A TIME EFFICIENT ARCHITECTURE FOR MULTIMEDIA APPLICATIONS

Philippe OWEZARSKI, Michel DIAZ, Christophe CHASSOT

LAAS - CNRS
7 Avenue du Colonel Roche, 31077 Toulouse Cedex 04, France

e-mail : {owe, diaz, chassot}@laas.fr

Phone : +33 5 61 33 62 00

Fax : +33 5 61 33 64 11

Paper received january, 15th, 1997, revised august, 15th and october, 6th, 1997

Abstract

This paper presents an architecture that enforces time requirements and gives minimal end-to-end delays for multimedia applications. The layers and mechanisms allowing the system to fulfil the selected synchronisation, i.e. the logical relationships and timed interval semantics, are presented. The proposed approach relies on the use of a formal model based on extended time Petri nets, i.e. the Time Stream Petri Net model (TStreamPN), that allows the user to completely specify the time requirements of a given application. The architecture implements, in the application layer and on top of asynchronous environments, the requested quality of service (perceived by the user) with respect to time. At the transport layer, the use of a partial order transport service improves the reactive response of the communication transfers. Its principles are presented together with a pre-synchronisation sub-layer, that makes the partial order transport service match the applicative synchronisation requirements. Moreover, measurements on the implementation of a videoconference system show that the requirements of the quality of service are fulfilled.

Keywords

Multimedia communication architecture, multimedia synchronisation, real-time presentation, partial order transport, videoconference

1 Introduction

This paper aims to study and develop an approach and a set of adequate mechanisms to ensure to users a set of quality of service parameters like audio quality, video quality, end to end delay. What has been emphasised in this study is the requirements related to the temporal semantics. For this purpose, we have deeply analysed the multimedia synchronisation requirements of applications to first specify and second ensure both intra and inter-stream temporal requirements. Controlling the intra-stream temporal requirements consists of observing jitters in the stream and keeping them under an acceptable maximal value. Ensuring time inter-stream synchronisation consists in keeping under a maximum value the possible drifts that may appear between the different streams due to the cumulative effect of the jitters (as in lip synchronisation for example).

These requirements prove to be difficult to fulfil because the available distributed systems are asynchronous, from operating systems to wide area networks. As users and systems see synchronisation problems in a different way, it is necessary to perfectly specify the time requirements that have to be maintained by the system. The solution we have followed is to use a formal model of the user perceived requirements using explicit values of time, and to have the system maintain the corresponding temporal synchronisation down to the communication layer.
To specify the perceptual temporal synchronisation as seen from the user, i.e. the presentation behaviour at the user interface, a TStreamPN (Time Streams Petri Nets) model has been used to define a presentation model. From the properties of asynchronous operating systems, we will show that an application model, the applicative view, which is *different and derived from* the presentation model, is needed to infer a temporal scheduling of the presentation processes.

This paper will also show how user quality of service can be improved using a new multimedia transport protocol, taking into account the network problems and ensuring a well defined time delivery of the data : Such a transport protocol uses a partial order concept. In particular the representation of the logical order of the delivered objects will be derived from the application model. The remainder of this paper is as follows. First, the important quality of service (QoS) parameters related to time are presented in Section 2, that also shows the difficulties related to the use of asynchronous distributed systems.
Section 3 gives the general protocol architecture, from the transport layer to the user interface, and presents how it is possible to enforce the needed temporal requirements using mechanisms existing in the Sun Solaris 2 operating system. Note that all problems reported and solved have been observed on Sun Solaris 2.x workstations. As the proposed solutions use POSIX advanced system mechanisms, they should be portable on the hardware and software of several constructors. Finally, Section 4 develops an example implementation, the PNSVS system. This is a videoconferencing system that first takes advantage of a multimedia partial order transport layer to improve the applicative QoS, and second fulfils at the user interface the adequate

temporal synchronisation requirements. Finally, some remarks and perspectives conclude this paper.

2 The QoS problems

The QoS parameters of a multimedia application can be classified into two groups, i.e. as static and dynamic. The static parameters (page of text, image size and compression, audio encoding, etc.) directly impact the amount of resources required by the applications, but have no influence on the applicative algorithms. The dynamic parameters, which will be emphasised here because of their importance, greatly impact the amount of resources needed because programmers have to design the applications and fulfil their QoS requirements. The main QoS parameters are : multimedia synchronisation, dynamic presentation quality, and presentation end to end delay.

