Chapter 1

Introduction

1.1. Multimedia systems

In the past few years we have seen great advances in computer technology. The current generation of computer CPUs are much faster than previous generations. Computer networks operating at 100 Megabits per second are becoming widely available, and storage devices with capacity larger than hundreds of megabytes can be found even in the simplest personal computers.

The advances in hardware technology have prompted the emergence of new software techniques. New types of digital media are increasingly common in current computer systems [Gall,91][Liebhold,91][Liou,91]. Among the new types of digital media some deserve a special mention: digital video, digital audio, formatted music encoding, and computer generated animation. These new types of digital media are known as continuous media because they continuously change over time [Anderson,91]. Continuous media incorporate an inherent temporal dimension that has to be maintained in order to preserve the integrity of the information.

The integration of continuous media with traditional computer applications has resulted in multimedia systems. The term multimedia is widely used in the computer community, but there are many different meanings associated with it. For instance, the PC industry uses the term multimedia to describe personal computers with CD-ROM and audio playback facilities (e.g.: multimedia PC). In the academic community, the term multimedia is used to describe the use of continuous media types together with traditional computer media.

Although the various definitions for multimedia differ considerably, almost everyone seems to agree that multimedia is somehow associated with the manipulation of continuous media data streams. A system that is not capable of handling at least one type of continuous media is not considered a multimedia system. This leads us to the following definition:

> ***Multimedia*** *is the ability to manipulate and possibly integrate continuous media data types (e.g.: digital video, digital audio, formatted music encoding (e.g.: MIDI), and computer animations) together with traditional*

computer media such as ASCII text, computer graphics, and real-life still images.

Very recently there have been great expectations for multimedia systems. One motivation for this is the wide range of new applications that are feasible by using multimedia resources. Also, a great number of existing applications, such as electronic mail and presentation systems, can be vastly improved by integrating multimedia with their current specifications.

Multimedia computing is in its very early stages and we can expect great changes in the way computers interact with users. Multimedia-ready workstations from Silicon Graphics and Apple are setting the standard for computer-user interaction. These workstations have built-in hardware to manipulate digital video and digital audio. It is possible to foresee a wide range of new multimedia applications that will be developed to explore the multimedia capacity of such hardware.

1.2. Distributed multimedia systems

Distributed systems are characterised by the use of computer networks to interconnect otherwise independent computer nodes [Tanenbaum,92]. The network is used to exchange information between nodes, which then cooperate to solve a specific task. This way the work is not performed by a single node. The work is divided into tasks that are then distributed to multiple nodes that will work together in order to produce the final result.

Distributed systems have several advantages over non-distributed [Tanenbaum,92]. Firstly, multiple CPUs can work together to solve a single problem. This increases the computational power by the number of CPUs available on the system. Secondly, data for the application can be scattered over different locations. Thirdly, sharing expensive peripherals such as CD-ROMs and printers is a very cost effective solution. And fourthly, a distributed system allows smooth degradation in case of failure.

Distributed systems also introduce some new problems. For instance, issues such as network reliability, communication protocols, data representation format (e.g.: big-endian vs little-endian) [Zahn,90], and overall performance have to be addressed during the design of distributed applications. These issues can potentially increase the complexity of new applications.

Despite the increase in complexity of new applications, it is generally accepted that the advantages of distributed systems outweigh the disadvantages. In the case of distributed multimedia systems, two main reasons lead us to this conclusion.

Firstly, the storage requirements for multimedia presentations are in the order of hundred of megabytes or even gigabytes [Little,90b], which is well above the storage capacity available in most workstations and personal computers. Using a distributed multimedia system the information can be stored away from the user workstation and fetched at real time during the presentation. In addition, multimedia data can be stored over different physical locations and

still be accessed transparently. This contributes to the implementation of dedicated storage servers as proposed by [Gemmel,92]. Storing data in a public storage server also allows the information to be shared by a large number of users without the need to duplicate the data.

Secondly, new multimedia applications are inherently distributed [Williams,91]. They usually involve a group of people that need to interact with each other from independant computer nodes. Multimedia applications that involve group presentation rely upon the ability to deliver the information simultaneously to a group of individuals scattered in different locations. A distributed multimedia system allows the application to select a group of individuals and then transmit the information to them independent of their locations. Applications such as remote class education and news broadcasts can benefit immensely from these facilities. Multimedia applications that involve Computer Supported Collaborative Work (CSCW) [Pehrson,92] require a great level of integration. Most of the time there is a need for a two-way flow of information between each pair of participants. Usually the participants of a CSCW session are located in different physical locations and a distributed system is necessary to allow the exchange of information. Multimedia applications such as video conferences and joint authoring tools fall into this category [Knister,90][Watabe,90].

In order to fully exploit the potential of new multimedia applications, a distributed multimedia system is essential. It provides a very efficient mechanism to facilitate the interaction of people in different locations. It also promotes a better utilisation of data and peripherals by sharing them among multiple users.

1.3. Media synchronisation

Continuous media data streams need constant attention from the CPU during their presentation [Anderson,91][Campbell,92]. They include a constant time interval between the playback of successive frames that must be strictly maintained by the CPU. It is important that this interval be kept constant so that the information is represented correctly. The characteristics of the application may also impose similar requirements on traditional data streams. For example, in an audio visual presentation it may be required to display a sequence of pictures with a pre-defined time interval between each picture, so that the user has enough time to analyse and understand each picture. This temporal relationship between successive frames requires an accurate synchronisation during the playback, so that this important characteristic of the media stream or presentation is correctly maintained.

Multimedia presentations are usually composed of multiple physically independent data streams. The ordering of the presentation is controlled by establishing a temporal relationship between the various data streams [Horn,93][Little,90b][Little,91a][Rowe,92]. The temporal relationship specifies the order in which events should happen. For example, consider a multimedia audio visual presentation comprised of still pictures and a sound track. For each picture displayed there is a corresponding audio track. The temporal relationship must ensure

that the corresponding audio track is always in synchrony with the image being displayed. The correctness of the presentation depends upon the capacity to maintain the time ordering of events as specified by the temporal relationship.

On current computer systems the synchrony of a presentation can easily be lost. Considerations such as the processing speed of the CPU, the number of processes running on the system, the load on the network sub-system, and the maximum data retrieval capacity can greatly influence the processing speed of a given media stream. This results in variations in processing delays and consequently interferes with the synchrony of the presentation.

The temporal dimension associated with multimedia data streams and multimedia applications creates the requirement for media synchronisation. A media synchronisation environment is necessary to accurately maintain the ordering and timing of events during multimedia presentations. The synchronisation environment must be able to quickly detect and recover from any loss of synchrony resulting from variations in delays.

1.4. Research objectives

As described above, media synchronisation requires a synchronisation environment to correctly maintain the temporal relations during multimedia presentations. The aim of this thesis is to provide a framework in which such an environment can be built.

This work concentrates on the design of a Distributed Synchronisation Architecture (DSA) that implements media synchronisation transparently to multimedia applications [Bastian,94]. The design attempts to meet two major goals:

- To provide a synchronisation formalism to specify the temporal relations required in multimedia applications. The formalism must hide the details about the different characteristics of each media stream.
- To provide a platform that supports the described synchronisation formalism and transparently provides multimedia applications with the correct synchronisation. The platform must provide multimedia applications with a set of controlling functions so that applications can accurately control the behaviour of presentations.

1.5. Organisation of the thesis

The rest of this thesis is organised as follows:

Chapter 2 attempts to provide a comprehensive list of characteristics and requirements of continuous media streams.

Chapter 3 defines media synchronisation, the problems facing media synchronisation on distributed systems, and the techniques available to correctly maintain synchronisation.

Chapter 4 presents the state of the art in media synchronisation on distributed systems. The research is grouped into three sections: synchronisation at the operating system level,

synchronisation at the network communication level, and synchronisation at the presentation level.

Chapter 5 presents a distributed framework for media synchronisation support. The distributed framework provides a synchronisation formalism and a distributed platform that implements media synchronisation transparently to multimedia applications.

Chapter 6 describes the application interface and an experimental implementation of the distributed platform.

Chapter 7 draws some general conclusions and highlights some limitations wherein further research is necessary.

