- restriction on information access
- information encryption/encoding.

We recommand [WK85] for an extensive description of the way these problems are solved in the UNIX system.

A multimedia document structure should allow the definition of mechanisms so as to protect the information itself by encryption, coding, compression, and similar technics. Moreover, the structure itself can contain information on its access restrictions (something as the red stamps on classified intelligence materials) such as locks, permission restrictions and other similar mechanisms.

3.5.3 Code transportation

The amount of data implied by multimedia presentation leads to high data rates so as to meet tight presentation requirements. Thus, increasing the data rate at presentation time is unlikely to be considered soon. But, for preorchestrated presentations described in a complex data structure and for which a deferred presentation is possible, *code transportation* can be considered. By code transportation we mean the ability of the data structure to carry over the code that can handle it. This code can be machine code or high-level language. Code transportation, rather difficult to achieve, is not further developed.

3.6 Data compression

The data rates needed for the presentation of multimedia documents, as showed in first chapter[7], are close or beyond the highest available rates. At the same time powerful CPUs and dedicated hardware architectures[8] are available that allow the use of computer cycles for compression/decompression of large amounts of data. The efficiency of the algorithms is obviously a critical parameter and currently implies a lot of research in the field of data compression. A complete review of that issue can be found in [Com91].

The definition of standards for still images and video compression is under study (see [JTC91] and [JTC90a]). With the development of ISDN, the upcoming videotelephony needed a video coding standard. The current CCITT Recommendation H.261 (Video Codec for Audiovisual Services) considers the use of a so-called $p \times 64kbit/s$ video coding scheme corresponding to the number of ISDN B channels used. A real-time coding/decoding scheme is developed with a less than $150ms$ delay, and, because of the low bit rate of ISDN B channels, the emphasis is put on the definition of few overhead. With $p = 1$ or 2, the image quality is poor and can only fit videotelephony. But, for teleconferencing, at least $p \geq 6$ is needed. Compression can be favorably performed on any data stream. In the following section, we however put the emphasis on video and still images compression. A useful multimedia data structure must be able to contain information compressed using various coding schemes.

3.6.1 Source/Entropy coding

There are two main types of compression schemes for video: *source coding* and *entropy coding.*

Source coding operates on native video streams by approximating the data. Thus, it is lossy and the quality of the presented image is degraded. Source coding is used in [UO91] to build a software video codec on Sun Sparc workstations. This scheme can be either *intraframe coding* or *interframe coding.*

[7] compare tables 1.1 and 1.3.
[8] see [HKL+91] for example (i750).

Intraframe coding, also called truncation compression, defines a tolerance Δ. A given frame is divided in tiles in which the variance of the data is compared to Δ. If it is greater, the tile is further divided in smaller tiles and the process continues till the variance is smaller than Δ: the data in such tiles is then averaged. Interframe coding, also called difference compression, considers the difference between two consecutive frames. Some parts of the considered frame that have less difference than a given threshold with the corresponding parts in the previous frame can be considered as transparent and do not need to be redisplayed.

Entropy coding is a lossless compression scheme that eliminates redundancy in the data stream. The most famous and used algorithms are the *Huffman coding* scheme and the *arithmetic coding* scheme. The Huffman coding scheme uses correspondence tables usually application dependent. Thus, the tables have to be sent before the data for presentation purposes. The arithmetic coding method uses statistical figures to encode the image. Thus, it is more complex than the Huffman scheme but 5 to 10% more efficient in terms of compression.

3.6.2 Compression specifications

The compression level applied on a given CDU is dependent upon quality criteria. JPEG[9] considers a large range of *visual fidelity*[10] from "very good" to "excellent". Such levels of quality should be studied precisely in terms of compression/quality parameters that a codec needs to perform the right compression algorithms on the considered CDU. Thus, a multimedia data structure describing CDUs must contain these specifying parameters. In the remainder of this report, we however do not address this problem which should be part of future work.

Currently, compression specifications must take various application dependent and hardware dependent aspects into account such as:

- *storage devices*: the retrieval rate of data implies given levels of compression

[9] ISO Joint Photographic Experts Group.
[10] see [Wal91].

- *symmetry of compression*: asymmetric applications use decompression very often, but use once for all compressed data. Symmetric applications equally use compression and decompression. Symmetric and asymmetric applications may require different coding algorithms
- *random access*: it implies the compressed data stream be accessible not only as a whole, but in parts
- *fast forward/reverse/play back*: the compression algorithms must allow that easily for some applications
- *synchronization*: as stated in chapter 3, some CDUs require continuous synchronization and so compressed and continuously synchronized data streams must have a structure allowing such a synchronization
- *error level*: this feature has two aspects:
  * some communication channels are not always error free and so the compression scheme must be robust
  * compression/decompression algorithms introduce more or less definition loss
- *delays*:
  * some applications may specify tight end to end delays thus requiring some guarantees on the compression/decompression delays, that is real-time data compression/decompression
  * parts of data streams should be independently coded so that the editing time is reduced or at least bounded.

3.7 Conclusion

Multimedia data structures gather characteristics that are usually independently considered and that need to be integrated. In the remainder of this paper, we put the emphasis on the presentation of synchronized CDUs. Thus, versioning, compression and protection are not addressed, not that we think these aspects are less important, but an independent approach of these problems may be needed to more precisely define the implications of these features.

Chapter 4

Data Structure Definition

4.1 Introduction

This chapter intends to define a structure called *multimedia document.* First, a model for multimedia systems is described and the field of multimedia documents clarified. Then, we give our definitions of notions that can be misunderstood because of many context dependent definitions in the literature. The following sections define the multimedia *document* and *link* structures.

The information is handled in computers as a succession of bits that represents the data in a discrete manner. This information can be either control information or manipulated data. The latter has a structure that the former must preserve.

What we call here a *consistent data unit* (CDU) can be intuitively defined as an amount of data free from control signals (other than self-descriptive, for example timing information relative to external CDUs or to processes other than the presentation process of the considered CDU) that has physical and semantic continuity and integrity that cannot be reduced. A CDU is a basic indivisible element, an atom of information. Intuitively, two video sequences can be understood as two different CDUs as well as a text segment, a pixmap, a sound sequence and so forth are CDUs.

In order to precise that notion, let's take two examples :

1. a video segment can be considered as a whole and sent on a network to a display station that will handle it knowledgeably as a whole video segment, as a *consistent data unit* that may be synchronized with other CDUs.
2. a similar result for the end user can be obtained with a succession of pictures sent on the same network with synchronization constraints between each other, that is to say with a set of synchronized *consistent data units.*

Anyway, a video sequence is either a single CDU or a collection of CDUs, but not both: this is not dependent upon the context, application or user. Tools can be designed to split a single CDU in several of them for composition or presentation purposes.

4.2 Multimedia systems

A multimedia data structure allows the integration of several CDUs by providing bases for mechanisms[1] to solve the problems presented in chapter 3. In order to understand what the design of such a structure deals with, let's describe a model for the typical multimedia systems.

4.2.1 Model

A multimedia system is a distributed system comprising (in terms of functionalities):

1. *user interfaces*: typically, workstations in the system run user interfaces and integrate hardware facilities to display and input audio and video. These workstations provide functions for local manipulation of CDUs: synchronization[2], composition, compression/decompression, coding, raster operations, geometric functions, and functions for conversing with the remainder of the system. The user interfaces manage the interactions with the end user.
2. *multimedia applications hosts*: workstations and software modules on the network that run the multimedia applications[3]. Each of these applications works with given user interfaces and accesses the multimedia servers and network services through functions implemented on the application hosts.
3. *networks*: on these networks, protocols are designed to be able to give guarantees on some typical communication parameters: data rate, error rate, end-to-end delay, delay jitter (see [VZF91, VF90]), stream skew. A lot of research is currently achieved in that field. We assume we can use reliable network services that provide bounds for the above parameters. We make no assumption about the underlying networks.
4. *multimedia servers*: with mass storage devices, such servers can handle documents and CDUs. They are responsible for database

[1] *mechanism* stands for "basic algorithmic tool".

[2] *synchronization* is used here as a generic term to point out a set of functions that allow the definition of a timed dependence relation between CDUs.

[3] chapter 2 describes some of theses applications.

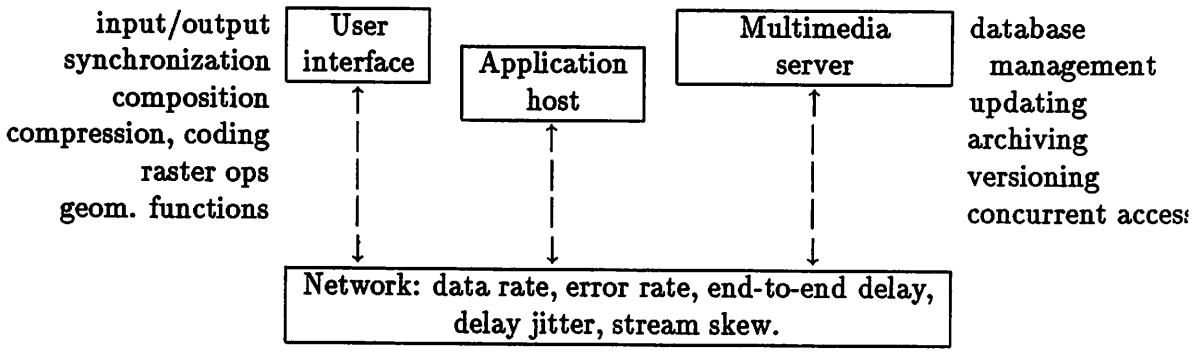


Figure 4.1: Multimedia system model (functionalities)

management, updating, archiving, versioning, concurrent access management, protection. Any underlying database management system can be used and many servers can coexist. Applications call primitives to use the services provided by multimedia servers.

Of course, this model, no particular implementation being supposed here, allows the coexistence of several of the four major parts described above on the same physical devices. Figure 4.1 summarizes that model.

Thus, the functionalities of a multimedia system are:

- user interface
- communication
- document management
- application.

The remainder of this chapter deals with data structures used by user interfaces, applications and multimedia servers. The interface to the multimedia servers and network services is achieved through a communication process to which the structures are sent. Thus, our model is reduced to three major components (see figure 4.2):

- user interface/application
- multimedia server
- communication process.

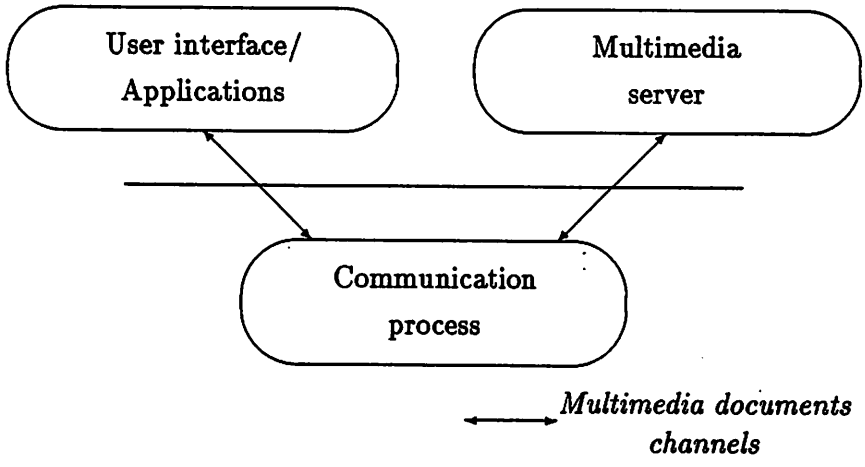


Figure 4.2: Multimedia system model

4.2.2 Correspondence with the OSI model

The correspondence between our model and the OSI reference model (ISO/IS 7488) is not immediate. The network entity suits the five first layers of the OSI model. But, the user interface, application hosts and multimedia servers are understood in the previous subsection as comprising the services provided by the presentation and the application layers of the OSI model and not only these of the application layer. The presentation layer, in the OSI model, has four main functionalities:

- providing the application layer with means to access the services of the session layer
- providing means to specify complex data structures
- managing the most frequently used data structures
- carrying out the conversions between internal and external representation.