- The multimedia synchronisation parameters define the intra and inter-stream requirements of multimedia objects. These parameters are temporal : they define the maximum jitter acceptable on each stream and the maximum drift that can appear between two or more streams.
- The presentation quality parameters are essentially static parameters (image size and quality). However the maximum admissable number of discontinuities during the presentation of the media is a dynamic parameter. In fact, it appears that one lost data packet causes a synchronisation discontinuity that can be difficult to handle. To solve this problem, the lost data is usually replaced by other data temporally equivalent (generally the preceding data is presented twice), but this does not prevent a discontinuity in the video stream presentation. Discontinuities are hurtful for QoS, and their number has to be reduced as much as possible.
- The end to end delay defines the time between the grabbing of one object on the sending workstation and its display on the receiving one. This delay has to be controlled and must not be greater than a given value to provide good interactivity between the users.

Asynchronism of supports

All QoS parameters have to be taken into account by the application which has to include adequate mechanisms to ensure that user requirements are fulfilled. In the general case, the corresponding design and implementation should use asynchronous distributed systems for two reasons :

- almost all computer systems available, as LANs, Internet and operating systems on workstations (UNIX, DOS, …), are asynchronous ;
- with real time scheduling classes appearing in UNIX systems (like in Solaris 2), it becomes possible to implement real time requirements, which was not the case before [1, 2]. Also [3] has showed that high speed wide area synchronous systems cannot be realised.

Using asynchronous support to process isochronous multimedia data introduces several problems. The most important one comes from the temporal variability of computing : an operation has no upper bound. This variability or asynchronism appears at different levels in distributed systems, as communication support and protocols (no upper bound of the transit delay, etc.), non real-time operating systems such as UNIX (time shared scheduling, memory pages swapping, etc), and the application itself.

Model of the synchronisation

Such asynchronous support means that the implementation of synchronised multimedia applications needs to consider all problems due to temporal variability, such as jitter and drifts, and to link them to the requirements of multimedia objects.
To define the synchronisation properties of the multimedia objects themselves, a model allowing the author of a multimedia application to define the synchronisation requirements is needed. Several studies have already been realised in this domain, and some models have been proposed. The first of them uses a formal approach based on timed Petri nets. In particular, the OCPN (Object Composition Petri Net) model [4] and its extensions [5] [6] only considers nominal computing times, and does not address the computing time variability of asynchronous systems. Using the TPN (Time Petri Net) model [7] does not allow an easy modeling of inter-stream synchronisation between parallel streams.
The limitations of these models has led us to a new model [8], called Time Stream Petri Nets, (TStreamPN), providing good expression and modeling powers. TStreamPNs use temporal intervals that are located on the arcs leaving places (allowing us to compute each stream apart from the others). This allows the users to take into account both the temporal non determinism of distributed asynchronous systems and the presentation time variability of multimedia objects. The temporal intervals are triplets (x^s, n^s, y^s) called validity time intervals, where x^s, n^s and y^s are respectively the minimum, nominal and maximum presentation values.

The TStreamPN model is very well adapted to model multimedia stream requirements in asynchronous environments. Because of its high modeling and expressive power, this model easily expresses complex synchronisation scenarios.
For instance, the TStreamPN model has been used to describe the synchronisation requirements of a videoconference application called PNSVS (Petri Nets Synchronised Videoconferencing System). Figure 1 describes the user requirements of the videoconference application which provides a 10 image/s throughput with acceptable audio and video jitters of 10 ms per object (per unit of presentation). It can be seen that :

- 10 image/s defines the nominal presentation time of a video object, i.e. 100 ms ;

- the maximum intra-stream jitter determines the temporal validity intervals [90, 100, 110]; this interval appears in Figure 1 on all arcs leaving a place (for all objects of both streams).

In TStreamPNs, inter-streams temporal drifts can be controlled in a very precise way using 9 different inter-streams transition rules. They have well defined interval semantics depending on the position of the time intervals on the arcs leaving the places and entering the transition. Using these transition rules, it is possible to specify synchronisation mechanisms driven by the earliest stream ("or" synchronisation rules), the latest stream ("and" synchronisation rules) or by a given stream ("master" synchronisation rules). These synchronisation semantics define synchronisation instants led by an arc statically or dynamically chosen. For more details [9] gives a formal definition of the model and the different firing semantics.

Let us come back to the example of Figure 1 and assume : a) that the audio media is the most important one ; and b) that the inter-stream drift must remain under 100 ms, as less than 100 ms [10] of temporal gap between audio and video cannot be heard by a human being.
These QoS parameters determine the complete TStreamPN of Figure 1 :

- the inter-stream synchronisation is of "and-master" : a) the sound is more important than the video (master) ; and b) this rule tries as much as possible to respect the requirements of the video stream to reduce its discontinuities (and-master) ;
- the inter-stream drift must be less than 100 ms ; the maximum drift on 5 objects being 50 ms for each stream, the inter-streams drift cannot be greater than 100 ms.