Chapter 2

Continuous Media Characteristics and Requirements

2.1. Introduction

Continuous media streams are the core base for multimedia systems. They add some new characteristics to those of traditional data streams. Unfortunately, computer systems and their sub-systems (e.g.: computer networks) are not well designed to support these new characteristics [Sammartino,91]. New mechanisms and techniques are needed to support continuous media streams on distributed environments. This chapter attempts to describe the characteristics of continuous media and the requirements for the current generation of computer systems.

2.2. Characteristics of continuous media

2.2.1. Temporal dimension

Continuous media streams are achieved by continuously "freezing" or sampling the incoming signal at pre-defined fixed intervals, and storing the result into digital form. For example, digital audio is recorded by repetitively sampling the analog signal and recording the wave's amplitude at the time of the sampling [Luther,91]. The sequence of digital values can be used to accurately reconstruct the initial wave. Digital video is recorded in a similar way. At fixed time intervals the image is "frozen"[1] and the colour of each pixel in a two dimensional matrix of pixels is stored in digital form. This technique enables the reconstitution of each frame and consequently, of the video stream.

Every continuous media stream is composed of a sequence of samples or frames, which convey meaning only when presented continuously in time. Each frame contains different information that has to be presented at the correct time. Speeding up or slowing down the presentation of successive frames will result in different information than originally recorded.

The continuity of both presentation and recording, associated with the time interval required between samples, creates the temporal dimension of continuous media. This is the characteristic that clearly distinguish continuous media from traditional media streams.

---

[1] In reality, most digital video streams are generated by continuously converting an analog signal from an existing analog video format (eg: NTSC or PAL) into digital values.

2.2.2. Tolerance to errors

Continuous media streams have various degrees of tolerance to errors [Anderson,91][Campbell,92]. Unlike traditional media, where even the smallest error in the data stream could be disastrous, continuous media streams can successfully deal with some degree of data error.

Because of the temporal characteristic of continuous media streams, the data presented in one frame is quickly overwritten by the subsequent frame. Therefore, frame n+1 overwrites frame n, correcting any errors introduced during the presentation of frame n. This allows continuous media to successfully tolerate data errors.

There are two types of data error. Firstly, because of distortion into computer peripherals, the original electrical or optical signal may deteriorate and a misinterpretation of the binary stream will result. This type of error generates a change in the quality of the presentation and the user may or may not perceive the error. Secondly, information may be lost. When part of a frame or the whole frame is lost, the missing information can be substituted for "dummy" data, just to maintain the correct temporal relationship. The presentation is corrected as soon as the next frame is played/displayed.

When the occurrence of such errors is kept to very small levels, the temporal characteristic of continuous media can successfully recover the original meaning of the information. Even though both types of data error incur a loss of quality, the information can still be successfully manipulated and presented to the user.

2.2.3. Storage demands for continuous media

The storage requirements for continuous media greatly exceed the requirements for traditional media [Little,90b]. The amount of storage required by a continuous stream is a factor of the quality of the presentation and its duration.

Digital audio quality can be measured by the amplitude of the analog wave and its bandwidth. The amplitude of the analog signal dictates the sample size and the bandwidth dictates the sampling rate required to properly digitise the original signal. Speech is represented by a 4 khz bandwidth and requires 8000 samples per second, 8 bits per sample to be properly digitised [Tanenbaum,88]. CD quality audio has a 22 khz bandwidth and requires 44000 samples per second and 16 bits per sample.

Digital video quality depends on the size of the image, the frame rate, and the colour accuracy required. NTSC [Luther,91] format video uses 640x480 pixels per frame and a frame rate of 30 frames per second. PAL [Luther,91] format video uses 768x576 pixels per frame and a frame rate of 25 frames per second [Correl,92].

The table below summarises the storage requirements for these types of continuous media.

CM type	Details	1 second	1 minute	1 hour
Speech	8 bits/sample 8000 samples/sec	64 kbps	480 Kbytes	28.8 Mbytes
CD audio	16 bits/sample 44000 samples/sec	704 kbps	5.2 Mbytes	316 Mbytes
NTSC format	640x480 pixels/frame 24 bits/pixel for colour 30 frames/sec	221.1 Mbps	1.6 Gbytes	99.5 Gbytes
PAL format	768x576 pixels/frame 24 bits/pixel for colour 25 frames/sec	265.4 Mbps	1.9 Gbytes	119.4 Gbytes

Figure 2.1. Storage requirements for continuous media

When continuous media have to be delivered in real-time, the storage requirement shown above for one second also represents the network and retrieval throughput required for the continuous media stream.

Clearly, the storage requirements of continuous media can saturate most workstations and personal computers. Also, real-time transmission of continuous media can pose a real problem even for fast networks such as Fiber Distributed Data Interface (FDDI) [Tanenbaum,88].

2.2.4. Time requirements for continuous media

The temporal dimension of continuous media imposes timing requirements when recording or presenting the information. These timings must be strictly controlled to avoid lost of fidelity. The following table highlights the time granularity for the continuous media types described in the previous section.

CM type	Rate	Time granularity
Speech	8000 samples/sec	125 microseconds
CD audio	44000 samples/sec	22.72 microseconds
NTSC format	30 frames/sec	33.3 milliseconds
PAL format	25 frames/sec	40 milliseconds

Figure 2.2. Timing requirements

2.3. Continuous media requirements

In the past two decades the trend was to share computer resources. Great effort was put in to develop techniques that would allow multiple applications to share scarce computer resources. The idea was to improve the total processing throughput while decreasing the throughput for a single application.

Sophisticated scheduling mechanisms were developed so that multiple processes could share the same CPU [Bach,86][Leffler,89]. Data multiplexing and packet switching mechanisms allowed computer networks to support multiple connections simultaneously [Tanenbaum,88]. Buffering and retrieval algorithms were fine-tuned for high performance when storage devices were being accessed concurrently [Leffler,89].

The above techniques have been successful with traditional media streams because delays did not cause applications to produce the wrong information. However, any small delays when manipulating continuous media can render the information useless. The temporal dimension of continuous media requires a constant degree of performance and any variations result in a loss of quality.

To properly support continuous media, computer systems have to exploit the particular characteristics of continuous media streams. These characteristics impose new constraints that must be taken into consideration when designing multimedia systems.

Continuous media requirements can be grouped into four sub-categories: storage requirements [Gemmel,92][Little,90a], which comprises storing and retrieving continuous media files; communication requirements [Anderson,90][Ferrari,93]; operating systems requirements [Jeffay,92][Witana,93]; and general requirements. These categories are described in the following sections.

2.3.1. Storage requirements

2.3.1.1. Storing continuous media files continuously

Continuous media data are usually stored in rewritable storage devices. Because of their large size, continuous media files are generally stored in non-contiguous blocks of information scattered on the physical device. When the data is retrieved, the file system is responsible for providing transparent sequential access to the data. Nevertheless, at a physical level, the data has to be retrieved from non-continuous spaces, which increases the retrieval time.

Another consequence of larger files is that some file systems incur performance penalties for files larger than a threshold. The performance of the Unix file system in particular decreases as file size increases[Leffer,89]. As the size of the file grows, the file system uses multiple indirections to locate the data. This may cause a performance problem for continuous media files because files larger than a few tens of megabytes are very common to store digital audio or digital video.

Continuous media files do not change often, therefore storing them contiguously on the file system may considerably improve the performance of the retrieval mechanism as well as eliminate the multiple indirection problem. Gemmel [Gemmel,92] has shown that for continuous media files, contiguous allocation achieves better performance than non-contiguous allocation mechanisms.

2.3.1.2. File system buffering

File systems tend to have a buffering mechanism to improve performance when reading and/or writing into a file [Bach,86][Silberschatz,88]. The buffering mechanism is composed of two parts: a buffer cache and a read-ahead algorithm.

The buffer cache mechanism maintains frequently accessed data in the system buffers, so that accessing this data is faster than accessing the physical device. The read-ahead algorithm assumes that the application will soon need the data which is just ahead of the information being currently accessed. Therefore, it reads a few more blocks than the initial amount requested by the application and keeps these blocks in the buffer cache.

The buffer cache is not necessary for continuous media streams. During a multimedia presentation, data that has already been displayed is rarely needed again. Therefore, buffers are better utilised if they are returned to the free pool immediately after their use, so that they can be reused.

Standard cache systems are global per storage device and not per process. Each buffer mirrors a specific block of the storage device that was accessed recently. When all buffers are allocated and more data has to be accessed, the buffer which has the oldest time is chosen to contain the new data. This technique works fine for non-continuous media streams. In case of continuous media streams, the large amount of data that has to be brought into the system buffers may quickly overwrite all data in the buffer cache, resulting in poor performance for the cache mechanism.