The five first layers do not take the syntactical representation of the transferred information into account, this layer does. The coding, compression, protection and encryption functions are part of this presentation layer. All the document composition and CDUs synchronization

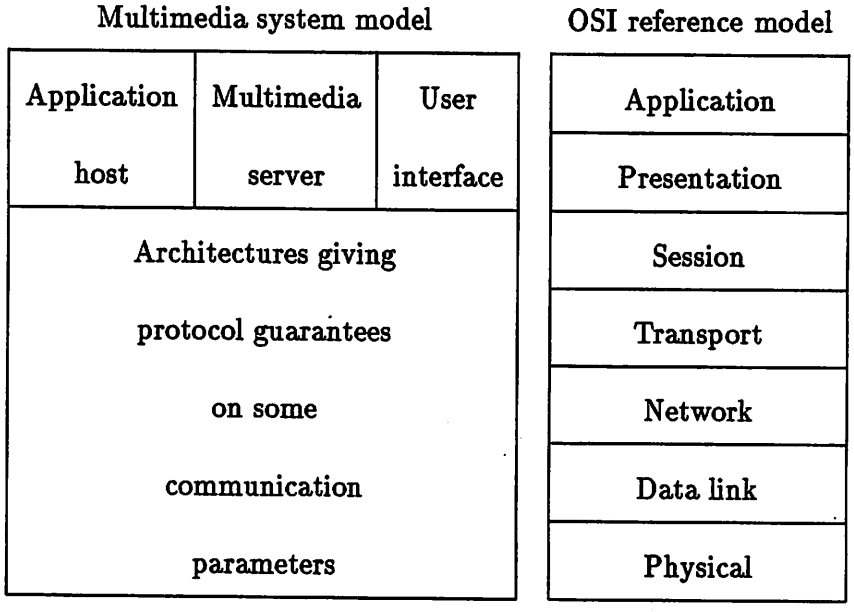


Figure 4.3: Correspondence with the OSI model

are realized in this layer. Thus, what we call application host, multimedia server and user interface are in fact two-layers entities. The first of these layers is the presentation layer that provides services to the application layer.

The first layers of our entities communicate between each other through service calls to 'their' presentation layer. The implementation of the latter can take into account the fact that several of these entities can be on the same machine and thus can realize the communication services without calling the lower layers services.

Figure 4.3 makes the correspondence between our model and the OSI model.

4.3 Local definitions of general notions

As everyone has his own definitions of general notions such as synchronization, sequencing and real-time, and in order to clarify the remainder of this paper, we give our own acceptation of these notions. These

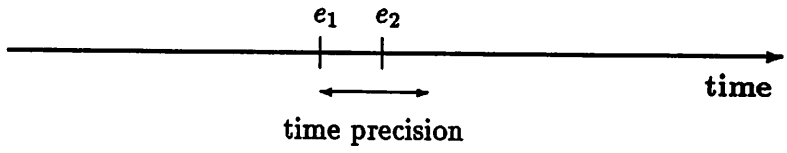


Figure 4.4: Simultaneity

definitions may lack strong formalism, but is not necessary here.

For the remainder of this section, we call *temporal distance* the application that to any couple of events (e_1, e_2) associates the absolute value of the difference between the two events' date of occurrence t_{e_1} and t_{e_2}. Practically, the temporal distance is related to a time precision.

4.3.1 Simultaneity

Given a non-relativist system, we first consider the relation between two events. We refer to absolute time as the time (in the instant acceptation) given by a GMT synchronized clock in the given system. The two events will be said to occur *simultaneously* for a given observer in the system given a *time precision* (duration acceptation) *iff* this observer can watch them at two dates separated by a duration less than the time precision δ (see fig. 4.4), ie $\mid t_{e_2} - t_{e_1} \mid \leq \delta$.

We insist on the fact that the considered time precision δ is not related to time measurement but to an application context. For example, a video segment and its associate sound segment must have a simultaneous display within a few milliseconds, whereas a graphic and its explicative text can be displayed simultaneously within one second without being a discomfort for the end user.

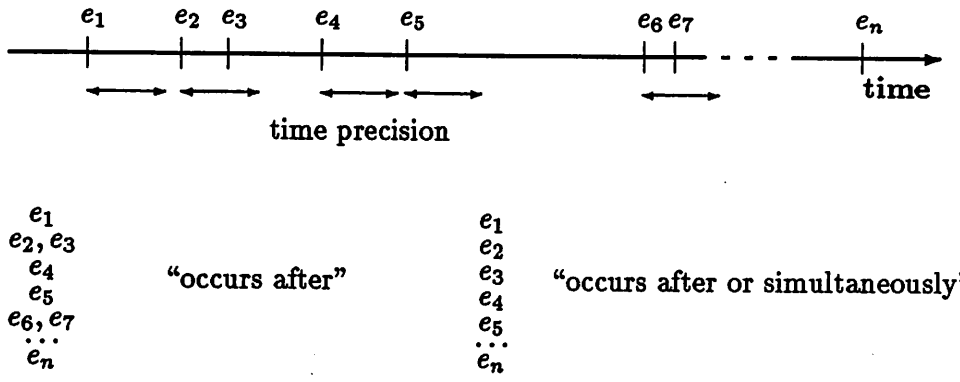


Figure 4.5: Sequencing

4.3.2 Sequencing

Let's define the order relation *"occurs after"*. The event e_1 *"occurs after"* the event e_2 *iff* the duration separating the date of occurrence of e_2 and the date of occurrence of e_1 is superior to the given time precision, ie $(t_{e_2} - t_{e_1}) > \delta$ (see fig. 4.5). By extension, the relation *"occurs after or simultaneously"* is obviously defined. Then, if we consider a set of events $\{e_1, e_2, \ldots, e_n\}$, we call *sequencing* the partial order given by the relation *"occurs after or simultaneously"* (see fig. 4.5).

4.3.3 Real-time

Real-time is sometimes understood as the ability for an information system to compute the received data as soon as they arrive, so, very often, before some more data arrive. Thus, the faster a computation is achieved, the more real-time capacity the system has. Our own definition follows.

Lets $E = \{e_1, \ldots, e_n\}$ be a set of events. Any event e_i ,$i \in \{1, \ldots, n\}$,

implies two other events: e_i' is called *beginning of the computation implied by* e_i and e_i'' is called *end of the computation implied by* e_i. To each event e_i, we associate a *deadline* τ_i. Given a time precision, the event e_i is said to be processed in *real-time* if the deadline τ_i *"occurs after"* the event e_i''. By extension, the set E is said to be processed in *real-time* if $\forall i \in \{1, \ldots, n\}$, e_i is processed in real-time (see fig. 4.6.a).

Restrictions can be added to this definition so as to meet particular requirements. Thus, let's consider the case when $n = +\infty$. In some particular contexts, the set $T = \{\tau_i, i \in N\}$ can be defined relatively to E by the relation: $\forall i \in N, (\tau_i + \delta) \geq t_{e_{i+1}'}$, thus expressing the hypothesis "$\forall i \in N, e_{i+1}$ *occurs after* e_i''", ie the computation implied by e_i is processed before the event e_{i+1} begins to be processed (see fig. 4.6.b).

Obviously, other restrictions can be stated, but a common definition of real-time can be our restricted definition mainly in the field of embedded applications.

4.3.4 Synchronization

Let E be the set of events $\{e_1, \ldots, e_n\}$, $n \in N$. The event e is said to be *synchronized* relatively to the set E *iff* a dependence relation, in which the time t is a parameter, between e and E is defined. In other words, e is synchronized relatively to E *iff* there exist a function f such that $t_e = f(t_E, t) = f(t_{e_1}, \ldots, t_{e_n}, t)$.

Temporal constraints can be added for example by imposing a temporal distance[4] between e and E. For example, it is possible to synchronize the display of a video frame relatively to the last displayed frame, to the first displayed frame or to a set of already displayed frames, thus playing on the possible corrections.

4.4 Basic definitions

We first present loosely the background of the remainder of this chapter and then give the basic definitions so as to express what we consider

[4]here, the temporal distance is different from that defined at the beginning of this section, but is a classical extension of it.

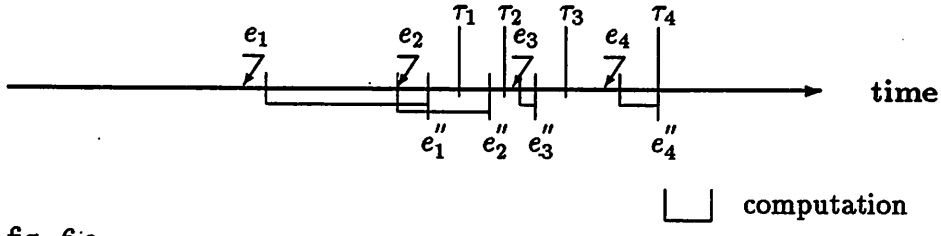


fig. 6.a

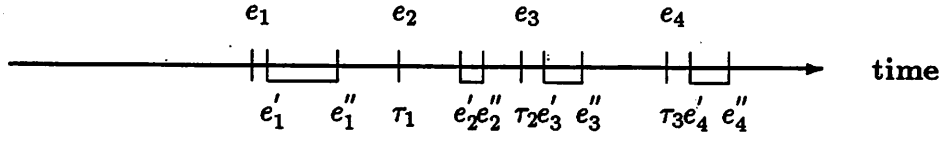


fig. 6.b

Figure 4.6: Real-time

a medium is.

4.4.1 Preamble

The data structure we define, as stated in chapter "Data Structure Requirements", is a descriptive structure that can be used either for preorchestrated multimedia documents retrieved for their presentation, or for sessions involving media with continuous synchronization as pointed out in section 4 of chapter 3.

For that, we define a hierarchical structure using the concept of *link*, which is a complex data structure defined in section 5. At given levels, it is possible to describe continuous synchronization and quantify it by specifying values for predefined attributes, while it is possible to describe preorchestrated documents thanks to the links. The coexistence of the two mechanisms allows a very flexible creativity for documents in which both of them are used. Furthermore, we do not make the difference between stored media and acquired media[5]. That way, it is easier to describe any combination of such media.

4.4.2 Definitions

In that part, we first try to formalize the notion of *consistent data unit* (CDU) from which the remainder originates. Then, we define the characteristics of CDUs on which the definition of *consistent medium* relies.

- Definition 1 : Consistent Data Unit
  We call *consistent data unit* an amount of data that can be handled as a consistent data stream by given functions. This is an indivisible atomical element of information. The *void* CDU is defined as carrying no information.

  examples:
  - a video sequence[6],
  - a text segment,
  - a sound sequence[6],
  - ...

[5] see chapter 1, section 5.2.

– Definition 2 : Class of a CDU

The *class* of a consistent data unit is an attribute of that data unit whose value is in the set

$$E_c = \{ \textit{static, dynamic} \}$$

Each class $c \in E_c$ is bijectively associated to a distance **d** on a given set **E** on which the CDU is mapped (for example, a text segment can be mapped on $\Re$). In other words, we associate bijectively to each class a metric space (**E**,**d**). Each metric space is associated to a set of units E_u. The distance **d** is expressed in one of the units contained in E_u. Note that by definition of a distance, the range of **d** is $\Re^+$.

example:

$$\begin{cases} \textbf{CDU} : & \textit{text segment} \\ \textbf{class} : & \textit{static} \\ \textbf{E} : & \Re^+ \\ E_u : & \{ \textit{bit, byte, bloc256, bloc512, ...} \} \end{cases} \qquad \begin{array}{ll} \textbf{d} : & \textbf{E}\times\textbf{E} \longrightarrow \Re^+ \\ & (x,y) \longmapsto | y - x | \end{array}$$

– Definition 3 : Presentation dimensional attribute

To each CDU we associate two couples : *(handling unit, precision)* and *(rate, precision)*. These two couples are together called *presentation dimensional attribute.*

The *handling unit* is a spatial unit defining the unit in the CDU manipulated by the presentation process. The *handling precision* is expressed relatively to the handling unit (so it is always ≥ 0 and ≤ 1) and defines an error rate in the displayed handling unit relatively to this same unit as it was when created or intentionally modified. For example, an image $320 \times 200 = 64000 \textit{pixels}$ with a handling precision of 0.0001 can be presented with 6 erroneous pixels.