The TStreamPN of Figure 1 allows the user (author) to precisely describe the (perceived) presentation requirements of one period of the videoconference application. Note that the formalism can be judged to be complex, but we feel that using such an approach is the only possible way to derive a general, generic and coherent architecture. Also, a general model for hypermedia objects is still more sophisticated [9] but its relationship to a general architecture has not yet been proposed .

From the model to the architecture

Since a global architecture must be defined in order to enforce these QoS parameters, the next section presents the mechanisms used to fulfil the QoS requirements and shows the original contribution of the paper in terms of layering, design and implementation. In particular it emphasises the respective functions of the transport and application layers and the way they share the constraints resulting from the time parameter. It will be seen that, as the advanced transport layer is able to deliver SDUs to the application layer as early as possible, it strongly supports the application layer to achieve the required jitter QoS. The videoconferencing system

that has been implemented on top of different LANs and WANs will illustrate the complete design.
A similar approach has been presented in [5] and in [6] but these approaches do not use an interval semantic for the time parameters and are based on a general, not fully defined, architecture. In particular, no precise definition of the transport layer is given and it is not related to the present transport implementations, such as UDP and TCP. Furthermore, the approach presented here models and precisely defines by different TStreamPNs the different layers and sub-layers. As a consequence it make explicit for the first time the formal temporal semantics of a global layered architecture.

3 A new distributed multimedia architecture

Section 2 has shown that the purpose of the architecture currently under study is to implement a distributed multimedia application having well defined synchronisation requirements. It appeared that formal appropriate models as TStreamPN can be used for that purpose. Let us now discuss the corresponding communication software.

Current transport solutions and their limits

The emergent generation of high speed networks now correctly address high throughputs. Recently, a lot of studies have been performed around the design of light weight transport protocols, such as NETBLT [11], VMTP [12] or XTP [13] [14]. These are more suited to support multimedia data transfers than TCP or TP4 protocols ; however, these proposals do not provide a sufficient solution to multimedia requirements. In particular, multimedia synchronization issues (both spatial and temporal ones) are not addressed and temporal constraints management mechanisms remain implementation dependant : a new generation of high speed, multimedia and co-operative architectures has then to be designed on top of high speed links. More recently, several communication architectures have been envisaged within different projects which may be classified as follows.

QoS architectures

Two main approaches have been proposed to design future advanced multimedia communication systems.

- The first one, called ALF/ILP [15], is based on the use of a network as simple as possible. The designer of the new protocols implements the software of the high speed multimedia applications at the user level, that is in the user space, on top of a simple network, such as IP. Therefore, the user is able to tune the software and develop the most appropriate solutions, these solutions depending on the application characteristics [16] ; nevertheless,

each user who develops a new application has to develop its own software. Even using high level languages [17], this needs a high global investment. This type of approach can be called "Network-Aware Application" (NAA) because the application must have the ability to adapt itself to actual network performance.

- The second solution explores the opposite view, i.e. aims at building an advanced communication system able to handle the requirements of sophisticated multimedia applications. This approach, of course more complex, leads to network support software more advanced and more general than existing platforms. The fundamental advantage is that the user software becomes much simpler, as the communication system provides some required functionalities ; also, any new user does not need to again develop these functionalities. This approach, which can be called "Application-Aware Networking" (AAN), has been and is still currently performed within several projects and architecture proposals, particularly OSI 95 [18], RACE CIO [19], BERKOM [20], TENET [21], QoS-A architecture [22], OMEGA [23], Function-based Communication SubSystem [24], End System Architecture [25] or CESAME [26].

This paper presents a global multimedia architecture that integrates the advanced transport AAN approach initiated in [27 - 31] and gives the corresponding implementation and experiments. In the following sections, we will show that the AAN approach is quite appropriate to build an architecture where the time requirements are of the utmost importance. For this, a new multimedia communication architecture will be derived from the synchronisation requirements, and a transport service and protocol will be designed to be able to indicate as soon as possible any loss of data in the network. How a distributed multimedia application can take advantage of such a communication service will then be presented.

3.1 A partial order based transport service and protocol

3.1.1 The Partial Order Connection concept

Currently transport protocols are either based on the connection-oriented (CO) paradigm or on the connectionless (CL) one :

- on one hand, TCP-like protocols provide their users with full reliability and sequential order ;
- on the other hand, UDP-like protocols introduce much less increase in transit delay or reduction in throughput but provide independent PDUs and no reliability guarantees.