Continuous media files can greatly benefit from dedicated file systems that are fine tuned to account for their characteristics. Gemmel [Gemmel] has proposed buffering schemes to support these type of file systems.

2.3.2. Communication requirements

2.3.2.1. Transport protocols

The transport layer is the fourth layer in the ISO Open System Interconnection (OSI) 7 layers specification [Tanenbaum,88][La Porta,91]. The transport layer is the first end-to-end layer and it provides important services for the layers above. This includes error free data delivery, usually through retransmission and checksum, flow control, and management of end-to-end connections between hosts.

Current transport protocols (e.g.: TCP [Postel,81] and TP4 [Tanenbaum,88]) were designed to effectively support error free data delivery. Sophisticated mechanisms for checksum and retransmission are incorporated in these protocols. The use of checksums guarantees that on arrival the packet is free from errors. The retransmission mechanism guarantees that packets which are lost due to network congestion or error are retransmitted by the originator of the packet.

It has been shown that computing the checksum on every packet consumes a considerable amount of time from the total packet processing time [Doeringer,90][La Porta,91]. Also, the retransmission mechanism needs to allow enough time for a packet to be acknowledged before retransmitting the information. This is necessary to avoid duplicating packets and consequently causing network congestion.

Continuous media streams can tolerate a small degree of bit error. Current generation computer networks provide very small error rates during normal operation. For an FDDI network, the maximum tolerable error rate is 1 error in 2.5×10^{10} bits [Tanembaum,88]. This rate effectively withdraws the need for checksums when transmitting continuous media streams. The CPU time saved by not computing the checksum can be an important factor when transmitting continuous media streams.

Error recovery by retransmission is another technique which is not well suited to the transmission of continuous media streams [Doeringer,90]. The temporal relationship of continuous media streams often requires very short times (e.g.: 33 millisec or less) between the playback of consecutive packets. A continuous media packet is useless if it is not present at the precise time it is needed. To guarantee that a packet is present at the required time, the retransmission mechanism is limited by the time interval between packets. Therefore, retransmission has to take place within the maximum time interval between packets. These time intervals are so small that they can not be used to trigger retransmission without causing duplication of packets. This limitation renders retransmission mechanisms useless for continuous media streams.

2.3.2.2. Performance guarantee

The temporal characteristic of continuous media streams requires a constant demand of network resources once a connection has been established [Ferrari,92][Fry,93]. Parameters such as the required throughput, maximum end-to-end latency, and maximum delay jitter are constant during the connection lifetime.

The use of packet switching technology does not automatically reserve the necessary resources required for continuous media streams when the connection is established. Network resources are free to be used simultaneously by multiple virtual circuits. The result is that a sudden surge in traffic may change completely the performance of any virtual circuit operating on the network. This may cause unacceptable delays when transmitting continuous media streams.

To guarantee the transmission performance for continuous media streams, virtual circuits that can statically reserve the required performance for every connection are necessary.

2.3.3. Operating systems requirements

2.3.3.1. Scheduling mechanism

Conventional multitasking operating systems are not well suited to handle the real-time characteristics of continuous media streams [Bulterman,91][Jeffay,92]. These systems typically use a round-robin scheduling mechanism which associates a time slicing to each runable process in the same priority level [Bach,86][Leffler,89]. However, processes handling continuous media streams need a more deterministic scheduling mechanism so that they can meet the deadlines imposed by continuous media streams.

The performance of the scheduling mechanism for conventional operating systems depends on how scheduling priorities are allocated. The Unix scheduler uses a mechanism that tries to be fair with all processes running on the system. It systematically adjusts the priority of processes according to the CPU usage of each process. The scheduler decreases the scheduling priority of CPU intensive processes and increases the priority of I/O bounded processes. This dynamic change of process priorities does not favour continuous media processes because they need constant attention from the CPU. On a medium to large CPU load, continuous media processes are likely to miss deadlines as the scheduler reduces their priorities to allow other processes to run.

In order to properly support continuous media processes, operating systems need to provide scheduling mechanisms with fixed, high-priority scheduling levels that can be statically associated to continuous media processes.

2.3.3.2. Preemptible kernel

At the kernel level, the Unix operating system is non-preemptible [Leffler,89][Bulterman,91]. That means that if a process is executing a system call and another process with a higher priority becomes ready to run, the process with the higher priority will not receive the CPU until the system call for the first process is completed. Depending on the system call being executed, this delay may vary from a few microseconds to up to a few hundred milliseconds.

For continuous media processes, the wait for lower priority processes to finish may compromise their correct execution [Mercer,92]. Therefore, operating systems that can support immediate scheduling for higher priority processes are required to properly support continuous media processes.

2.3.3.3. Data transfer

The read and write operations to transfer data from and to devices are widely used in operating systems [Bulterman,91[Campbell,92]. Every time an I/O operation has to be performed, one of these two system calls is executed. These operations are rather costly, because they require switching back and forth to the operating system kernel. Furthermore,

there is generally a need to transfer data from the process address space to kernel buffers during a write operation and vice-versa during a read operation.

These conventional interfaces are potentially inefficient at handling continuous media streams. It is inherent to continuous media processes that I/O is necessary at continuous rates [Campbell,92][Govindan,91]. There is no need to reissue I/O operations during manipulation of continuous media streams. Instead, the data could be made automatically available to applications through the use of shared circular buffers [Govindan,91]. If the location of the information is already known, data can be transferred by re-mapping page tables so that the operating system buffers are directly mapped into the process address space.

2.3.4 General requirements

2.3.4.1. Data compression

The large amount of data generated by continuous media streams poses another problem for computer systems. The throughput required for playback of a full-screen digital video (e.g.: 221 Mbps) exceeds the capacity of fast networks such as FDDI (e.g.: 100 Mbps). Also, the great majority of workstations and personal computers are unlikely to have the hundreds of megabytes required to store digital movies and the corresponding sound tracks. A full-screen, one hour digital movie with stereo sound track requires a storage space of 100 gigabytes.

Compression techniques have been used successfully to compress traditional media streams with compression ratios generally in the order of 2:1 or 3:1. However, in order for video compression to be effective, compression ratios of at least 10:1 are necessary. The solution for this problem is to use dedicated compression mechanisms that take into consideration the characteristics of each continuous media type to provide very high compression rates.

Two standard compression techniques have being developed for digital video: the Motion Pictures Expert Groups specification (MPEG) [Gall,91] and the CCITT H.261 [Liou,91] specification. Both compression techniques suppress spatial and temporal redundancies on a sequence of video frames, achieving very high compression rates. Compressing the spatial redundancies is known as intra-frame compression, while compressing the temporal redundancies is called inter-frame compression. These two compression techniques provide compression ratios of 20:1 to 200:1 [Jurgen,92].

Both video compression standards are lossy in that the reconstructed data are not identical to the original. They exploit aspects of the human visual system and the resulting video stream is hard to distinguish from the original.

The disadvantage is that these compression techniques are computationally expensive. Compression of video streams can only be performed off-line or by dedicated compression hardware. Decompression is less expensive than compression because it does not require the extensive match and search routines used during compression. However, it is still a lengthy process and can only be performed by fast processing units.

2.4. Quality of service reservation

The notion of guaranteed performance required by continuous media streams plays a prominent role in distributed systems. Continuous media streams require a constant Quality of Service (QoS) from all computer sub-systems [Campbell,92][Ferrari,93]. File systems and computer networks have to be able to maintain the required throughput, while operating systems need to support the real-time deadlines of continuous media streams.

Because most computer sub-systems are shared by multiple applications, it is not possible to guarantee the same level of performance for the lifetime of the presentation. It is generally agreed that some sort of resource manager that can provide a guaranteed QoS to continuous media is necessary.

Resource reservation mechanisms would allow a single client to specify the QoS it needs for a correct execution and the negotiated QoS will be honoured for the lifetime of the client. If for any reason the QoS negotiated during the start up of the application cannot be maintained, the client would be notified and another QoS could be negotiated. The resource reservation mechanism guarantees that a correct playback is always possible except in extreme error conditions such as a hardware fault.