The *rate* is a time unit associated to the handling unit and defining the presentation rate of the handling unit. The *rate precision* defines the maximum skew per handling unit in the presentation. For example, a video sequence presented at $1/30 \textit{ths}$ and with a

[6] here we make the hypothesis that a video stream or sound stream in general can be divided into bounded units.

Current CDUs	*Handling unit*	*Handling precision*	*Rate*	*Rate precision*
Formatted text	whole	0	0	0
Image	whole	0	0	0
Image	plan	0.001	0	0
Video	frame	0.001	1/30 s	2 ms

Table 4.1: Presentation dimensional attributes

specified rate precision of $2ms$ can be temporarily presented at a rate between $(1/30s - 2ms)$ and $(1/30s + 2ms)$.

The presentation dimensional attributes do not define a mandatory presentation but typical presentation parameters. For example, an application may apply a multiplicative factor on the rate.

examples: Table 4.1 gives a list of current CDUs and possible presentation dimensional attributes. These examples are not based on current applications. Note that formatted text can be associated to a given handling precision, depending on the importance or the redundancy of the text.

– Definition 4 : Type

We associate to each CDU a *type*. This type is a structure that contains the following:

- the *class* and the associated metric space instantiated with a unit and a precision. This is implicitly given by a couple *(unit, precision)*
- a *length* expressed in the class unit and on which the precision applies
- a *presentation dimensional attribute*
- a tuple of *handling functions*.

Thus the type associated to a CDU is *((class,(unit, precision)), length, ((handling unit, precision),(rate, precision)), (function)$_n$)*. The precision of the class defines a sphere around each point of the metric space associated to a class. In other words, the precision is the radius of the sphere, the center being the considered point.

A *handling function* is any program that can use the considered CDU. In the type definition, the $(function)_n$ tuple is composed with tuples uniquely identifying the considered handling function. Handling functions are for instance operating systems primitives, network services and other functions.

examples:

$$\begin{cases} \textbf{CDU} & : & \textit{formatted text} \\ \textbf{class} & : & \textit{static} \\ \textbf{(unit, precision)} & : & (\textit{byte}, 0) \\ \textbf{length} & : & 2050 \\ \textbf{(handling unit, precision)} & : & (\textit{whole}, 0) \\ \textbf{(rate, precision)} & : & (0, 0) \end{cases}$$

$$\begin{cases} \textbf{CDU} & : & \textit{video sequence} \\ \textbf{class} & : & \textit{dynamic} \\ \textbf{(unit, precision)} & : & (\textit{second}, 1ms) \\ \textbf{length} & : & 5.72 \\ \textbf{(handling unit, precision)} & : & (\textit{frame}, 0) \\ \textbf{(rate, precision)} & : & (1/30s, 2ms) \end{cases}$$

- Definition 5 : Consistent medium

  A *medium* is a typed consistent data unit. So as not to be trapped by the discussion around the fuzzy general meaning of the word *medium*, it is also called *consistent medium*. The void CDU allows the definition of the *void consistent medium*.

  The consistent medium thus defined is the basic data structure for the remainder of this chapter.

  Remark on definition 5:

  * In that definition, it is understood that the notion of medium is relative to the considered CDU. Thus, if the CDU is a pixmap, the medium is the image contained in this pixmap. But from pixmaps ordered together as a single CDU, video segments medium can be built. Some will argue that such a an adaptive definition is not desirable. If you think so, just remember to change the word "medium" for the phrase "typed CDU" in the remainder.

- Definition 6 : Multimedia Application, Multimedia Document

A *multimedia application* is an application that handles several consistent media. These consistent media are linked together in a data structure called *multimedia document.* This structure can provide many bases for various functions and mechanisms.

4.5 Multimedia data structure definition

In that part, we give a definition of *multimedia document.* It remains independent of a possible implementation and some aspects, such as the formal definition of links, are presented in the next section.

4.5.1 Definition

A *multimedia document* is a complex data structure consisting of a set of consistent media as well as multimedia documents[7] called *subdocuments.* We refer to any of these consistent media or subdocuments as a *constitutive unit* of the multimedia document. Thus, a multimedia document will contain a list of constitutive units (or units list), a list of links between the constitutive units and other multimedia documents, and other attributes.

The *void document* is defined as a document with an empty list of constitutive units. A document can have an empty list of links, but, as links refer to constitutive units in the units list, a document cannot have an empty constitutive units list with a non-empty list of links. The units list should at least contain the void consistent medium.

The list of units defines a set of media and documents with simultaneous presentation. More precisely, the events *beginning of the presentation of constitutive unit* i, $i \in \{1, .., n\}$, where n is the number of units in the list, are simultaneous as defined in section 3. This definition implies that each *multimedia document* has an attribute called *document precision* whose value is the temporal precision associated to the definition of simultaneity.

An attribute called *continuous synchronization attribute* (CSA) gives the list of the units that need continuous synchronization between each other. Thus, this attribute is a tuple of couples

[7] see definitions 5 and 6 previous section.

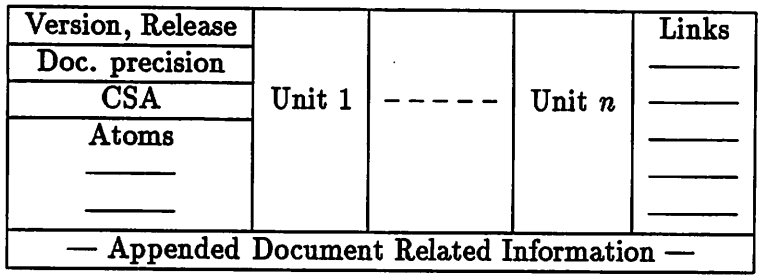


Figure 4.7: Multimedia document

$((medium\ x, medium\ y), (medium\ x', medium\ y'), \ldots)$.

A multimedia document has other attributes called *atoms*. Atoms[8] are couples *(name, value)*. The atoms belong to the document and two atoms with the same name can belong to two different documents. They are known (unless a general agreement) only from the interested applications and each name must be unique in a given document. Atoms allow application dependent extensions of the semantics of the multimedia document structure. Each name must be unique in a given document.

Finally, it is possible to append some other 'document related' information. This possibility is given to allow the extension of the structure to code transportation[9].

4.5.2 Implementation attributes

Even if the notions we have presented till now are implementation independent, it is important to consider the fact that two implementations of the same structure can coexist without being totally compatible. In that perspective, we define two attributes identifying the implementation that should form the head of the document. These attributes are *Version* and *Release*.

Figure 4.7 gives a diagrammatic summary of what a consistent document is.

[8] see [Nye90b] for an analogy to the Xwindow system.
[9] see chapter Data structures requirements.

4.5.3 Considerations for interactivity

During the presentation of a document, the end user's actions can be expressed in terms of *events*. These events can be associated to a *date* and a *location* in the document. Both the spatial and temporal dimensions can be significant for both static and dynamic CDUs. It is important to see that the class of a CDU is independent from the interactivity considerations, but, as some authors don't agree with that, we give two examples.

For instance, a word or a phrase in a text can be linked to another document in such a way that it has no more significance after a given period of time. For example, an advertising text for the concert season of a symphonic orchestra can mention several authors and works for which a few words are the root of a link to a reservation form. After the date of the last performance of a given title, the link is out-of-date. So we have here a static CDU for which the temporal value of an event has some importance.

Another example is interactive video[10]. For example, a real estate agency can build multimedia documents for the visit of houses. The client is first presented the entrance, and the fact that a given door is selected (through any pointing device) leads to a video sequence about the room behind the door. The presentation of the entrance can either be the beginning of a guided tour with no client actions, or the information displayed while waiting for an event from the client. Here, we have a dynamic CDU (the video sequence showing the entrance) for which both spatial and temporal event aspects are important.

From these two examples, it is obvious that any event occurring during the presentation of a document needs to be expressed in terms of time and location regardless to the class of the medium being presented.

[10] currently, most of the time, video browsing is called *interactive video* and should not be confused with what we are talking about.

4.6 Links and related structures

4.6.1 Definition

A *link* is a complex data structure relative to a given multimedia document. It is composed of two lists: the *link root* is a list of units from the given multimedia document with interactive attributes and among which a logical relation is defined. The *link target* is a list of multimedia documents with presentation attributes and among which a logical structure is defined.

4.6.2 Link root

Let us first define the notion of *trigger* which is needed for the definition of *link root.*

- Definition 7: trigger

  In a medium, a *trigger* is typically described by a couple *((date, precision),(location, precision))* relatively to the constitutive unit *Unit_ID*. This defines an area which is a sphere with the *location* as the center and the *precision* as the radius which is *active* at the given *date* given a time *precision.*

  If the constitutive unit is a medium, the range of *date* is in $\Re^+ \cup \{end, \infty\}$ where *end* is the period of time between the dates of occurrence of the events *beginning of the presentation of unit Unit_ID* and *end of the presentation of unit Unit_ID*[11], and where ∞ is the infinite period of time from the date of occurrence of *beginning of the presentation of unit Unit_ID*. The range of *location* is the same as the range of *date.* The value *end* is the length of the unit and ∞ means *anywhere.*

  The *void trigger* is defined as a trigger in the void medium. The void trigger is never active.

  This notion is extended to subdocuments by restriction of the semantics. The location has no meaning and is understood, whatever its value, as *anywhere.* The definition of more complete semantics will be part of future work.

[11] this is the presentation duration that can be associated to an *"end exception"*.

Unit_ID		
date,	precision	triggered mode
location,	precision	triggering mode
other attributes		

Figure 4.8: Trigger

Each trigger has attributes one of which is called *triggered mode* and has values in the set $\{stopped, unstopped\}$. The semantics of this attribute are:

* when *stopped*, the presentation of the unit is suspended
* when *unstopped*, the presentation of the unit goes on.

Another attribute is called *triggering mode* and defines the state of activity of the trigger with values in the set $\{automatic, non-active, user\}$ The semantics attached to this attribute are:

* when *automatic*, the trigger is considered as active when the area defined by the attributes $(date, precision)$ and $(location, precision)$ is active
* when *non-active*, the trigger is never active
* when *user*, the trigger is active when the area defined by the attributes $(date, precision)$ and $(location, precision)$ is active and when, in this active area, an event described in *other attributes* occurs.

Figure 4.8 gives a diagrammatic description of a trigger.

– Definition 8: link root

The *link root* is a binary tree. The non-terminal nodes contain logical information from the set $\{OR, AND, NOT\}$ and are called *logical nodes*. Each subtree of such nodes have the same semantics as the link root. The terminal nodes contain triggers and are called *trigger nodes*.

A subtree including only one trigger is said *active iff* the trigger is active. The *void subtree* is defined as a terminal node containing the void trigger and, as a consequence, is never active.

To each logical node attributes are associated, one of which is called *logical attribute*. In each trigger node *trigger attributes* can be defined.

Let's define the semantics of the logical node for the generic tree, ie with two subtrees and one logical node. In case the logical attribute is:

* OR, the tree is active *iff* one of the two subtrees is active. This is not an exclusive OR
* AND, the tree is active *iff* both the two subtrees are active
* NOT, the tree is active *iff* both the two subtrees are not active. Note that one of the two subtrees can be void.

As a consequence of these semantics, a subtree containing a trigger is active *iff* the trigger is active.

One of the logical node attributes is a *duration* which defines a closed time interval. The lower bound of the latter is the date of occurrence of the event *"becomes active"*. The measure of this interval is *"duration"* and upper bound coincides with the date of occurrence of the event *"becomes no more active"*.

Figure 4.9 gives an example of a link root[12].

Thus, a link root allows to define a complex event occurring in the presentation of a document by composition of elementary events.