The classification of these protocols using two axes, order and reliability, suggests that a conceptual family of transport layer protocols should exist between TCP and UDP. This extension, the *Partial Order Connection (POC)*, for which TCP and UDP appear to be special cases, has been introduced and theoretically investigated in [29, 30]. The basic principles of this new concept are as follows.

A conceptual extension of the connection concept

A POC is an end-to-end connection that allows its users to define and use for transferring data any partially ordered/partially reliable services from no order/no reliability (typically a UDP-like service) to total order/total reliability (typically a TCP-like service). In a POC, order and reliability appear as two specific QoS parameters specified by the service user during the connection set-up. Once known by both sending and receiving POC entities, order and reliability are translated into protocol parameters used to run the corresponding protocol mechanisms. In a POC, service data units (SDUs) can be delivered to the receiving user in an order that is different from the sending order : the acceptable difference between the submission sequence and the different but acceptable delivery sequences precisely results from the definition of the selected partial order.

A suitable concept with regard to multimedia applications features

It has been seen in Section 2 that the TStreamPN model formally describes multimedia synchronisation scenarios in asynchronous distributed systems. The underlying Petri net, i.e. the Petri net deduced from a TStreamPN model by removing all time values, provides a logical representation of a set of temporal synchronisation requirements. Futhermore, a Petri net represents a partial order.

Consider for instance the *multimedia object* pictured on the left part of Figure 2, composed of different *monomedia objects* (a logo, two fixed images and two video sequences) numbered from 1 to 99. The picture provides the expected object display at the receiving side. The right part of the figure gives the partial order deduced from the application defined TStreamPN model, that illustrates (among other things) that logo 1 and images 2 and 3 may be displayed at the receiving user side independently (in any order) as they are in parallel between the first and the last transition. For objects 4 to 99, the logical synchronisation requirements between the two video sequences are expressed by the intermediate transitions linking objects 4 and 5 to objects 6 and 7, etc.

In a partial order transport connection, objects (typically transport SDUs) may be delivered to the service user in any sequence consistent with both spatial and temporal synchronisation requirements, these different delivery sequences lead to transfer speed-up and save resources at both sending and receiving sides. [27 - 30] illustrate this last point through multiple examples and two different theoretical analysese.

[32] considers a system where a multimedia document (describing routine maintenance procedures performed on a car) is being retrieved from a remote server over the Internet and displayed in real-time as the content arrives at a user workstation**.** Particularly, it shows that

using a network layer service whose loss rate is high, a partially ordered/partially reliable protocol provides for graceful degradation and simplifies application development by providing appropriate mechanisms for synchronization, reordering and reliability. [33] presents an analytical study of a partially reliable transport service provided by sender-based loss detection and recovery. It is shown that such a service provides considerable throughput, admission rate and delay improvement over reliable transport service when the underlying network service is lossy and the application has a high loss tolerance.

3.1.2 The MM-POC architecture

As multimedia application requirements are multiple and diversified with respect to the data they imply, it is necessary to define different transfer characteristics for each of their flows. For instance, assume an application to be composed of partially synchronised text and video : the transport service has to guarantee both a perfect reliability with respect to the text transfer, and a sufficient enough throughput with respect to the video. However, a totally reliable transport connection is not needed for the video flow and a high speed transfer is not a major requirement for text-like communication. Moreover, one can mention that the use of a reliable service would imply transmission latency, inconsistent with an acceptable high speed video distributed application.

Addressing this point, most recent research led to either an extension of the QoS concept [18, 34] or to the proposal of new communication architectures [19 - 25]. As far as this latter point is concerned, two kinds of architecture have been developed :

- the first provides its users with a given set of service profiles, each of them being able to handle requirements of a specific data flow. For instance, [21] defines two service profiles, *real-time* and *non real-time*, respectively dedicated to (temporally) constrained data transfers and unconstrained data transfers ;
- the second defines a transport interface whose parameters (throughput, transit delay, or transit delay jitter for instance) have to be specified and then negotiated for each flow between service users and providers [19, 22]. Note that these use investigations around the QoS concept of [18].

Although pursuing the AAN philosophy, these two approaches do not tackle inter-flow synchronisation issues at the transport level ; indeed, QoS parameters are defined for each flow, but none of them takes into account dependency relationships between these flows.