Chapter 3

Media Synchronisation

3.1. Introduction

Media synchronisation is automatically associated with multimedia presentations due to the inherent temporal relation of continuous media streams. In addition, temporal relationships between independent media streams are often required when multiple media streams are used in a presentation. This chapter presents the definitions for media synchronisation and the problems associated with maintaining the correct synchronisation on distributed multimedia systems.

3.2. Media synchronisation definitions

3.2.1. Synchronisation

Multimedia presentations often contain multiple media streams that have a real-time temporal relationship [Little,90b][Little,91a][Steinmetz,90]. The temporal relationship describes the order in which events must happen and tie together previously independent media streams. For example, consider a movie presentation composed of a video stream and an independently stored audio track. When there is no formal relationship, the two streams can be seen as completely independent media. However, when presenting the movie, an inherent, constant data consumption rate is attached to both streams so that the correct relationship is maintained.

In the field of multimedia, synchronisation refers to the ordering and relationship of otherwise independent events. Synchronisation ensures that events occur in a pre-defined order, independent of the content and of the playback speed of the media streams [Steinmetz,90].

3.2.2. Types of synchronisation

In this thesis, media synchronisation is classified according to the number of media streams involved in the synchronisation. Synchronisation that does not depend on external media streams is classified as *intra-stream synchronisation*, while synchronisation that depends on other media streams is known as *inter-stream synchronisation*.

3.2.2.1. Intra-stream synchronisation

Whenever there is a temporal relationship between different units inside the same media stream, we say that there is an intra-stream synchronisation requirement [Bulterman,91][Little,90b].

Intra-stream synchronisation may be inherent to the media stream or explicitly expressed by an application. Continuous media streams have an inherent intra-stream synchronisation that has to be strictly maintained during playback. Traditional media streams may have intra-stream synchronisation set up by multimedia applications. For example, it is possible to have a media stream containing a sequence of text annotations which have to be presented with a pre-defined time interval between them. This creates a temporal requirement just like ordinary continuous media streams. The difference is that the temporal requirement is not implicitly associated with the media stream, but it is imposed by the multimedia application.

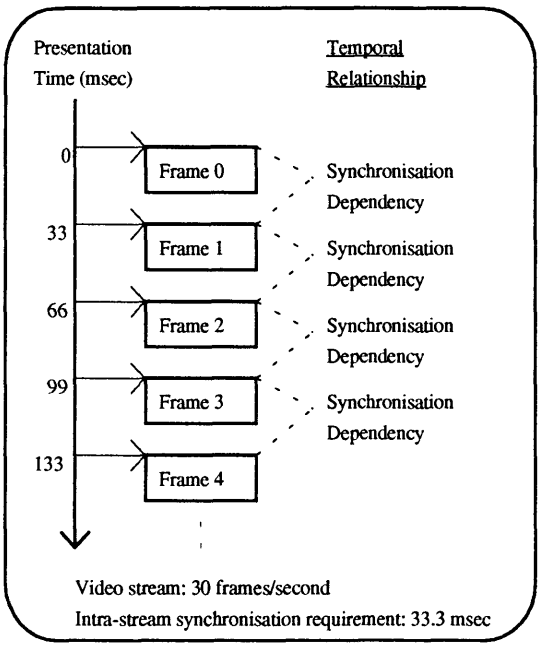


Figure 3.1. Intra-stream synchronisation

The details about the intra-stream synchronisation have to be known prior to playback. There are several ways to inform the device that will perform the synchronisation of the intra-stream synchronisation requirements:

- **File type or input device** - The intra-stream synchronisation characteristics for some continuous media streams can be inferred for stored streams by associating certain characteristics with a filename extension. For example, *.PCM files can be associated with audio files with 8khz sampling rate and 8 bits per sample. Some input devices may also have an association describing the characteristics of the continuous media stream generated by the device.

- **File header** - On stored streams, the intra-stream synchronisation details may be stored in a header located at the beginning of the data stream. For example, an MPEG video stream includes a header which contains some information about the characteristics of the movie (i.e. frame size, frame rate, etc.).
- **Information in each data unit** - When the intra-stream synchronisation is not constant for the duration of the playback, the synchronisation information may be described in a data unit basis. Each data unit contains its own intra-stream synchronisation information which describes the temporal relationship with previous or future data units.
- **Information in an external file** - The intra-stream requirements are stored in an external file. The information contained in the external file has to identify and describe the synchronisation characteristics for every data unit. This form of synchronisation location is the most flexible because the format and type of information stored in the external file is not constrained by the actual format of the data streams. However, the inclusion of an external file means that this file must be accessed in order to retrieve the synchronisation information which in turn may considerably increase the overhead on the system.

3.2.2.2. Inter-stream synchronisation

Inter-stream synchronisation exists when there is a formal relationship between units in different media streams [Bulterman,91][Little,90b]. Inter-stream synchronisation must be explicitly formulated by establishing the temporal relationship among the streams to be synchronised. For example, consider a multimedia slide presentation in which the presentation of each image is followed by a verbal annotation. The first image is displayed at time 0 and subsequent images are displayed after the verbal annotation for the previous image completes. Each image and the end of each verbal annotation represents a synchronisation point that must be formally described so that synchronisation is maintained throughout the presentation.

Another common form of inter-stream synchronisation is the synchronisation of video and audio, particularly lip-synchronisation [Nicolau,90]. Lip-synchronisation involves the synchronisation of the lips' movement on the screen with the spoken voice from the speaker.

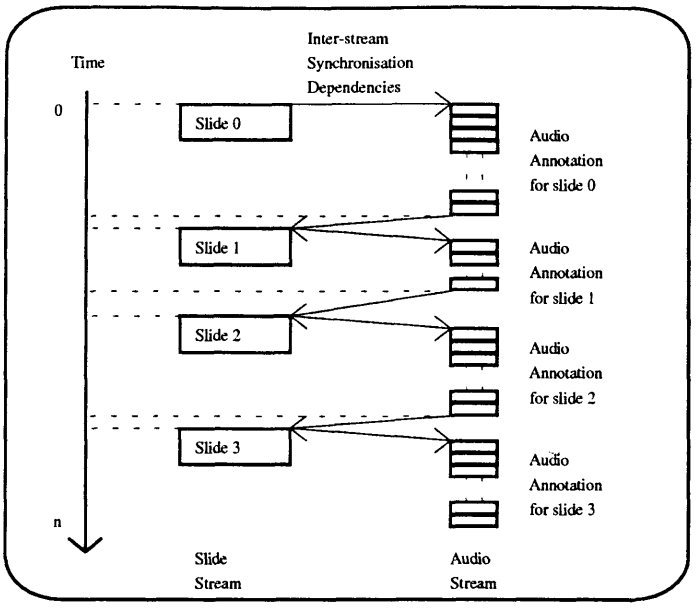


Figure 3.2. Inter-stream synchronisation

As with intra-stream synchronisation, there are also several ways to specify the inter-stream synchronisation requirements:

- **Cross references** - In this scheme, each data unit that has inter-media synchronisation requirements contains references to other data units in different streams. This approach is very versatile but it increases the complexity for specifying and maintaining synchronisation points.
- **Master stream** - A single media stream contains all synchronisation information required for synchronised playback. The master stream must be available during the lifetime of the presentation because all synchronisation information is contained within it.
- **Information in an external file** - The inter-stream synchronisation is stored in an external file. The external file contains references to the data units in the streams and logically groups them according to the synchronisation requirements for the presentation. This approach involves having the extra overhead and resources to process the external file.

3.2.3. End-user synchronisation

End-user synchronisation refers to synchronisation as seen by the audience of the presentation. It is not important if the system is internally altering the presentation as long as to the user's perception the presentation is synchronised. This is particularly important on distributed systems, because distributed systems involve three distinct processing steps: retrieval, transmission, and presentation. It is not necessary to strictly control synchronisation when retrieving or transmitting the information as long as the information is synchronised before it is presented to the user.

3.3. Sources of delays

At first it seems that after the temporal relationship for intra- and inter-stream synchronisation has been specified, maintaining the correct synchronisation is just a matter of outputting the information at the correct time. However, allocation of computer resources (i.e. CPU, network, disk, etc.) is non-deterministic in the sense that it is not possible to predict when a given resource will be available to the application [Bulterman,91]. This characteristic of current computer systems makes the task of synchronising media streams more difficult than initially visualised.

Due to the temporal relationship of multimedia presentations, any small variations in the processing speed of the media streams may result in delays in maintaining the temporal links. Such delays interfere with the correctness of the presentation and represent a decline in the overall quality of the presentation.