4.6.3 Link target

The *link target* is an oriented binary tree. The non-terminal nodes contain synchronization information from the set $\{SIM, THEN\}$ and called *synchronization nodes*. The terminal nodes are documents. The *void subtree* is defined as a terminal node containing the void document.

Each synchronization node is associated to *synchronizing attributes* that define the dependence relation between the triggers of the link and the documents. One of these attributes is the couple *((delay, precision), (continuance, precision))* defining a temporal distance between the subtrees of a synchronizing node. To each terminal node, linking attributes are associated.

Let's define the semantics of the synchronizing nodes. In case such a node contains:

[12] the triggering mode is considered as *automatic* in that example.

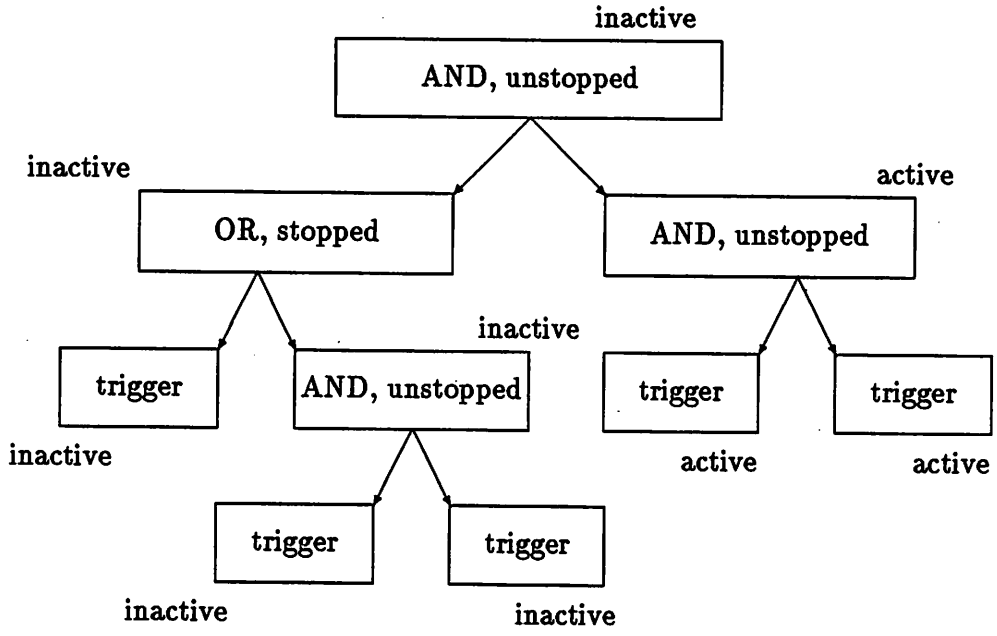


Figure 4.9: Link root (example)

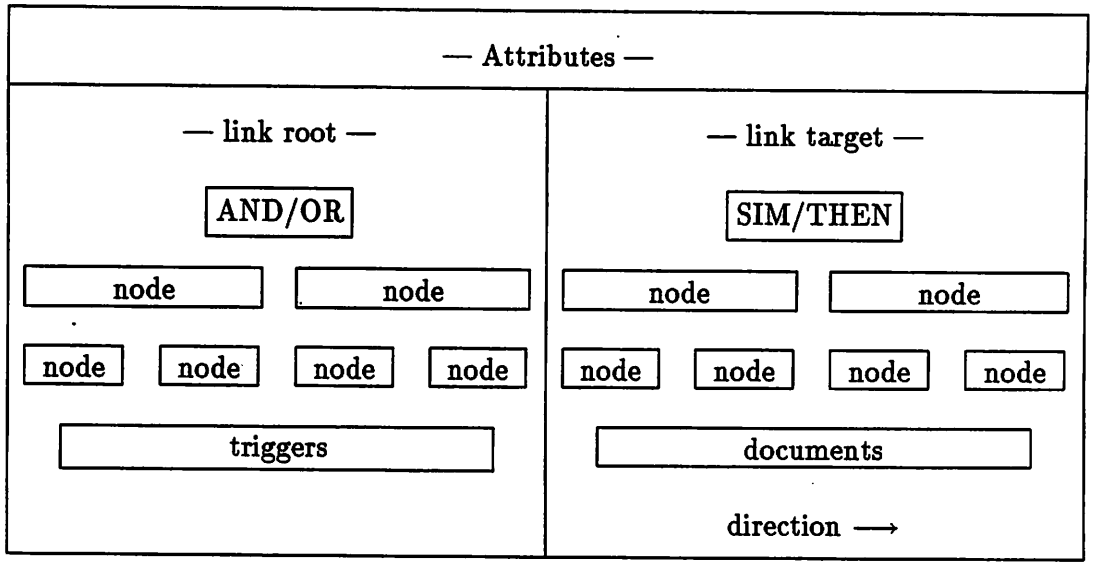


Figure 4.10: Link

- SIM, the first subtree, according to the orientation of the tree, is presented relatively to the event *activation of the link root* after the duration *delay* given the associated *precision.* The second subtree is presented relatively to the beginning of the presentation of the first subtree after the duration *continuance* given the associated *precision*
- THEN, the first subtree, according to the orientation of the tree, is presented relatively to the event *activation of the link root* after the duration *delay* given the associated *precision.* The second subtree is presented relatively to the end of the presentation of the first subtree after the duration *continuance* given the associated *precision.*

Figure 4.10 gives a diagrammatic summary of a link.

4.7 Basic justification

In that part, we intend to show that our structure is at least enough to describe the formal basic synchronization schemes between two CDUs. Thus, we use the presentation of simple synchronization in chapter 3 section 3.4.2. For each of the temporal

basic relationship, we give a way to achieve the construction of the structure describing the synchronization scheme. Note that it is possible to define a translation from the extended Petri nets in [LG90b] to our data structure with very simple rules.

The list of the basic schemes and the related data structures follow (the CDU P_δ is called *void*). We do not give a value for each attribute in each structure .

P_α before P_β

* CDUs: P_α,P_β
* Documents:
  - Doc_1:
    1. unit_1 = consistent medium
       $CDU = P_\alpha$
  - Doc_2:
    1. unit_1 = consistent medium
       $CDU = P_\beta$
* Links in Doc_1:
  - root:

    trigger = (unit_1,
    $(end + \tau_\delta, p_\delta)$,
    1. (∞,∞),
    unstopped,
    automatic)
  - target: Doc_2

P_α meets P_β

* CDUs: P_α,P_β
* Documents:
  - Doc_1:
    1. unit_1 = consistent medium
       $CDU = P_\alpha$
  - Doc_2:
    1. unit_1 = consistent medium
       $CDU = P_\beta$

* Links in Doc_1:
  - root:
    trigger = (unit_1, (*end*, P_δ), (∞,∞), unstopped, automatic)
    1.
  - target: Doc_2

P_α overlaps P_β

* CDUs: P_α,P_β, *void*
* Documents:
  - Doc_1:
    1. unit_1 = consistent medium
       CDU = *void*
  - Doc_2:
    1. unit_1 = consistent medium
       $CDU = P_\beta$
  - Doc_3:
    1. unit_1 = consistent medium
       $CDU = P_\alpha$
    2. unit_2 = subdocument = Doc_1
* Links in Doc_1:
  - root:
    trigger = (unit_1, (τ_δ, p_δ), (∞,∞), unstopped, automatic)
    1.
  - target: Doc_2
* Links in Doc_3:
  - root: AND (*tree_1*,*tree_2*)(p_γ)
    trigger = (unit_1, (*end*, ∞), (∞,∞), unstopped, automatic)
    1. *tree_1*:

trigger = (unit_2,
2. *tree_2*: (end, ∞),
(∞,∞),
unstopped,
automatic)

· target: what follows

P_α during P_β

* CDUs: P_α,P_β, *void*
* Documents:
  · Doc_1:
  1. unit_1 = consistent medium
     CDU = *void*
  · Doc_2:
  1. unit_1 = consistent medium
     $CDU = P_\alpha$
  · Doc_3:
  1. unit_1 = consistent medium
     $CDU = P_\beta$
  2. unit_2 = subdocument ≐ Doc_1
* Links in Doc_1:
  · root:
  trigger = (unit_1,
  1. (τ_δ, p_δ),
  (∞,∞),
  unstopped,
  automatic)
  · target: Doc_2
* Links in Doc_3:
  · root: AND (*tree_1*,*tree_2*)(p_γ)
  trigger = (unit_1,
  1. *tree_1*: (end, ∞),
  (∞,∞),
  unstopped,
  automatic)

trigger = (unit_2,
(*end*, ∞),
2. *tree_2*: (∞,∞),
unstopped,
automatic)

· target: what follows

P_α starts P_β

* CDUs: P_α,P_β
* Documents:
  · Doc_1:
  1. unit_1 = consistent medium
     $CDU = P_\alpha$
  2. unit_2 = consistent medium
     $CDU = P_\beta$
* Links in Doc_1:
  · root: AND (*tree_1*,*tree_2*)(p_γ)

trigger = (unit_1,
(*end*, ∞),
1. *tree_1*: (∞,∞),
unstopped,
automatic)

trigger = (unit_2,
(*end*, ∞),
2. *tree_2*: (∞,∞),
unstopped,
automatic)

  · target: what follows

P_α finishes P_β

* CDUs: P_α,P_β, *void*
* Documents:
  · Doc_1:
  1. unit_1 = consistent medium
     CDU = *void*

· Doc_2:

1. unit_1 = consistent medium
   $CDU = P_\alpha$

· Doc_3:

1. unit_1 = consistent medium
   $CDU = P_\beta$
2. unit_2 = subdocument = Doc_1

* Links in Doc_1:

· root:

1. trigger = (unit_1, ((P_β'*length*−P_α'*length*), p_δ), (∞,∞), unstopped, automatic)

· target: Doc_2

* Links in Doc_3:

· root: AND (*tree_1*,*tree_2*)(p_γ)

1. *tree_1*: trigger = (unit_1, (*end*, ∞), (∞,∞), unstopped, automatic)
2. *tree_2*: trigger = (unit_2, (*end*, ∞), (∞,∞), unstopped, automatic)

· target: what follows

P_α equals P_β

Easily built from P_α *starts* P_β and P_α *finishes* P_β.

Several of the presented schemes have equivalent model using Object Composition Petri Nets. The difference is in the restriction level of the considered hypothesis. Thus, P_α *starts* P_β and P_α *equals* P_β have the same modified Petri net model, but the former's hypothesis is restricted for the latter by $\mid \tau_\beta - \tau_\alpha \mid \leq \delta$,

where δ is a precision associated with the definition of simultaneity between the dates of occurrence of the events *end of* P_α *presentation* and *end of* P_β *presentation.* In our structure, such restrictions are described using the atoms. For instance, an application using the relation P_α *equals* P_β will generates a structure P_α *starts* P_β and defines the following atoms:

* ((*appl_name*)_EQUALS_(*relation_ID*)),Unit$_\alpha$_ID
* ((*appl_name*)_EQUALS_(*relation_ID*)),Unit$_\beta$_ID
* ((*appl_name*)_EQUALS_(*relation_ID*)),precision_value

Thus, it is possible to define four basic schemes families among the seven schemes with an hypothesis for each family and a restriction for each scheme:

1. P_α before P_β
2. P_α meets P_β
3. P_α overlaps P_β,P_α during P_β,P_α finishes P_β
   with $\mid \tau_\beta - (\tau_\alpha + \tau_\delta) \mid \geq \delta$ for P_α *during* P_β
   and $midtau_\beta - (\tau_\alpha + \tau_\delta) \mid \leq \delta$ for P_α *finishes* P_β
4. P_α starts P_β,P_α equals P_β
   with $\mid \tau_\beta - \tau_\alpha \mid \leq \delta$ for P_α *equals* P_β

In Appendix 2, wee show an algorithm for mapping OCPNs on our structure.