The transport architecture which is presented here differs from the previous ones on this specific point : it integrates the dependency between the flows at the transport layer. It is based on the use of the TStreamPN model at different levels of the communication system, and particularly at

the transport level, where a multimedia synchronization management is introduced. The resulting multimedia transport architecture is detailed in the following section (3.1.2.1).
Let us note that inter-flow synchronization issues have also been tackled at the transport level in other work, using the multimedia connection concept. From a Petri net based synchronization model (*RTSM* : Real Time Synchronization Model), [5] proposes a transport architecture providing a multimedia synchronization service ; this architecture, older than ours, differs on several points. First, synchronization relationships are not transmitted to the receiving transport entity at connection set-up ; as a result, the receiving entity has to deal with synchronization control according to a lot of protocol control information. The second and major point concerns time management. Our architecture does not tackle temporal constraints because of the asynchrony of systems, but provides its users with a logical multimedia synchronization service, temporal constraints being managed at the service user level. Let us now detail the design principles of the transport architecture proposed in this paper.

3.1.2.1 Design principles of a MM-POC

In order to tackle the "multi-flow" aspect of a multimedia application, a multimedia transport architecture providing a set of QoS is needed, each of them being dedicated to one of the different flows of the application.

Our transport architecture (named MM-POC as *MultiMedia Partial Order Connection*) is based on a multimedia connection concept, implying the set-up and then a specific co-ordination, at the transport level, of several monomedia connections, each of them providing a specific QoS. As an example, consider the MM-POC given Figure 3 ; in this example, three monomedia connections (in fact, partial order transport connections) with a given QoS have been established, each of them being able to provide transport support for a specific data flow (for instance, a video, an audio and a text-like data flow). The co-ordination of the different POCs (illustrated on the figure by a dotted line) is based on the management at the transport level of "order" and "reliability" between the connections.

Order management

The "order" parameter is managed at two different conceptual levels : (1) within each of the monomedia connections (being then a monomedia POC) and (2) between these connections. This management takes into account at the transport level both intra- and inter-flow logical synchronisation requirements, which can be deduced from a TStreamPN model of the application.

Consider for instance the partial order given on the right part of Figure 4, deduced from a TStreamPN model of the object illustrated on the left part. If one colour is used to indicate a

specific QoS requirement, the associated MM-POC will be made of four monomedia POCs, each providing transport support of the corresponding coloured objects (SDUs) In this example :

- "SDU 6 has to be delivered after SDU 4" is an example of an intra-flow dependency relationship ; such requirements are managed within the corresponding monomedia POC (here the one providing the white QoS) ;
- "SDU 6 has to be delivered after SDU 5" is another example of an inter-flow dependency relationship ; such requirements have to be managed at a higher conceptual level than the POC one, but still within the MM-POC.

We also define an application programming interface (API) allowing service users to set up, use and release a multimedia partial order transport connection. The defined service primitives include "order" (among others) as service parameters. Service users may then select the most suitable multimedia partial order service with respect to temporal synchronisation requirements expressed by the application, that is the one whose "order" parameter is deduced from a TStreamPN model of the application.

Reliability management

In a monomedia POC, the protocol does not need to recover all protocol data unit (PDU) losses when the resulting lost SDUs do not generate a *degradation* of the selected reliability (*degradation* meaning that the required reliability is out of the requested boundaries). [30] shows how reliability may be managed in two different manners, resulting in each case in a transit delay improvement at the cost of an acceptable reliability decrease.

Both error control mechanisms are based on the following rule :

"delivery of a given SDU makes obsolete (ignores) all SDUs that are not yet delivered (they can be lost or not) preceding it in the multimedia partial order".

When processing an out of partial order SDU (i.e. not deliverable with respect to the multimedia partial order), the delivery order mechanism delivers it to the transport service user if the number of SDUs made obsolete does not exceed the maximum loss level on each POC. Such a process allows the service provider to deliver user data as soon as possible, at the potential cost of an acceptable loss level. In other words, transit delay is decreased at the cost of an acceptable reliability degradation, still respecting order constraints as they are expressed through a TStreamPN model of the application. To control the maximum acceptable loss level, two reliability management mechanisms have been proposed : *media per media* and *per group of media* ; these two mechanisms are now described and analysed.

First, assume a multimedia partial order connection to be composed of n monomedia POCs, each of them being identified by an indice i (i varying from 1 to n). Suppose now that reliability QoS is expressed by the maximum number of consecutive SDU losses, say k_i, the service user may tolerate on POC_i .

"Media per media" reliability management

When processing delivery of an out of partial order SDU, say A, on POC_i, the protocol will deliver A if PDUs made obsolete on POC_i still fulfil reliability requirements. That is, with our reliability definition assumption, if the number of consecutive SDUs made obsolete does not exceed k_i. However, it cannot deliver it if this delivery would need the loss declaration of one or more SDUs on any of the other POCs. A *media per media* reliability is thus defined.