In a distributed environment, the sources of delays can be organised in three distinct groups: *retrieval delays* [Gemmel,92], *transmission delays* [Escobar,92], and *presentation delays* [Anderson,91][Rowe,92].

3.3.1. Retrieval delays

Retrieval delays account for time variations during the retrieval of media streams from storage devices or input devices (i.e. video camera, microphone, etc.). It represents the time elapsed since the application requested the data to the moment that the application can use the information.

In multiprogramming environments, storage devices may have multiple requests for data retrieval being executed almost simultaneously by different applications. In this case, the maximum device transfer rate will be shared by all the applications requesting data from the device. The result is that there is no guarantee that the required transfer rate will be available to retrieve the media stream, even if the device transfer rate is greater than the maximum transfer rate required by the media stream.

In addition, there are the issues of the bus transfer rate and how fast a process waiting for I/O will execute after the I/O operation has completed. Depending on the load on the system and the number of processes sharing the CPU, the timing for these issues can vary widely.

3.3.2. Transmission delays

Transmission delays account for every delay introduced after the data is delivered to the network sub-system and before the data is available at the destination application.

On distributed systems, the information is often stored in different physical locations and has to be transmitted to the destination machine before it can be presented. This step involves multiple processing layers with different time overheads. Usually, the time taken for transmitting a packet includes the operating system buffering time, the protocol processing

time, and the actual transmission time. These times vary according to the implementation, the length of the packet to be transmitted, and the load imposed on the system.

3.3.3. Presentation delays

Presentation delays account for the time elapsed from the moment the data is available to the destination application until the information is actually presented to the end-user. Presentation delays includes delays wich arise due to the multiprogramming environment and the scheduling policy used by the operating system.

In multiprogramming environments, all processes are competing for the CPU and the degree of CPU attention required for each process varies considerably. The Unix scheduler uses a mechanism that tries to be fair to all processes, decreasing the priority of a process as its CPU usage increases. As processes dealing with continuous media streams need constant CPU attention, this policy does not favour them and they are likely to miss deadlines.

3.4. Techniques for maintaining synchronisation

During the course of a multimedia presentation, any small delay can interfere with the correctness of the temporal relations, resulting in an incorrect or "poor quality" presentation. Also, depending on the degree of delay being introduced, the presentation of continuous media streams may turn out to be useless because the user can not understand the information that is being presented. It is important that multimedia systems detect and deal with delays in order to maintain the correct temporal relationship for the presentation.

The techniques available to smooth delays and maintain the correct synchronisation are: buffering media streams, discarding frames, changing the QoS, and pausing the presentation. The rest of this section describes these techniques.

3.4.1. Buffering media streams

The effect of delays can be reduced by using buffers to pre-fetch the information before starting the presentation [Little,92][Ramanathan,93]. Buffering the information in the destination machine can effectively smooth retrieval and transmission delays. However, buffering is limited by the number of buffers available and by the time taken to fill up the buffers. Ramanathan [Ramanathan,93] has shown that the buffering requirements for continuous media streams are many times above what a normal computer system can support. Also, the time that it takes to fill up the buffers at the start of a presentation adds up to the time that the user has to wait before the presentation can be commenced.

Nonetheless, buffering can successfully smooth small variations in delays during a multimedia presentation.

3.4.2. Discarding frames

The technique of discarding frames is particularly useful when dealing with continuous media streams [Anderson,91][Rowe,92]. It explores the tolerance to errors of continuous media streams by discarding frames or samples which are known to be late for their playback time. Discarding a frame speeds up the playback of the media stream in relation to the real-time clock, effectively eliminating delays.

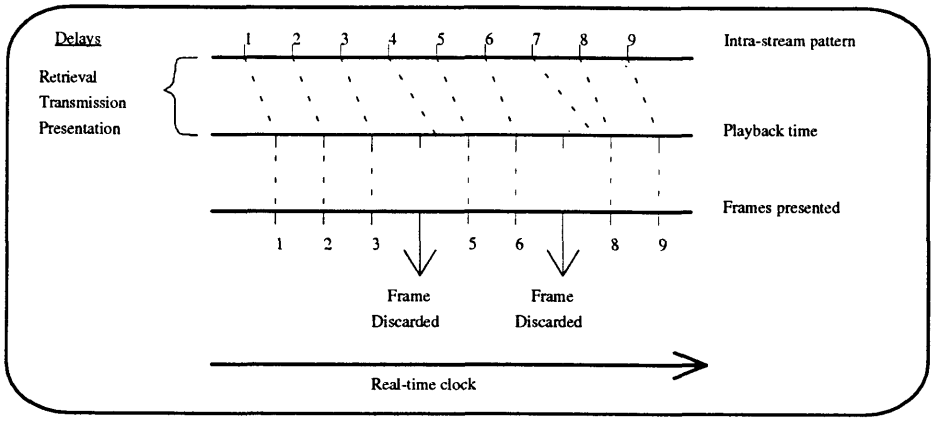


Figure 3.3. Discarding late frames

3.4.3. Changing the QoS

Changing the quality of the presentation to a lower quality reduces the amount of information that has to be manipulated and results in better performance for the overall presentation [Campbell,92][Rowe,92]. It effectively improves the synchronisation as long as the overhead on the system is being caused by the presentation. If the system is overloaded with information generated by other applications, this technique does not produce much better results.

Reducing the quality of the presentation can be accomplished by using media streams that contain the same information although with a lower overall quality. For example, a black and white video stream that contains fewer frames per second compared to a colour video stream with 30 frames per second.

3.4.4. Pausing the presentation

Pausing the presentation is the last resource to maintain synchronisation. If all the previously described techniques fail to maintain the synchronisation, the presentation needs to be paused until it can be resumed.

Chapter 4

The State of the Art in Media Synchronisation Support

4.1. Introduction

This chapter presents ongoing or recent research efforts which are related to multimedia synchronisation. Most of these research efforts address particular characteristics of multimedia synchronisation and are not necessarily aimed at end-user synchronisation.

Multimedia synchronisation requires proper support from the underlying software and hardware and some particularly interesting research projects that address these issues are included in this chapter. Although they are not directly related to multimedia synchronisation they provide some insight into techniques to support media synchronisation.

This chapter has been divided into three sub-sections:

- Synchronisation at the operating system and storage device level
- Synchronisation at the network communication level
- Synchronisation at the presentation level

The chapter concludes with a summary of the research and points out some areas where more research is necessary. Some of these areas are directly addressed in the work described in this thesis.

4.2. Synchronisation at the operating system and storage device level

4.2.1. University of North Carolina

At UNC, research has been undertaken in real-time requirements for multimedia ready operating systems [Jeffay,92]. Rather than adding real-time scheduling to an existing operating system, they have considered a new operating system specifically biased to provide real-time services for multimedia. The YARTOS (Yet Another Real-Time Operating System) was designed so that application programmers can specify both real-time throughput and latency requirements for individual processes.

YARTOS supports two basic abstractions: tasks and resources. Tasks are units of execution and they only execute in response to the arrival of events. When a task registers its interest in an event, it has to specify the deadlines for invocation and completion of each

processed event. Resources provide applications that do not need real-time deadlines with a non-time constrained communication facility based on shared memory. For a given workload, YARTOS guarantees that all tasks are completed before their deadlines. Using a resource reservation mechanism, new tasks can know in advance if their deadlines can be successfully met.

YARTOS provides the basic support for resource reservation and performance guarantee required for multimedia synchronisation. However, the use of a non-standard operating system makes this approach somewhat less attractive.

4.2.2. Simon Fraser University, Canada

This work [Gemmel,92] establishes a theoretical framework for the retrieval and storage of delay-sensitive multimedia data. It concentrates on the storage and retrieval of digital audio, but the authors emphasise that the results apply equally to any similar delay-sensitive data. Gemmel and Chrstodulakis have developed a set of theorems which account for the requirements of continuous media and analyse the impact in the way continuous media data is stored in storage devices.

Buffering and start time requirements for playback systems with dedicated and non-dedicated storage devices are represented by functions that depend on the device characteristics (e.g.: sector size, reading speed, number of samples, granularity of the sample). Storage performance for various storage strategies including interleaved and non-interleaved techniques are also analysed. The main conclusion of the work is that storage techniques that are biased to handle the characteristics of continuous media are necessary to properly maintain the required input/output flow when manipulating this type of information.