4.8 Conclusion

The design of a data structure must take various issues into account (as presented in chapter 3). Assuming that these issues can be independently considered, we focus on synchronization problems. This leads to the definition of a structure with several levels of synchronization links:

* a document contains media and other documents; this is the *hierarchical* linking
* a document can be linked to others: this is the *orthogonal linking.*

The concepts presented here can be extended and be part of future work. Thus, the notion of trigger should be extended more

completely to subdocuments (see section 4.6.2) and not only defined by restriction of the semantics. Moreover, the other aspects presented in chapter 3 need to be described:

* by extension of the semantics (to solve partially spatial composition for example)
* by extension of the structure itself (to solve the problem of protection for example)

The issue of compression has to be analyzed so as to make the distinction between structural aspects and application dependent aspects. Besides, some elements of the currently defined structure may be considered as application dependent by some readers. For each of these aspects, a precise and wide study is needed.

Chapter 5

Xwindow for Multimedia

5.1 Introduction

The Xwindow system can be considered in itself as a component of a multimedia system in that it handles text and graphics for presentation to the end user. At the beginning it was intended to provide a hierarchy of resizable windows and to support high performance device independent graphics. But the X protocol, which is device and network independent[1], is easily extensible.

Some attempts have been made to define an extension for video. Thus, workstations integrating video interfaces (see [Lud87]) have been developed and Xwindow used as an interface for video applications using VEX (see [Bru89]).

One of the Xwindow features that makes it difficult to use it directly as a user interface for multimedia is that the requests are processed asynchronously. The synchronous mode[2] is usually used as a debugging mode, for, the performances drop dramatically as soon as it is operated. However, this mode is not sufficient for multimedia.

5.2 The Xwindow system

Xwindow is a system on which any style of user interface can be built. It uses an asynchronous protocol, unlike most similar systems that use remote procedure control (RPC) or system calls. This allows:

* a network transparent connection on either a local or remote station running X, for both the user and the application
* a possible implementation on almost any hardware on which a reliable byte stream communication channel is provided.

The overhead due to the use of a protocol, even with local connections, does not significantly affect performances that are limited by the drawing time.

The success of X is due to the portability of the applications developed on it that, if properly written, will compile and run

[1] all needed is a reliable duplex byte stream (without urgent data) to transport the protocol requests, replies, events and error messages.

[2] synchronous here means that any request built by a client is sent immediately over the network.

immediately on any system on which X is available. Moreover, the code that implements X is freely available along with many applications[3].

5.2.1 Server and client

An *X server* is a program that is run on a computer and controls screens, a keyboard and a mouse[4] on this computer. A keyboard with a mouse and a set of screens are called a *display.*

An *X client* is a process that runs on a station on the same network as the server, sends requests to the latter so as to have it draw, execute specific actions on given resources or ask for some information maintained by the server. It receives, from the server, replies to the queries, events resulting from the user interaction with the display, and error messages.

Many clients can be connected to a single server and a single client can communicate with several servers. Figure 5.1 from [SG86] shows four applications displaying on a single screen controlled by an X server.

On multitasking systems, a client and a server can run on the same machine, but on non-multitasking systems, such as on PCs with MS DOS, only the server can run unless the addition of a multitasking layer. Some terminals called *X terminals* are widely available now that have the server software in ROM.

Some aspects of X, such as non-exclusive access to the resources by the clients, allow the design of window managers that can control the overall layout of the windows displayed on a given screen. This feature is a problem if the resources have to be secured.

The development of a client consists in building a program on a layer called *Xlib* that is a hardware-dependent library presenting a hardware-independent interface to the client. The latter can be developed partially on another layer, called *toolkit* layer that provides a simplified interface for the manipulation of complex object-oriented entities called widgets. Figure 5.2 shows how an application interfaces with the different layers.

[3] use `ftp export.lcs.mit.edu` and log as anonymous.

[4] in X11R4, the *mouse* is an input device with up to 5 buttons.

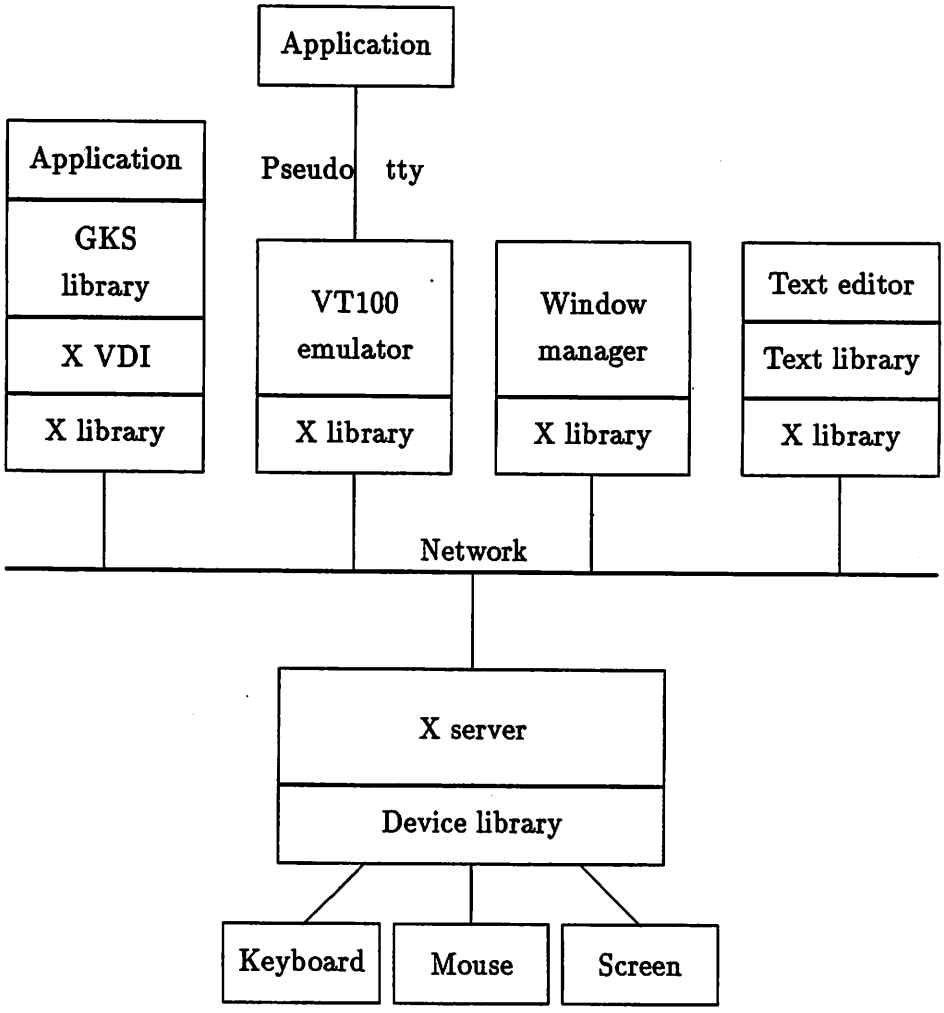


Figure 5.1: The Xwindow system

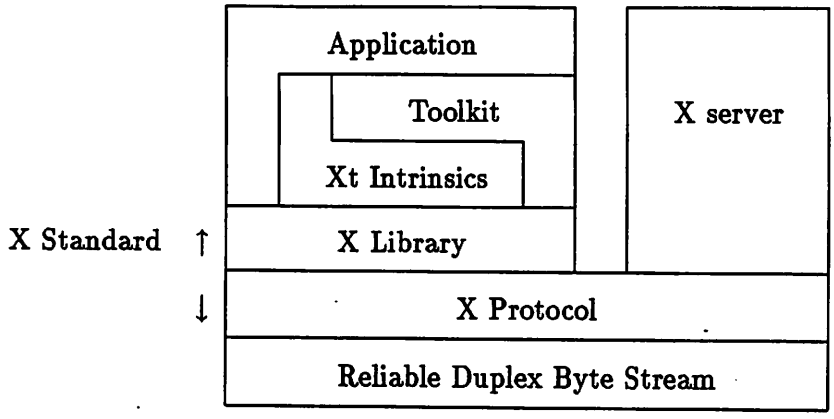


Figure 5.2: Software layers for X applications

5.2.2 The X protocol

The X protocol actually defines the Xwindow system. Many different implementations can qualify as an Xwindow system as soon as the protocol is properly implemented. Below the X protocol, any connection can be used as soon as it provides a reliable duplex byte stream. This can be, for example, when a client and a server run on the same machine, an interprocess communication channel (IPC), shared memory or a UNIX socket[5]. This protocol is asynchronous for the following reasons:

* a synchronous protocol would imply round trip, decreasing speed and the number of connections. Usually, the round trip speed is much less than the network speed. Such a policy allows continuous polling at the application level
* this allows the packetization of requests and, as a consequence, reduces the overhead information.

The communication between a client and a server is achieved on a single channel on which several windows can be multiplexed. Thus, the number of windows is limited, not by the allowed number of channels or file descriptors, but by the number of possible window IDs.

[5]the sample implementation from MIT of the C Xlib uses Berkeley UNIX sockets.

5.2.3 Transferred information

Four types of messages are used in the Xwindow system. Only the requests are sent by the client to the server, the replies, events and error messages being sent by the server.

* requests: carry information for resources allocation, resources modification, resources query, display functions, ... Requests requiring a reply are called *round-trip requests*
* replies: answer the round-trip requests
* events: contain information about a device action or a side effect of previous requests
* error messages: similar to events, but handled differently by the client. An error-handling routine of Xlib is used for such messages.

The X protocol messages always have a length multiple of 4 bytes to simplify the implementation on some specific machines. Figure 5.3 from [Nye90a] gives an example of an Xwindow client/server session.

5.2.4 Functional aspects of clients and servers

The server has the following responsibilities:

* managing the interfaces to the device drivers on its display
* managing server maintained resources or abstractions: windows, pixmaps, graphic contexts, off-screen memory, fonts, cursors, colormap
* sending the appropriate events to the client
* replying to the round-trip requests
* sending error messages when needed.

The client is responsible for using the right request for a given action. For example, it is not necessary to reset the graphic context all the time or to query for the same information many times.

5.3 Multimedia aspects

The multimedia system model presented in chapter 4 section 2 considers the user interface as a major component of the system.