"Per group of media" reliability management

Generalising the previous approach, the second mechanism may now deliver SDU A even if it generates a tolerable number of losses of one or more valid SDUs on any POC. That is, still with our reliability definition, when the number of valid consecutive SDUs made obsolete on POC_j does not exceed k_j for j from 1 to n. This reliability management is said to be *per group of media*, the group including here the n monomedia POCs.

Independently of their implementation complexity and the processing time overhead they generate, both mechanisms may be compared as follows :

- when a *per group of media* error control is applied, transit delay is optimised on each POC at the cost of a maximal but acceptable reliability degradation ;
- differently, a *media per media* error control does not fully benefit from the partial reliability concept (transit delay is not optimised), but independence between POCs is preserved.

In conclusion, both mechanisms induce transit delay improvement while enforcing the reliability QoS on each POC. Allowing service users to be notified of (acceptable) losses as soon as possible, the MM-POC multimedia transport architecture makes possible a new management of temporal synchronisation requirements at the application layer. This point is developed in Section 3.2 and then illustrated through an example in Section 4.

3.1.2.2 Implementation of a MM-POC

The MM-POC protocol has been implemented in kernel space using the SUN SOLARIS stream concept. A minimal service interface allows the users to select a given partial order / partial reliability for each application data flow. In order to provide continuous media transport support, protocol mechanisms ensure flow continuity preservation, fast retransmissions and bandwidth saving, while enforcing both order and reliability QoS parameters. Major algorithms

and experimental results comparing the respective impact of *media per media* and *per group of media* reliability management on transit delay may be found in [35].

3.2 Global architecture of multimedia applications

After having discussed the transport layer, let us now consider a methodology aiming to build a multimedia application on top of a multimedia partial order connection, particularly with respect to the QoS and temporal synchronisation requirements.

3.2.1 The temporal synchronisation task in the multimedia architecture

In asynchronous distributed systems, temporal synchronisation operations have to be performed at the top level of the architecture, i.e. at the application level of the receiving machine. In fact, enforcing final temporal synchronisation operations at a lower level, for instance in the communication layer, is not useful. This is because such synchronised data will be desynchronised when passing in the operating system and in the multimedia presentation subsystems (e.g. an interface board). Moreover, the application layer is the only place where developers can master process scheduling (in the kernel, developers cannot control the scheduling of threads). However, after having been synchronised, data is sent to the multimedia subsystems through the kernel, and is again impacted by asynchronism (Figure 5).

In fact, it is impossible to control the behaviour of data in the kernel and in multimedia subsystems : data is computed by separate subsystems and managed in independent ways. Nevertheless, as these subsystems are managed by the kernel with high priority level interrupts (a higher priority than the time-sharing applications and system tasks), it is possible to assume that multimedia subsystems latency times are constant. This hypothesis has been verified on all hardware we have tested. As long as hardware solutions for display are not completely synchronous, the result of applicative synchronisation will depend on the truth of this hypothesis.

This assumption about multimedia subsystems latency reduces temporal synchronisation tasks to operations processed at the application level. Nevertheless, the application in user space has to be "weak synchronous". It is required to use (in user space) a real-time class (existing in recent operating system versions like Solaris 2). Using such a real-time (RT) scheduling class and the hypothesis about multimedia latencies, it is possible to design synchronisation mechanisms enforcing the temporal requirements of multimedia applications. This point will be detailed in the implementation part (Section 4.3) of the case study discussed in Section 4.

As a result of these asynchronous systems characteristics, the behaviour of the applicative synchronisation task can be different from the presentation one modeled in Section 2. Indeed, if latency times are different, the synchronisation scenario implemented at the application level can differ from the one existing at the interface between the user and the machine. The media processing times can be different, the inter-stream synchronisation can change, etc. It follows that the presentation synchronisation scenario modeled by a presentation TStreamPN will induce, at the application level, two different applicative synchronisation, and so two TStreamPN models. One models the synchronisation on the sending site, and the other on the receiving site [36].

Let us now consider the transport layer. Using a classical transport service such as UDP can lead to many applicative discontinuities when losses occur through the network. When data is missing, the normal protocol behaviour consists in waiting for the lost PDU, the waiting time being bounded by the maximum presentation time of the object. When the presentation time reaches its maximum, the data is considered as lost and exception handling is started which consists in presenting substitution data. However, the time equal to the maximum jitter allowed has been lost, and the end to end delay has grown. Consequently (and this is bad), the corresponding delay can provoke losses on its stream when performing inter-stream synchronisations or when handling end to end delay control mechanisms.
It follows that multimedia distributed applications require a transport service that indicates the lost data as soon as possible. It is shown in the next section how a partial order based transport service can fulfil this requirement.