This work provides a good framework for implementing dedicated data retrieval mechanisms that can properly deal with continuous media. Intra-stream synchronisation during retrieval can be guaranteed using the theorems provided, however a higher level synchronisation mechanism is needed to provide the end-user synchronisation required by multimedia applications.

4.2.3. Syracuse University

At Syracuse University a model for storing and retrieving multimedia objects with temporal dimensions is being developed. The model attempts to provide documents structured using an enhanced version of the Petri Nets formalism with an infra-structure to allow synchronisation and composition of multimedia data [Little,90a][Little,90b].

Object Composition Petri Net (OCPN) includes a new dimension which allows time to be modelled. The database scheme preserves the semantics of the OCPN model to facilitate reproduction and storage of multimedia synchronisation information. The hierarchical database model contains three types of nodes: *terminal nodes*, *non-terminal nodes*, and *meta nodes*. Terminal nodes maintain the pointers to the actual data. They do not contain any temporal

information and may have many parents. Non-terminal nodes contain a left and right child pointer and the temporal information between the children. Meta nodes are like non-terminal nodes but they are allowed to have many pointers to other nodes. Meta nodes associate many temporal intervals with a single temporal relation.

The proposed database model allows quick access to the information. It is particularly important for interactive multimedia applications where non-sequential access to the information is necessary in response to user interaction.

4.3. Synchronisation at the network communication level

4.3.1. Purdue University

A set of protocols for providing synchronisation was proposed at Purdue University [Little,91b]. The protocols are based on two levels of synchronisation support. The Network Synchronisation Protocol (NSP) provides support for intra-stream synchronisation while the Application Synchronisation Protocol (ASP) provides support for application defined synchronisation which may involve inter-stream synchronisation.

The NSP deals with single end-to-end connections and assumes that the network sub-system can guarantee the required level of performance defined during connection establishment. At the time of connection establishment, the NSP provides the communication requirements (e.g.: traffic, packet size, etc) and expects the network sub-system to return the channel delay and the variations in the interarrival time for consecutive packets.

The ASP works by retrieving OCPNs (see Section 4.2.3) from a database and generating scheduling deadlines and calls for the NSP layer.

This approach is specially designed to interface with the formal synchronisation specification provided by an OCPN. The main limitation of this work is that because of the static nature associated with the OCPN database, it does not account for synchronisation when user interaction is required.

4.3.2. BBN

Julio Escobar [Escobar,92] at Bolt Beranek and Newman Inc (BBN) has described a Flow Synchronisation Protocol that provides synchronised data delivery to multiple sites. The protocol time-stamps data at the source and equalises delays at the destination based on synchronised network clocks [Mills,91].

Multiple flows are synchronised by time-stamping the information as it leaves the source machine and calculating the maximum delay that the data will take to travel from source to destination. Once the data arrives at the destination, the protocol holds the data until the local time is equal to the timestamp plus the maximum delay informed to the protocol by a controlling device. This way multiple flows with variations in delay always appear to have the same overall delay.

Flows with destinations in the same synchronisation group are synchronised to each other by the protocol. Processors in the same synchronisation group regularly exchange information to compute a common synchronisation delay, allowing the equalisation delay to change dynamically to reflect changes in the communication sub-system.

The delayed delivery provides synchronised data delivery to multiple destinations, although the protocol does not account for delays introduced by the application that processes the information. Such delays may greatly interfere with the final presentation. In this scheme, applications have to build their own synchronisation scheme on top of the synchronisation protocol in order to guarantee the end-user synchronisation required during presentations.

4.3.3. Lancaster University

At Lancaster University, an orchestrator mechanism that dynamically manages and co-ordinates the information flow in a distributed multimedia session has been developed[Campbell,92]. The mechanism relies on the ability to establish a consumption ratio between multiple related streams (e.g.: ten sound samples for each video frame). For each continuous media stream a target is given for a specified interval, these targets ensure that each stream is within a time limit imposed by a master clock.

The mechanism is divided in two layers: the high level orchestrator (HLO) and the low level orchestrator (LLO). The HLO is responsible for interfacing the application to the orchestration sub-system at orchestration initiation time. The HLO elects a node, know as the orchestrating node, from which the multiple streams will be coordinated. The orchestrating node supplies the LLO with rate targets over specified intervals for each orchestrated connection. The LLO attempts to meet the required rate target and reports its success or failure at the end of the interval.

The HLO and LLO schemes provide the network support for implementing media synchronisation over distributed multimedia system. However, the formal specification of the temporal synchronisation is left to be handled at the application level and hence at least one more synchronisation layer is required to provide end-user synchronisation.

4.3.4. University of Pennsylvania

An integrated multimedia approach has been developed at the University of Pennsylvania [Nahrstedt,92]. This work differs from the previous works in the sense that it provides synchronisation in an integrated end-to-end basis. Multiple media streams are combined into a single stream that contains the events related to a given unit of time. The scheme requires the multiplexing of multiple media streams at the source machine and the demultiplexing at the destination machine. The advantage is that most synchronisation problems are eliminated, since on arrival the data contains all information necessary for that given time.

The approach consists of a composition and a decomposition protocol. The composition protocol runs on the source machine and is responsible for integrating the different media

captured in real-time into an Integrated Multimedia Message (IMM). Each IMM contains all information produced at a given time in the source machine. The decomposition protocol receives the IMM, demultiplexes the information, and outputs the information to the user interface. The following figure demonstrates the interaction of the two protocols.

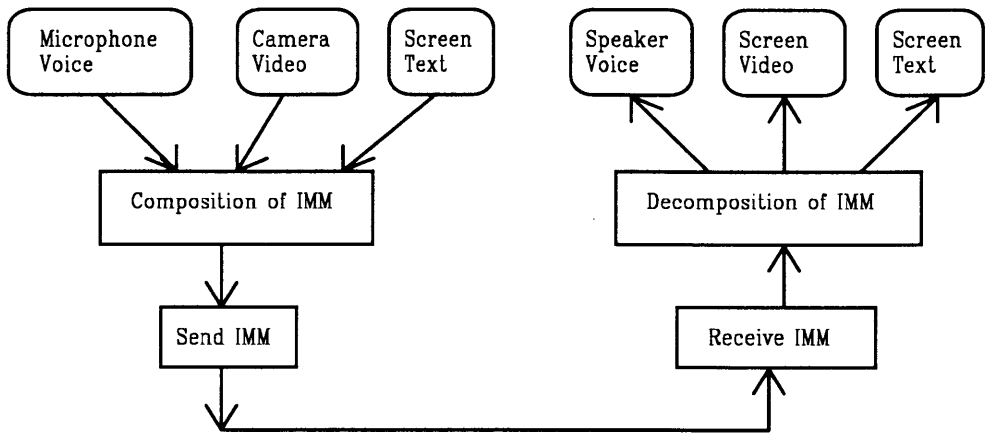


Figure 4.1. University of Pennsylvania - composition and decomposition protocols

Campbell [Campell,92] has pointed out some drawbacks of this scheme, including the overhead and complexity of multiplexing/demultiplexing and the non-availability of multiplexing when data originates from different sources. However, this approach may have strong advantages when used in real-time networked multimedia systems.

4.3.5. University of California at San Diego

The design of techniques and protocols for providing intra-stream and inter-stream synchronisation in distributed multimedia systems is being investigated at UCSD [Ramanathan,93]. A feedback technique, which transmits feedback units back to the server, is used to dynamically adjust and maintain the playback continuity. The technique does not assume the use of synchronised clocks, but assumes that communication delay can be restricted within a minimum and a maximum threshold.

Synchronisation is controlled by the destination node returning feedback units to the server when a given media unit is played back. Feedback units allow the server to estimate the playback time of the media unit, hence the multimedia server can adjust the transmission time of sub-sequent media units to avoid buffer overruns or starvation at the destination.

The temporal relation between multiple data streams is represented in the form of Relative Time Stamps (RTS), where the RTS of a media unit represents its time of playback relative to the commencement of the playback.

This work differs from other work in the sense that synchronisation is performed in the node that transmits the media units rather than on the destination machine. This approach is particularly suited for systems where the destination nodes do not have enough resources to implement complex synchronisation techniques.