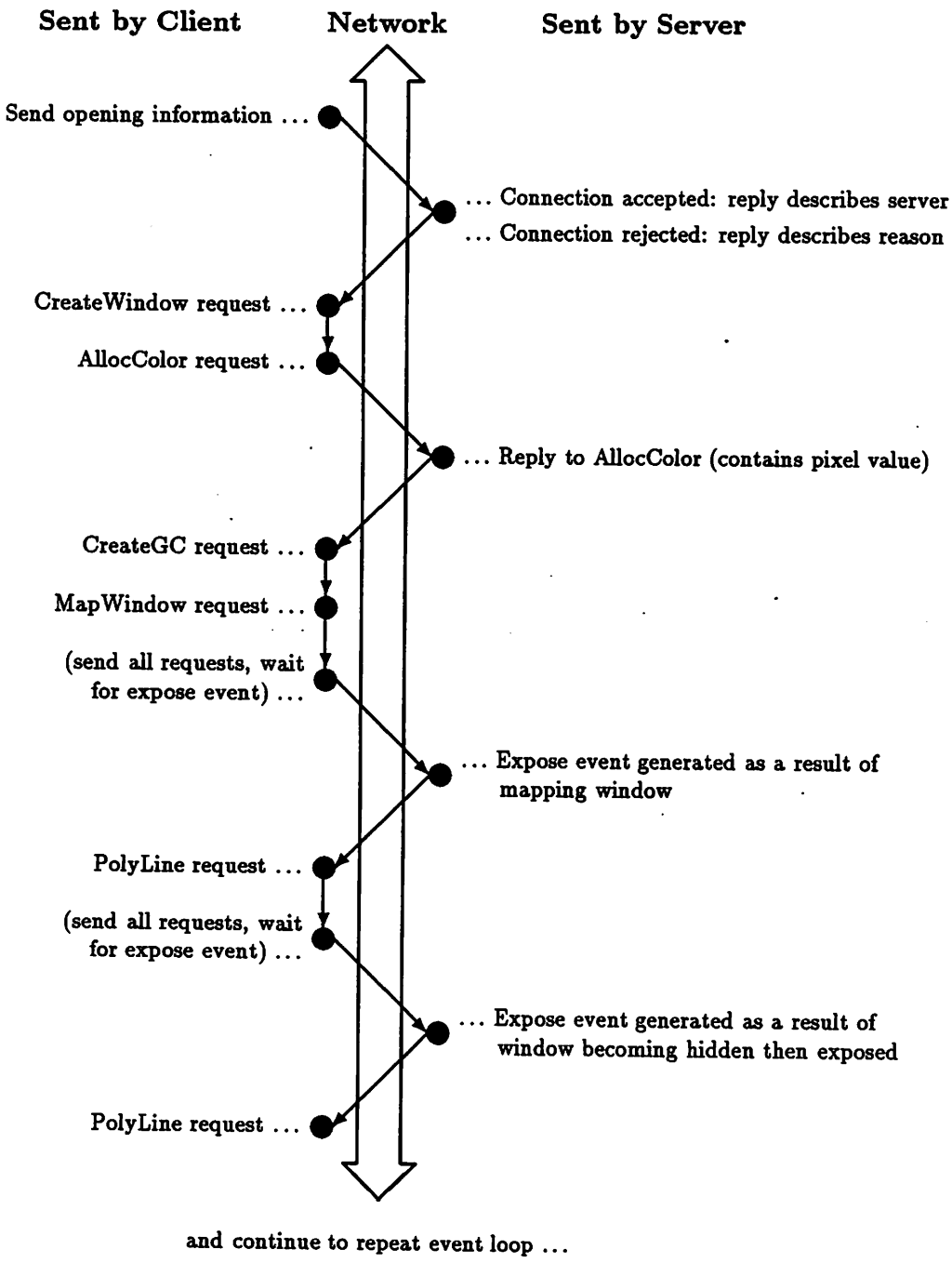


Figure 5.3: A sample Xwindow session

The Xwindow system qualifies partially for this. The functionalities provided by X and required by our model are:

* at the presentation layer level:
  - access to the session layer services, by construction of the Xwindow system
  - management of the most frequently used data structures such as windows, pixmaps, ...
  - conversions between internal and external representation of information.
* at the application layer level:
  - functions for the local manipulation of static CDUs[6] such as text, pixmaps
  - spatial composition functions and raster operations
  - management of the user interactions.

But some functionalities are missing for X to fully qualify as a multimedia user interface as presented in our model. Next section focuses on that.

5.4 Main issues

The aspects missing for Xwindow to be considered as a user interface entity as presented in our model are:

* at the presentation layer level:
  - the possibility to specify at runtime new complex data structures and have some of them maintained by the server (see second next subsection)
* at the application layer level:
  - functions for the display and input of audio and video. This should be part of the X11R5 version of X by extension of the X protocol
  - synchronization functions. This implies the implementation of the system on an operating system providing specific synchronization primitives. The asynchronous aspect of X is detailed in next subsection

[6] see definition in chapter 4 section 4.

- compression/decompression ability. This is implied by the ability to display high quality video and sound
- security functions/coding. The displayed information being sent on unprotected networks, coding functions can be required for confidential information handling. As stated in chapter 3 section 6, the access to data is another aspect of security. The X protocol connection information sent from the client to the server includes space for an authorizing name and associated data[7]. Another problem is the possible access of any application to other applications' resources

* at the protocol level (see next subsection).

The following subsections focus on three main issues in Xwindow for multimedia.

5.4.1 Asynchronous transmission

The X protocol is asynchronous for efficiency purposes. Indeed, when manipulating static CDUs such as text and pixmaps, the real-time and synchronization aspects of multimedia are not tight. This is not necessary because the end user does not feel much incommoded by delays and jitter in the presentation of independent static CDUs. But, for video and sound and for any kind of CDUs linked by a complex presentation structure, synchronization functions are required.

On many implementations, the X library queues the requests of the client and empties the queue only under one of the four following conditions:

* the client calls a blocking Xlib routine to get a special event, but no matching event is available in the arriving events buffer
* the client requests some information from the server, requiring an immediate reply
* the client asks for the explicit flush of the request queue by calling the `Xflush` Xlib routine

[7] a basic connection authorization scheme using the name MIT_MAGIC_COOKIE_1 has been implemented (from release 4).

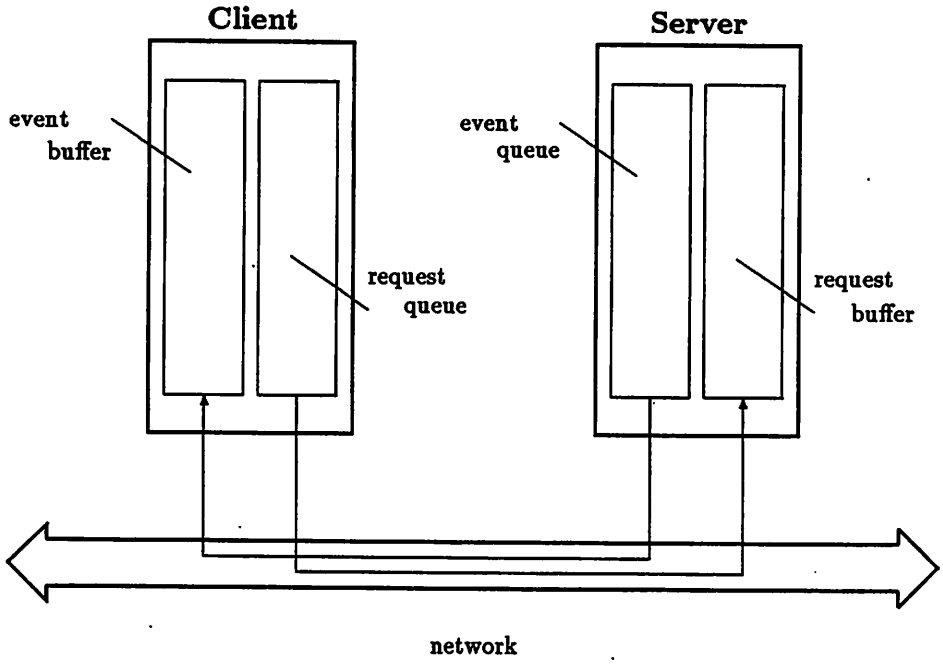


Figure 5.4: Queuing policy in X

* the request queue is full.

Such a policy makes the Xwindow communications asynchronous at the requests level.

In the server, such a queuing policy is implemented, too. The events are[8] queued and then sent over the network in a packetized way. The client buffers them till it can process them (see figure 5.4).

The `XSynchronize` Xlib call turns Xlib in synchronous mode, thus eliminating the queuing policy. As a consequence, the overhead rises up and the performances drop beyond the user's comfort requirements.

Thus, the functional aspects of the network entity of our model (section 4.2.1) could not be used by X11R4. The only parameter Xwindow takes into account is the error level: the underlying

[8] as stated previously, this is a possible implementation. Some implementations do not operate like this.

byte stream must be reliable[9]. Moreover, other aspects such as real-time requirements in the presentation mean the definition of special functions accessing underlying operating system services that usually are not available on the current platforms. But this is not only an Xwindow issue but a problem of definition and access to operating system primitives.

5.4.2 Distributed applications

Under Xwindow, an application sends requests to a given server. But, there is no basic mechanism for the distribution of the requests among a set of servers on the network. Thus, for the design of distributed Xapplications, three solutions are possible:

* design of an application handling multiple connections to a set of servers via another protocol, and maintenance of a list of connected servers by the application itself. A distributing module[10] has to be designed to send each request to each connected server and handle the events, replies and error messages of each connected server
* redesign of the Xwindow system by extension of the X protocol and addition of a dispatching module
* design of an X protocol interceptor for protocol distribution.

The first solution means the design of applications taking distribution into account and also the redesign of the current applications that, most of the time, set their only display server reference when started.

The second solution means the use of special servers for the distributing purposes and leads to non-portability of applications developed on it because of use of "non-classical" servers.

The third solution allows the use of the current applications, with no modification at all, for distributing purposes and does not imply modification of the development methods of Xapplications. This idea has been developed in [AWF91].

[9] static CDUs such as text and graphics do not tolerate errors whereas video or sound do.

[10] in the application.

5.4.3 Data structures

In the model presented, application/user interface entities provide functions for the presentation of synchronized CDUs and documents. This means the ability to maintain and apply specific functions at runtime on defined structures such as the consistent medium structure, or even on multimedia document structures. This implies the ability to describe these structures not only in terms of memory space, but of functions applied on them.

5.5 Conclusion

The Xwindow system provides many of the functionalities of the user interface entity described in the multimedia system model in section 4.3.2. But, several aspects are missing for it to fully qualify as a multimedia user interface as considered in our model: asynchronous protocol, video and sound are not supported, no possibility to define and have data structures maintained by the server, no support for distributed applications.

Anyway, Xwindow has been adopted as a virtual standard by a large part of the computer industry in the world and it is the most adaptable user interface available on current workstations. This makes it important to study the improvements that are necessary for multimedia.

5.6 Keywords

Xwindow, multimedia system, multimedia user interface, asynchronous protocol, queuing, distributed application

Conclusion

Multimedia systems are expected to cover many fields in computer science: a first attempt to understand their inherent properties has been given in this paper by discussing them and presenting some of their implications concerning hardware, applications and communications.

In order to study the logical aspect of multimedia composition, leading to complex data structures, a review of the current applications is given in chapter 2. This shows that many of the resulting problems are related to synchronization in the presentation of several related documents of different nature, and, as a consequence, to the ability to give guarantees on performances.

The manipulation of a multimedia document implies the definition of system primitives and protocols giving guarantees on synchronization, on real-time communication, as well as, using several protection levels, real-time compression/decompression and versioning. After stating that these aspects can be independently considered, we focus on the presentation of synchronized consistent data units, and thus define a complex data structure with several levels of synchronization and links. The study starts from the definition of basic models for general multimedia systems and discusses their composition.

The interaction with the user and external world is taken into account through structures defining windows in time and space: the triggers. The user interactivity is here taken as a main point of multimedia systems.

The defined structure is considered as extensible and should at least contain the information required to implement the aspects of multimedia presented in chapter 3. This can be achieved either by extension of the semantics of the structure, or by extension of the structure itself, a balance between these two solutions needing to be determined.

A study of the ability of one of the most used user interface to qualify as a multimedia user interface is presented in last chapter. Xwindows provides many of the needed functionalities, but several aspects are missing regarding synchronization.

Future work will focus on some of the aspects presented in chapter 3 and 4 so as to complete the data structure. This will take place in a wider definition of a multimedia system in which each layer

of the basic model will be developed. Many of the aspects of this model are already being investigated in both LAAS and UCB.

Bibliography

[AWF91] H.M. Abdel-Wahab and M.A. Feit. Xtv: A framework for sharing xwindow clients in remote synchronous collaboration. In *TRI COMM'91*, 1991.

[BERS89] N.S. Borenstein, C.F. Everhart, J. Rosenberg, and A. Stoller. Architectural issues in the andrew message system. In E. Stefferud, O-J. Jacobsen, and P. Schicker, editors, *Message Handling Systems and Distributed Applications*. North-Holland, 1989.

[Bru89] T. Brunhoff. Vex: Version extension to x. Tektronix, Inc, 1989. Version 5.5.

[Com91] Communications of the acm, Apr 1991. volume 34, number 4.

[Con87] J. Conklin. Hypertext: An introduction and survey. *Computer*, Sep 1987.

[EAC+91] J.R. Ensor, S.R. Ahuja, R. Connaghan, D. Horn, M. Pack, and D.D. Seligmann. Control issues in multimedia conferencing. In *TRI COMM'91*, 1991.

[Edw91] W.K. Edwards. The design and implementation of the montage multimedia mail system. In *TRI COMM'91*, 1991.

[Fox91] E.A. Fox. Standards and the emergence of digital multimedia systems. *Communications of the ACM*, 34(4), Apr 1991.