3.2.2 Distributed multimedia applications and partial order transport protocols

The architecture developed to run multimedia applications on top of partial order transport protocols is given in Figure 6. Note that it does not directly interface the synchronisation layer on top of the partial order transport. The reason for this is that the transport service is not precise enough for handling the temporal requirements. A pre-synchronisation layer is required to compensate for the temporal deficiencies of the partial order transport.

The architecture has three conceptueal layers : transport, pre-synchonisation and synchronisation.

- The partial order transport opens high speed connections for communicating multimedia objects, and provides earliest delivery as has been explained in Section 3.1. Nevertheless, as it does not explicitly manage time, a partial order transport service does not provide users with temporal guarantees. Moreover, long sequences of losses are detected very late (when

receiving the first PDU following the sequence of losses), potentially generating important and unacceptable jitter.

- The pre-synchronisation layer has been added between the transport and the synchronisation layers to perform the temporal functionalities the transport layer does not ensure. This layer provides a temporal control on the data delivered or lost by the transport. It checks the maximum time between two transport indications (delivery or loss) and is able to detect long sequences of losses.
- The third level consists of the synchronisation task that has to present the flows by respecting the temporal requirements defined by the user, and modeled by an applicative TStreamPN.

Benefits of this architecture

Because of asynchronous systems, temporal synchronisation has to be performed at the top level of the application and some real-time provisions have to be used. Such an approach allows three major improvements : (1) the non synchronisable data is discarded and exception handling is started as soon as possible ; (2) processing is performed once and as early as possible (i.e. such that their effects will not be annihilated at the higher levels), in order to minimise time and ensure a maximum QoS ; (3) storage is reduced : only the pre-synchronisation layer manages buffers in the user space.

In the next section, we show how a point to point videoconference application can take advantage of this architecture and what benefits can be derived from it.

4 Case study : the PNSVS videoconference application

PNSVS is a point to point videoconference application ensuring synchronisation requirements as modeled by a presentation TStreamPN and using the architecture proposed in the preceding section.

The application TStreamPN model has been given in Figure 1, for 10 images per second, 10 ms maximum of jitter and an inter-stream drift of less than 100 ms.

4.1 Behaviour modeling of the synchronisation layer

In this TStreamPN, for a normal inter-stream synchronisation, any audio object *i* must be synchronised with image *i*. Nevertheless, the two multimedia boards do not have the same latency time : 50 ms for a video board and 250 ms for an audio board. Consequently, if the application process followed the presentation of Figure 1, the final presentation would not be

synchronised : the audio part would be 200 ms late with respect to the video part, i.e. audio object *i* would be synchronised with image *i*+2.

Latency times

To solve this problem, an artificial drift has to be introduced in the rendez-vous. The difference between the audio and the video board latency times being 200 ms, it is sufficient to synchronis the audio object *i* with the image *i*-2 : after having been handled by the presentation boards, the audio object *i* will be synchronised with the image *i.* In this example, the applicative TStreamPN modeling the synchronisation of the application, with the drifted rendez-vous, is given on Figure 7.

However, this example is a particular case as the difference between the latency times is a multiple of the presentation time. In the general case, for instance if the audio and video latency times equal respectively 230 and 50 ms, the difference between the given latency times is 180 ms. In fact, the drift that can be modeled in the rendez-vous corresponds to two "drift objects", i.e. a 200 ms drift. After this 200 ms drift, a 20 ms drift remains between the audio and video streams. To ensure the maximum inter-stream drift remains under 100 ms (cf. Figure 1), the TStreamPN inter-stream synchronisation period has to be modified. If the inter-stream synchronisation is enforced each 5 images, because of the 20 ms remaining drift, the inter-stream drift of the audio/video streams would be in the interval [-80 ms, 120 ms], outside the allowed 100 ms. The inter-stream synchronisation must be enforced every 4 images, in order to keep the inter-stream drift in the interval [-60 ms, 100 ms].
The presentation TStreamPN modeling the multimedia synchronisation scenario and the applicative TStreamPN modeling the applicative processes behaviour have different shapes, because of the drifted rendez-vous. Moreover, the inter-stream synchronisation period can be modified.