4.3.6. University of California at Berkeley

At UCB a suite of real-time protocols is being developed [Ferrari,92]. The suite consists of a Real Time Channel Administration Protocol (RCAP), a Real Time Control Message Protocol (RTCMP), a Continuous Media Transport Protocol (CMTP) [Wolfinger,92], a Real Time Message Protocol (RTMP), and a Real Time Internet Protocol (RTIP) [Zhang,93]. The RCAP is a control protocol that establishes end-to-end connections and reserves the resources necessary for the connection. The data transfers at the internetwork layer are performed by the RTIP, which schedules data packets for transmission according to the resource reservation made by the RCAP. The RTMP is a message based protocol providing real-time message transmission between two endpoints. The CMTP offers a stream based interface and a time-driven mechanism suitable for transmission of continuous media streams. The RTCMP provides control functions similar to those of ICMP in the internet suite.

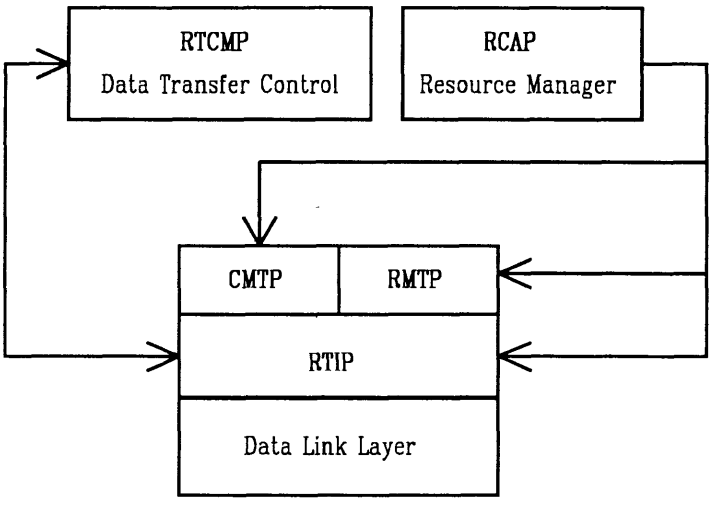


Figure 4.2. The Tenet real-time protocol suite

For this thesis, the CMTP is the most interesting protocol because it provides the intra-stream synchronisation required for continuous media streams. The CMTP provides unreliable, in sequence transfer, simplex connection between two ends with performance guarantee on loss, delay, and throughput. Traffic and performance parameters are defined in relation to two basic units: the stream data unit (STDU) and the periodicity of the transmissions. An STDU transmitted by the sender at time t (i.e. relative to the beginning of transmission) has to be received at the destination at time $\Delta 0 + t$, where $\Delta 0$ is the time corresponding to the first STDU delivered to the application.

The strongest aspect of the CMTP is its ability to reserve the resources required for the connection and its capacity to notify the application when an error occurs in the data stream or when part of the information was lost during the transmission. However, the CMTP does not support inter-stream synchronisation in any form. It is necessary to build a synchronisation

layer on top of the CMTP layer to achieve the degree of synchronisation required by multimedia applications.

4.4. Synchronisation at the presentation level

4.4.1. University of California at Berkeley

Anderson [Anderson,91] has described a continuous media I/O server that controls access to I/O devices and provides synchronisation for concurrent continuous media streams. The continuous media I/O server has functions similar to the X-Windows server and the authors suggest that the two servers could be merged into a single server. The ACME server (Abstraction for Continuous Media) controls the access to several physical devices (i.e. speakers, microphones, video displays, and video cameras) and allows clients to create logical devices bound to single physical devices.

ACME divides the data streams into media units, and each media unit has a timestamp for playback or a timestamp expressing the time the media unit was generated. The ACME synchronisation abstraction is called a Logical Time System (LTS). Each logical device is bound to an LTS, and an LTS can have several logical devices (both input and output). Logical devices attached to the same LTS are synchronised in the sense that media units with the same timestamp are displayed or retrieved at about the same time.

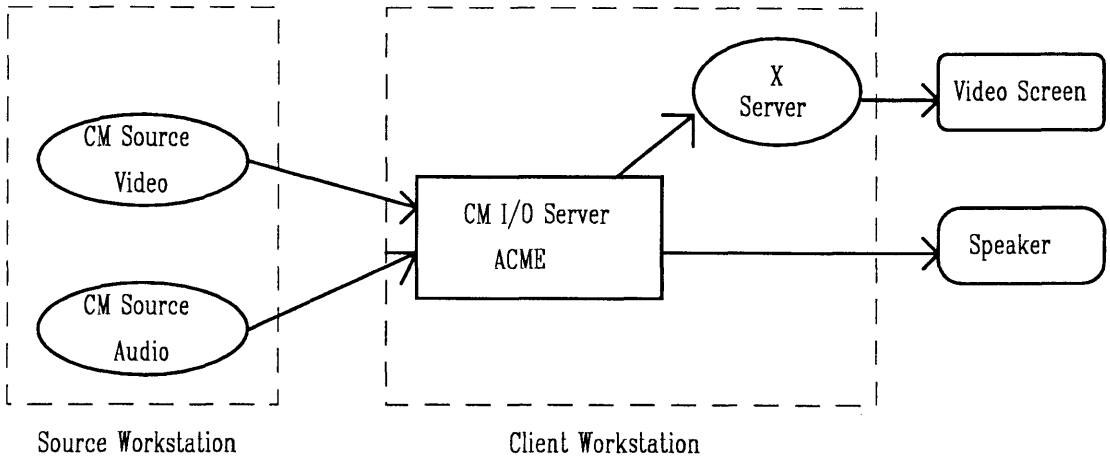


Figure 4.3. The continuous media I/O server

The ACME server maintains synchronisation by skipping frames or pausing the presentation. When the logical device time is running ahead of the LTS, the server pauses the presentation on the logical device until the LTS time catches up. On the other hand, if the logical device time is running behind the LTS, the server skips some data units in the logical device so that it can catch up with the LTS.

A more recent work on the continuous media I/O server has been done by Rowe [Rowe,92]. This work differs from Anderson's work in three aspects: the use of the Network Time Protocol (NTP) [Mills,91]; a new data model for synchronisation; and a network protocol dedicated to continuous media transmission. The use of the NTP allows time to be synchronised between the I/O server and the sources of the data streams so that the timestamp associated with the data at the source can easily be interpreted at the server machine.

Data streams are now organised in clips, where each clip is composed of a sequence of frames. A frame is a playable unit similar to the media unit concept used in ACME. The duration of each clip is stored so that applications can support operations to seek a particular time within the presentation. The continuous media network protocol provides an adaptive rate control based on the available resources on the network. The I/O server calculates a penalty rate based on the performance of the logical device and the occurrence of packet loss during transmission. The calculated penalty is regularly send back to the source process, which then adjusts the transmission rate based on the penalty rate.

We believe that the continuous media I/O server approach is the most comprehensive work in providing end-user media synchronisation in distributed multimedia systems. The continuous media I/O server effectively controls the pace of the presentation and can perform the required actions to maintain the correct synchronisation. The downfall is that the I/O server has to contain code to handle every single type of continuous media manipulated by the server, including a MPEG video decoder, an H.261 video decoder, and many types of audio servers. These may significantly increase the size and the complexity of the server code, potentially decreasing the total performance of the server.

4.4.2. IBM European Networking Center

Steinmetz [Steinmetz,90] at IBM ENC has concentrated on programming language synchronisation constructs to express media synchronisation. Media streams are seen as objects and each object executes its own synchronisation operations. Steinmetz describes two new elements that need to be addressed by synchronisation mechanisms: restricted blocking and real-time semantics.

Restricted blocking is related to what should happen when a process has to wait for an event to take place. Restricted blocking lets the programmer specify the action to be taken by the process while it is waiting for the synchronisation event.