[GP91] J. Grimes and M. Potel. What is multimedia. *IEEE Computer Graphics and Applications*, 11(1), Jan 1991.

[GYN90] N.D. Georganas, R. Yatsuboshi, and N. Naffah. Multimedia communications. *IEEE Journal on Selected Areas in Communications*, 8(3), Apr 1990.

[HG91] E.M. Hoffert and G.Gretsch. The digital news system at educom: a convergence of interactive computing, newspaper, television and high-speed networks. *Communications of the ACM*, 34(4), Apr 1991.

[HK86] W. Horak and G. Kronert. Document editing and entry based on the office document architecture standard ecma 101. Private Communication Systems and Network Division, 1986. Siemens AG, Munich.

[HKL+91] K. Harney, M. Keith, G. Lavelle, L.D. Ryan, et al. The i750 video processor - a total multimedia solution. *Communication of the ACM*, 34(4), Apr 1991.

[HSS90] D.B. Hehmann, M.G. Salmony, and H.J. Stuttgen. Transport services for multimedia applications on broadband networks. *Computer Communications*, 13(4), May 1990.

[IS-86] ISO IS-8613. Information processing - text and office systems - office document architecture. ISO Standard, 1986.

[JS91] K. Jeffay and F.D. Smith. System design for workstation-based conferencing with digital audio and video. In *TRI COMM'91*, 1991.

[JTC90a] ISO-IEC JTC1/SC2/WG11. Coding of moving pictures and associated audio. MPEG Video CD Editorial Comitee, Dec 1990.

[JTC90b] ISO-IEC JTC1/SC2/WG12. Coded representation of multimedia and hypermedia information. Multimedia and Hypermedia Information Coding Expert Group (MHEG), Working Document Version 2, Jul 1990.

[JTC91] ISO-IEC JTC1/SC2/WG10. Digital compression and coding of continuous-tone still images - part !, requirements and guidelines. Joint Photographic Experts Group (JPEG), Committee Draft 10918-1, Feb 1991.

[KG77] K. Kahn and G.A. Garry. Mechanizing temporal knowledge. *Artificial Intelligence*, 9(1), Aug 1977.

[KNS90] W. Klas, E.J. Neuhold, and M. Schrefl. Using an object-oriented approach to model multimedia data. *Computer Communications*, 13(4), May 1990.

[LBH+90] W.H.F. Leung, T.J. Baumgartner, Y.H. Hwang, M.J. Morgan, et al. A software architecture for workstations supporting multimedia conferencing in packet switching networks. *IEEE Journal on Selected Areas in Communications*, 8(3), Apr 1990.

[LeG91] D. LeGall. Mpeg - a video compression standard for multimedia applications. *Communication of the ACM*, 34(4), Apr 1991.

[LG90a] T.D.C. Little and A. Ghafoor. Network considerations for distributed multimedia object composition and communication. *IEEE Network Magazine*, Nov 1990.

[LG90b] T.D.C. Little and A. Ghafoor. Synchronization and storage models for multimedia objects. *IEEE Journal on Selected Areas in Communications*, 8(3), Apr 1990.

[Lub72] H.P. Lubich. Instants and intervals. In *Conference of the International Society for the Study of Time*, 1972.

[Lub91] H.P. Lubich. A proposed extension of the oda model for the processing of multimedia documents. In *TRI COMM'91*, 1991.

[Lud87] L.F. Ludwig. Interview of the integrated media architecture laboratory. Technical Report 22251, Bell Communications Research (Bellcore), Sep 1987. Version 5.5.

[Mey89] N. Meyerowitz. The link to tomorrow. *Unix Review*, 8(2), 1989.

[MK88] L. Mohan and R.L. Kashyap. An object-oriented knowledge representation for spatial information. *IEEE Transactions on software Engineering*, 14(5), May 1988.

[MSS90] E. Moeller, A. Scheller, and G Schurmann. Distributed multimedia information handling. *Computer Communications*, 13(4), May 1990.

[Naf90] N. Naffah. Multimedia applications. *Computer Communications*, 13(4), May 1990.

[Nic90] C. Nicolaou. An architecture for real-time multimedia communication systems. *IEEE Journal on Selected Areas in Communications*, 8(3), Apr 1990.

[Nie90] J. Nielsen. *Hypertext and hypermedia.* Acadamic Press, Inc., San Diego, CA, 1990.

[Nye90a] A. Nye. *X Protocol Reference Manual*, volume 0 of *The Definitive Guides to the Xwindow System.* O'Reilly & Associates, Inc., Sebastopol, CA, second edition, May 1990.

[Nye90b] A. Nye. *Xlib Programming Manual*, volume 1 of *The Definitive Guides to the Xwindow System.* O'Reilly & Associates, Inc., Sebastopol, CA, second edition, May 1990.

[Oxf89] The oxford english dictionary, 1989.

[Rob90] Le petit robert, 1990.

[Sak90] S. Sakata. Development and evaluation of an in-house multimedia desktop conference system. *IEEE Journal on Selected Areas in Communications*, 8(3), Apr 1990.

[Sch89] P. Schicker. Message handling systems, x.400. In E. Stefferud, O-J. Jacobsen, and P. Schicker, editors, *Message Handling Systems and Distributed Applications.* North-Holland, 1989.

[SG86] R.W. Scheifler and J. Gettys. The xwindow system. *ACM Transactions on Graphics*, 5(2), Apr 1986.

[She90] T. Shetler. Birth of the blob - multimedia databases will radically change the way you look at and work with information. *Byte*, 15(2), Feb 1990.

[Ste90] R. Steinmetz. Synchronization properties in multimedia systems. *IEEE Journal on Selected Areas in Communications*, 8(3), Apr 1990.

[Tan90] A.S. Tanenbaum. *Reseaux - Architectures, protocoles, applications.* InterEditions, Paris, 1990. French translation of "Computer Networks" second edition, Prentice-Hall, Inc., 1989.

[TFC+85] R.H. Thomas, H.C. Forsdick, T.R. Crowley, R.W. Schaaf, R.S. Tomlinson, and V.M. Travers. Diamond: A multimedia message system built on a distributed architecture. *Computer*, Dec 1985.

[TMMN86] S. Tsuruta, K. Mitsuhashi, M. Mera, and K. Niwa. Intelligent communication terminal for integrating voice, data and video signals. *IEEE Something*, 1986.

[Tri83] R.H. Trigg. *A Network-based Approach to Text Handling for the Online Scientific Community.* PhD thesis, University of Maryland, 1983.

[UO91] K. Umemura and A. Okasaki. Real-time transmission and software decompression of digital video in a workstation. Technical Report TR-91-004, International Computer Science Institute, Jan 1991.

[VF90] D.C. Verma and D. Ferrari. A scheme for real-time channel establishment in wide-area networks. *IEEE Journal on Selected Areas in Communications*, 8(3), Apr 1990.

[VZF91] D.C. Verma, H Zhang, and D Ferrari. Delay jitter control for real-time communication in a packet switching network. Technical Report TR-91-007, International Computer Science Institute, Jan 1991.

[Wal91] G.K. Wallace. The jpeg still picture compression standard. *Communication of the ACM*, 34(4), Apr 1991.

[WK85] P.H. Wood and S.G. Kochan. *UNIX System Security.* UNIX System Library. Hayden Books, first edition, 1985. sixth printing, 1990.

Glossary

This glossary partly originates from [Fox91] and has been added the significant words and phrases defined in this report. It is not exhaustive, but sometimes useful when reading specialized articles about multimedia where the reader is expected to know the subject as well as the author.

* Acronyms
  - **A/D** Analog to Digital
  - **ADPCM** Adaptive DPCM
  - **ASIC** Application Specific Integrated Circuit
  - **ATM** Asynchronous Transfer Mode networks (based on packet switching)
  - **BPP** Bits Per Pixel
  - **BPS** Bits Per Second
  - **CCIR** International Radio Consultative Committee
  - **CCIR 601** recommendation for digital video
  - **CCITT** International Telegraph and Telephone Consultative Committee
  - **CD-ROM** Compact Disc Read Only Memory
  - **CD-ROM XA** CD-ROM eXtended Architecture (specifies ADPCM qualities
  - **CDU** Consistent Data Unit
  - **CIF** Common source Intermediate Format
  - **CLUT** Color LookUp Table (so user can select best set of colors from large palette)
  - **CMTT** Committee for Mixed Telephone and Television (joint CCIR/CCITT)
  - **CSA** Continuous Synchronization Attributes
  - **DAC** Digital to Analog (D/A) Converter
  - **DCT** Discrete Cosine Transform
  - **DPCM** Differential PCM (pulse code modulation): transmit value differences
  - **DSP** Digital Signal Processing/Processor
  - **DVI** Digital Video Interactive (TM of Intel Corporation)
  - **FDCT** Forward DCT (ie usual form of DCT)
  - **HDTV** High Definition TeleVision (eg 1125 or 1250 lines)

- **Hausdorff distance** measure used comparing images for fractal encoding
- **Huffman coding** static set of minimal redundancy integral length bit strings
- **HyTime** Hypermedia/Time based document representation language
- **IDCT** Inverse DCT
- **IEC** International Electronic Committee
- **ISDN** Integrated Services Digital Network
- **ISO** International Organization for Standardization
- **JPEG** Joint Photographic Experts Group
- **JTC1** Joint Technical Committee 1
- **KBPS** Kilo Bits Per Second
- **MBPS** Mega Bits Per Second
- **MHEG** Multimedia and Hypermedia information Coding Expert Group (SC2)
- **MSE** Mean Square Error (used in many coding methods as quality measure
- **NTSC** (US) National Television System Committe (TV standard using YIQ)
- **ODA** Office Document Architecture [IS-86]
- **PAL** Phase Alternating Line (TV standard in much of Europe using YUV)
- **PCM** Pulse Code Modulation (used for example with CD audio)
- **QCIF** Quarter Common source Intermediate Format (1/4 CIF,eg 180×144)
- **SAR** Synthetic Aperture Radar (2-D imaging with high data rates/volumes)
- **SECAM** SEquentiel Couleur Avec Memoire (TV standard developed in France)
- **SGML** Standard Generalized Markup Language - ISO 8879 (markup metalanguage)
- **SLDL** Simple Link Description Language
- **SMDL** Simple Multimedia Description Language

- **SMPTE** Society of Motion Picture and Television Engineers
- **SMPTE time code** standard
- **VLC** Variable Length Coding
- **VQ** Vector Quantization: mapping k dimensional real vector to nearest code book entry
- **YUV** color space used in PAL: Y luminance, 1.3 MHz chrominance components U,V

* Words and Phrases
  - **adaptive** dynamically adjusting (parameters) to data stream
  - **adaptive bit allocation** allocating more bits to high-activity image areas
  - **arithmetic coding** message symbol probabilities reduce interval size in [0,1]
  - **chrominance** color information of signal (eg UV of YUV, IQ of YIQ)
  - **code book** (eg for VQ) indicates mapping input to output encoding
  - **codec** coder/decoder
  - **component signal** a separated part (eg luminance) of a color signal
  - **composite signal** single signal encoding the luminance and chrominance signals
  - **deadzone (threshold)** in quantization, levels below are zeroed
  - **entropy** measure of information in message (lower bound for compression)
  - **entropy encoder** lossless compression based on message-part probabilities
  - **filtering** eliminating parts of the data (eg high frequency emphasis)
  - **fractal image compression** associate rules for a close fractal with an image
  - **frame** in motion video, a single image (eg, every 1/25 or 1/30 second)