Real-time semantics is concerned with the real-time characteristics associated with multimedia synchronisation. Three time parameters have being incorporated into the language: *timemin*, *timeave*, and *timemax*. *Timemin* is the minimum acceptable delay between synchronisation events. *Timeave* is the ideal delay between synchronisation events. *Timemax* is the maximum acceptable delay between synchronisation events. The following program shows a synchronisation between an object showing full-motion video and audio signal about an audio visual presentation. The object that arrives at the synchronisation point (i.e. the

SYNCHRONIZE statement) first will execute the operation specified after the statement WHILE_WAITING.

```
program of object A:                program of object B:
-- full-motion video                -- audio
-- from HARDDISK                    -- from CD-ROM
...                                 ...
display (audio_visual)              play(audio_visual)
  SYNCHRONIZE                       SYNCHRONIZE
  WITH object B                       WITH object A
  AT end                              AT end
  MODE                                MODE
    restricted blocking                 restricted blocking
  WHILE_WAITING                     WHILE_WAITING
    display_last_image                  play(music_Bach)
  TIMEMIN 0                         TIMEMIN 0
  TIMEMAX 1s                        TIMEMAX 2s
  TIMEAVE 0
  EXCEPTION                         EXCEPTION
    display_last_image                  play(music_Bach)
  ...                               ...
```

Figure 4.4. IBM ENC synchronisation language syntax

The general approach of the work is to consider that synchronisation operations are part of the multimedia object and are therefore executed by the object interested in the synchronisation using the syntax construction provided by the synchronisation language. One problem with this approach is that the synchronisation language is directly dependent on the hardware and operating system where it is running. Depending on the resources available, the language implementation can be made more or less efficient and the resulting implementation will impact directly upon how well the synchronisation can be performed.

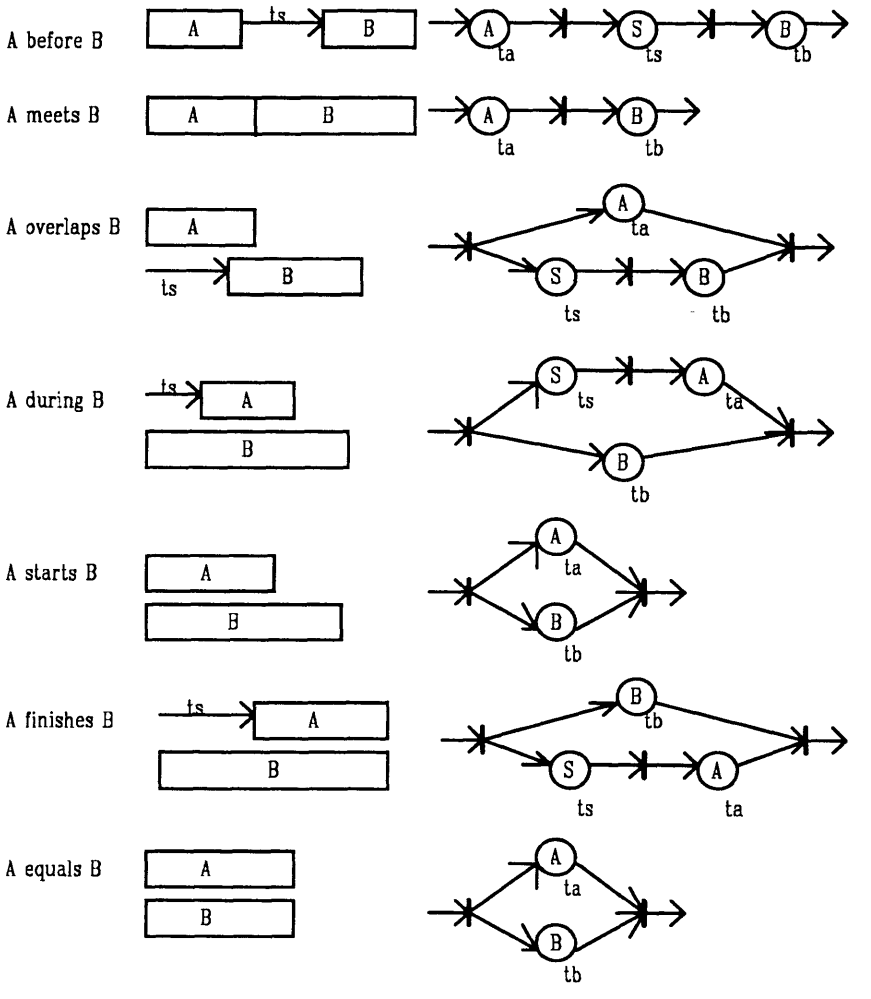
4.4.3. Cambridge University

Nicolau [Nicolau,90] has proposed a scheme for synchronisation that divides continuous media streams into two levels: Physical Synchronisation Frames (PSF) and Logical Synchronisation Frames (LSF). A PSF is the basic stream unit and is expected to be generated and transmitted at fixed time intervals. An LSF consists of a number of physical synchronisation frames. PSF's are intended as the unit of synchronisation within the communication sub-system, whereas LSF's are the unit of synchronisation for the controlling application.

This work defines an organisation for achieving multimedia synchronisation, however it does not addresses specific synchronisation services for applications.

4.4.4. Syracuse University

In this work, Little [Little,90a][Little,91b] proposes a graphical representation of the formal relationship for multimedia synchronisation. The graphical representation is based in a modified version of the Petri Nets formalism called Object Composition Petri Nets (OCPN) which includes a third dimension for representing the temporal relationship associated with multimedia synchronisation. OCPN fundamentals are based on the temporal relationship represented by Figure 4.5.



t: duration of the event

Figure 4.5. Temporal relations and correspondent Petri Nets

Presentation of an OCPN object is based in a real-time scheduling that accounts for the temporal constraints of the event, including deadline, minimum delay, and maximum delay.

Little's work is very significant for the design of formal relationship in multimedia applications. The OCPN formalism also fits well into the storage model described in Section 4.2.3. One limitation of this approach is that an OCPN structure is static and cannot be

modified during the presentation. If user interaction is necessary, it must be built into the application and the application must dynamically rebuild the OCPN relationship.

4.4.5. CNET, France

At the Centre National d'Etudes des Télécommunications, research is in progress to develop a multimedia programming environment that supports the real-time synchronisation required by multimedia applications [Horn,93]. The work focuses on a programming language that hides the characteristics of the underlying infrastructure so that application programmers do not need to know the characteristics of the hardware or of the operating system. CNET uses a real-time procedural language known as Esterel to achieve this goal.

An Esterel program can be viewed as a collection of parallel processes that communicate instantly with each other and with the environment. Esterel syntax allows arbitrary synchronisation points between multiple processes so that the required synchronisation can be performed.

The disadvantage with this approach is that the synchronisation is hardwired into the application code. Changing the final presentation requires changing the code of the application and hence requires programmers that can deal with Esterel language.

4.5. Assessment

The research efforts described in Section 4.2 are concerned with providing mechanisms to better handle continuous media streams. These mechanisms form the basis for correctly maintaining synchronisation. Although they do not directly address media synchronisation, they provide facilities which can be used to support the real-time requirements of media synchronisation.

Various researchers have addressed the communication issues for synchronisation. It is generally agreed that more flexible and configurable network protocols are necessary to understand and deal with continuous media data types. Three of the protocols described above (in Sections 4.3.2, 4.3.4, and 4.3.5) provide synchronised delivery of multiple data streams, while another three (described in Sections 4.3.1, 4.3.3, and 4.3.6) provide intra-stream synchronisation for independent virtual circuits, synchronisation between multiple streams is left for a higher level. Only one of the protocols presented (in Section 4.3.5) provides end-user synchronisation without the need for a higher synchronisation level.

Of the research on presentation levels, only two works (described in Sections 4.4.1 and 4.4.5) provide end-user synchronisation, the others are oriented towards specifying the formal relationship for presenting the information. Two of the works (described in Sections 4.4.2 and 4.4.5) propose the use of programming languages with built in synchronisation primitives suitable for multimedia synchronisation. The approach in Section 4.4.1 presents the concept of a multimedia server that controls the synchronisation specified by external applications.

As can be concluded by this review, a significant number of research efforts are tackling the problems of multimedia synchronisation. There is a consensus about how to transmit the information, but little has been achieved towards end-user synchronisation. We believe that there are three main outstanding issues need to be addressed to effectively provide end-user media synchronisation:

- There is a need to formally specify synchronisation so that the temporal relationships can be dynamically changed in response to user interaction.
- A synchronisation mechanism must closely monitor the issues of retrieval, communication, and CPU scheduling, so that it can maintain the best synchronisation support under a variety of adverse conditions.
- A synchronisation mechanism should concentrate on the temporal relationship between events and provide a set of functions that can be used by application developers to accurately specify both intra-stream and inter-stream synchronisation without requiring an intimate knowledge of the underlying hardware or operating system.

Chapter 5 describes a multimedia synchronisation framework that attempts to provide a synchronisation mechanism that addresses the above issues.