- **genlocking** synchronizing signals of external video source, computer video
- **human visual system (HVS)** considered when weighting transform coefficients to give best perceived image quality
- **hypermedia** hypertext linking with multimedia
- **interpolation (usampling)** reconstruction output from samples of input values
- **layered structure bit stream** coded bits organization (eg sequence, picture)
- **lossless (noiseless) compression** ensures original data is entirely recoverable
- **lossy (noisy) compression** original data is not completely recoverable
- **luminance** brightness (monochrome) of an image (eg Y or YUV or YIQ signal)
- **motion compensation** compress considering (partial) image shifts from motion
- **motion estimation** estimate motion of pixels or blocks (eg between frames)
- **multimedia** multiple media
- **prediction** value expected based on model, earlier data
- **prediction error** (transmit only) difference between prediction, actual value
- **progressive transmission** staged (increasing fidelity) image reconstruction
- **pyramidal encoding** uses multiple resolutions of image(s), often factors of 2
- **quantization** given (low) threshold and stepsize(s), approximate a value
- **run-length coding** replace sequences with count/token pairs
- **spatial redundancy** (compressible) repetition of patterns in 2-D image
- **subband coding** separate and code each (eg frequency) band differently

- **subsampling (decimation)** encode only a sample (eg 1/2 in X, Y directions)
- **temporal redundancy** (compressible) repetition in motion video between frames
- **transform coding** map statistically dependent pixels to independent coefficients
- **zigzag scanning** (prior to run-length) diagonalize data from (0,0)

* Multimedia Document
  - **atom** couple (name, value) in a multimedia document allowing application dependent extensions of the semantics of multimedia documents
  - **consistent medium** typed CDU
  - **implementation attributes** version and release of a multimedia document implementation
  - **link root** part of a link in a multimedia document. Defines a zone in document from which a link starts
  - **link target** list of the documents to which the link leads
  - **multimedia application** application handling multimedia documents
  - **multimedia document** complex data structure aiming at synchronizing CDUs and providing bases for various mechanisms
  - **presentation dimensional attributes** spatial and temporal units defining the manipulated unit of a CDU for presentation
  - **trigger** structure defining a zone in a document and that is active during a given period of time
  - **triggered mode** in a trigger defines the state of the active zone after the trigger is fired
  - **triggering mode** in a trigger defines how a trigger is fired
* Hypertext
  - **anchor** selection in a text such as a fluorescent highlight in a book

- **navigational linking** ability to connect anchors using persistent ties that enable them to move between references and documents
- **warm linking** navigational linking allowing the exchange of data over the links
- **hot linking** linking with automatic synchronization between anchors One anchor is the *master* and others the *instances.* When the master is modified, the instances are also modified
- **active anchor** anchors from which animations are started
- **web** collection of anchors and linking information among a set of documents.

Appendix 1

1.1 Introduction

This part presents an application as understood in chapter 4 section 2. This implies synchronization and real-time considerations and its requirements are expressed in terms of presentation to the end user. Section 2 presents the specifications of this application, section 3 presents some of the functional and performance implications. One of the requirements being to use the widely available Xwindow user interface, section 4 presents the feasibility of the application. The reader must be aware of the fact that this part does only intend to point out some issues implied by multimedia, regardless of any software development method.

1.2 Specifications

The application allows the user to see the animation of the computed highway traffic simulation. It must have the following features:

* on Xwindow, a window displays lanes and cars running on them
* milestones allow the user to know where the cars are at any time
* selected cars can be followed:
  - in a single window
  - in multiple windows
* the result must be comparable to video.

The data is gigabytes of digits. Raw files can be obtained containing the following fields: *time, position*(x), *lane #, ID, speed.* Note that NTSC video means 30 frames/second.

1.3 Functionalities and performances

The previous specifications have the following implications:

* video means access to real-time primitives so as to guarantee the display of 1 frame each $1/30^{th}$ of a second

* this high display rate implies quick access to the data and/or powerful computation. For example, if the images are stored on disk as uncompressed bitmaps, one image is $1024 \times 300 = 307200$ *bit*, that is, at the NTSC display rate, the uncompressed stream is $307200 \times 30 = 9216000$ *bit/s*
* quick access to data implies:
  - data prefiltering
  - data presorting
  - data compression
  - high access speed storage devices.
* powerful computation implies:
  - precomputation of images
  - prefiltering of data
  - dispatching of the computation
  - MIPS purchase.
* access to real-time functions:
  - with UNIX, need UNIX real-time
  - can be implemented on a PC or equivalent so that interruptions can be used to implement real-time functions
  - a dedicated architecture can be considered.

1.4 Feasibility using Xwindow X11R4

As Xwindow is part of the specifications:

* if Unix-based workstations are used, UNIX real-time needs to support Xwindow. That's not currently available, but is under development with Motif as a development toolkit
* if the application runs on a PC or equivalent without UNIX (otherwise we are in the previous case), no more processes can be run. As a consequence, another machine is needed to host the display process:
  - another PC
  - a workstation
  - an X terminal.

PC with the storage device	X terminal
- computes the Xprotocol drawing requests - time distribution	- draws on its own pixmap -displays when asked (synchronous mode)
OR	
- retrieves precomputed pixmaps - distributes time	- displays pixmaps

Figure 1.1: Possible configurations

Example: see figure 1.1

Note that even with the synchronous mode for the display and if the display rate is faster or equals the input data rate, assuming that the connection between the server and the application is steady, jitter bounds can not be guaranteed and the result can be jerky.

As presented in chapter *Xwindow for multimedia*, the Xwindow system has the following features:

* on most of the implementations, X works in asynchronous mode. The clients request are buffered by Xlib and sent under one of four conditions[11]. None of these conditions refer to absolute time.
* the `XSynchronize` request turns the Xlibrary in synchronous mode, but the performances drop. Moreover, this mode only implies that the requests are sent as soon as they are built. No guarantees are given on any of the typical parameters presented in chapter 3.
* an X server can:
  - maintain pixmaps and be orders to operate on them

[11] see section 5.4.1.

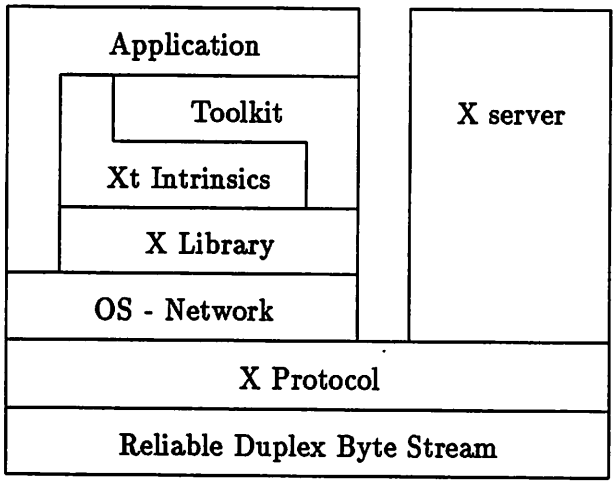


Figure 1.2: Software layers for the present application

- be ordered to display a pixmap sent over the network.

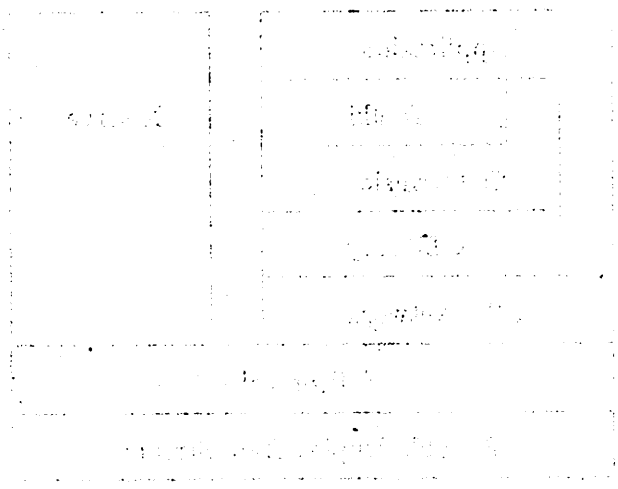
The implementation of X applications is presented in chapter 5. Figure 1.2 presents the differences that are to be considered for the present application. The application must access real-time functions and network primitives.

1.5 Conclusion

The application quickly presented here has very short specifications and should apparently easily be solved. Yet, it raises many of the issues developed in this report. Thus, the specified quality of the display result points out:

* the need to access real-time primitives
* the need to describe tight continuous synchronization
* the limitation of a current user interface.

Moreover, the data that need to be sent over the network raises performances issues. Thus, compression has to be considered and the need to specify data rate over the connection between the retrieval process and the presentation process.



Appendix 2

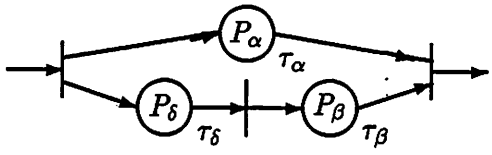


Figure 2.1: Unified OCPN model

2.1 Introduction

Object Composition Petri Nets have been developed in [LG90b]. The authors define OCPNs and show that, for any two atomic processes specified by temporal intervals, there exist an OCPN representation for their relationship in time.

A list of seven basic temporal relationship schemes between two of what we call *CDU presentation processes* is then shown to be consistent. This is used in chapter 4 to give a basic justification of the defined data structure. The development of this justification has shown that OCPNs can easily be mapped on our structure. Several mapping method are possible. We only give one of them.

2.2 Mapping method

In [LG90b], a unified model is considered (see figure 2.1). Our structure defines the simultaneity in the presentation between the constitutive units of a multimedia document. Thus, this feature is represented by the starting transition of the unified OCPN model.

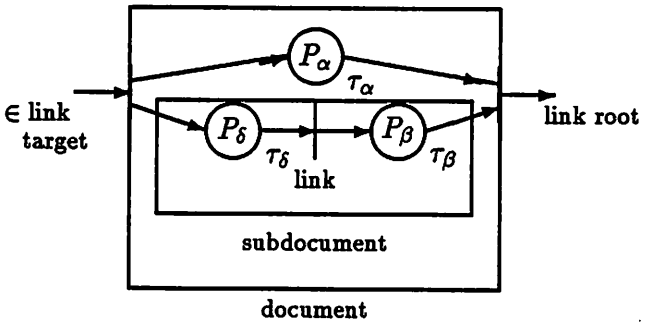


Figure 2.2: Mapping method

Between the starting and ending transitions, a multimedia document can be defined containing two constitutive units. P_α can be a CDU or a subdocument in case it is a subnet and the subdocument containing P_δ and a link to P_β embedded in a subdocument. Figure 2.2 presents that graphically.

2.3 Transition/link

Lets consider a transition in an OCPN. As stated before, this defines a multimedia document. The incoming arcs represent the target of a link whose node is in the previous document. Each outgoing arc defines a single constitutive unit of the document. Figure 2.3 give a diagrammatic view of this property.

2.4 Conclusion

This short presentation of the mapping of OCPNs on our structure can be formally studied. Yet, an intuitive approach of this has been given. A further step would be to define precise mapping algorithms.

However, the semantics of OCPNs are a reduced subset of the semantics of our structure, and using the latter only as a repre-

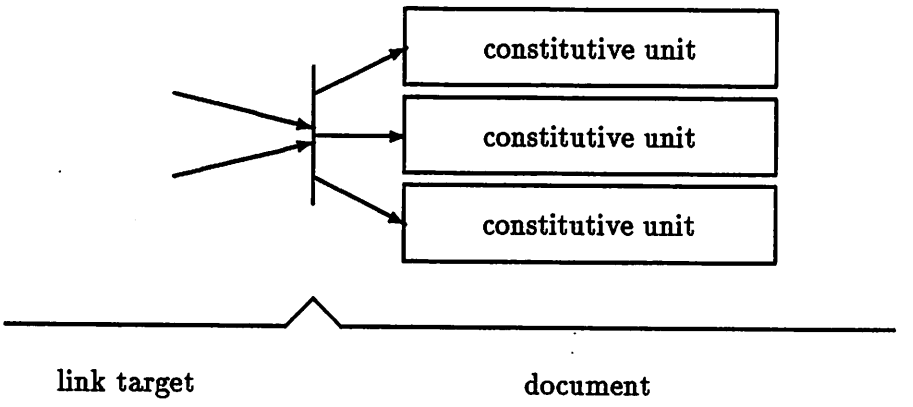


Figure 2.3: Mapping transition/document

sentation of OCPNs does not fully use the multimedia ability of the structure.