End to end delay control

To solve the jitter problem, PNSVS stores incoming data in buffers before presentation. However, storing the data increases the end to end presentation delay. This delay is the QoS parameter impacting the interactivity between the communicating users and implies the reception buffer sizes associated to each stream [10]. During the videoconference, the receiver controls that the number of objects stored in its buffers do not overrun a maximum bound, in order to control the end to end presentation delay.
Note that if the receiver becomes late, the new incoming data cannot be stored and it has to reduce its delay.

The end to end presentation delay is quite hard to control, and it must be kept as short as possible. Studies [10] have showed this delay must be under 250 ms and computing times must be reduced to their minimum values.

- Multimedia board latency times cannot be reduced by the application, neither can communication system delays.
- Reception computing times cannot be substantially reduced. Images are atomic data : it is not possible to reduce the decompression / presentation time. For audio streams, the atomic element is a sample of 8 or 16 bits, and segmenting each sample packet does not significantly reduce the reception computing time.
- It is however possible to reduce the sender computing times : not the grabbing/compression time of an image, but the preparation time of an audio packet as it depends on the packet size as follows : "the larger the packet is, the more time it requires to be produced". For instance, on the applicative TStreamPN of Figure 7 for which an inter-stream synchronisation period modification has been applied, the 100 ms audio packets (800 bytes) requires 100 ms to be produced and 2 or 3 ms to be packetised and sent. On the other side, if the computing time of one image (grabbing, compression, packetisation and sending) requires only 55 ms, the delay comes from the audio stream. As audio packets are non atomic, they can be divided. Dividing the 100 ms audio packet by 2 makes the audio computing time 52 ms, and the one on an image 55 ms. It is possible to gain almost 50 ms on the end to end presentation delay (Figure 8). Note that as this delay is induced by the atomic video stream, it cannot be reduced anyfurther.

Finally, these applicative TStreamPN given in Figure 8 models the behaviour of the synchronisation layer on the receiving workstation. Because of asynchronous behaviour of the operating system and multimedia, implementing synchronisation mechanisms respecting this applicative TStreamPN will induce, at the user interface level, the presentation scenario modeled in Figure 1.

To define the partial order transport behaviour, it is required to express the partial order the transport layer has to respect and its minimum QoS in term of reliability. To obtain the partial order requirements, a partial order determined from the previous TStreamPN has to be used. It is the one that defines the periodic part of the synchronisation between the two media. The reliability requirements are associated with the connections in the same way as the required throughput.

4.2 PNSVS architecture using a partial order transport

Because of operating system asynchronism, the required architecture to run PNSVS over a partial order transport integrates a pre-synchronisation sub-layer between the synchronisation and transport layers.
Given this synchronisation architecture, the modeling TStreamPN architecture has been extended in order to take into account the functionalities of these new layers.
The pre-synchronisation sub-layer has to detect late or lost data and to control end-to end delay. To model the pre-synchronisation sub-layer, a new TStreamPN is required : the pre-synchronisation TStreamPN. This new TStreamPN has the same shape as the receiving applicative TStreamPN, and contains the required parameters to be able to detect the late data and to control the end to end delay
To model the partial order transport behaviour, we must express the partial order the transport service has to respect, and its minimum QoS in terms of reliability . To model partial order constraints a Petri net determined from the pre-synchronisation TStreamPN is used. Reliability constraints are associated with the connections in the same way as the required throughput.
Figure 9 shows an entire example of the PNSVS modeling with the related TStreamPNs : applicative, pre-synchronisation and transport. For readability, not all parameters are given in the figure.

4.3 PNSVS implementation

The PNSVS application has been implemented on Sun Workstations (Sun SparcStation 10) running the Solaris 2.5 operating system, with Parallax video boards, and using an Ethernet and a 155 Mbits/s ATM network.
The synchronisation mechanisms aim to present audio and video objects while fulfilling their synchronisation requirements. In fact, the goal consists of respecting the QoS requirements defined by the presentation TStreamPN which is the same as the one given on Figure 1. The operating system is asynchronous, as no bound on computing times is ensured via the time-shared scheduling. It is required to use processes whose priorities are greater than the ones of system tasks, and to use a fully pre-emptive operating system. With the Solaris 2 operating system such a scheduling class is called real-time (RT). Nevertheless, even when they run with the RT priorities, the processes only own a few real-time characteristics : their essential feature is that their priority class is greater than the one of system tasks.
On the other hand, using the RT scheduling class can disturb the operating system because system tasks are deferred when a RT process runs. RT processing must therefore be kept short. For instance, if the workstation is overloaded by RT tasks, communications (in system class) will not be processed. Also, if a RT process makes a system call, it loses its RT feature, and gets the SYS scheduling class. The RT scheduling class is essential for respecting the temporal synchronisation requirements, but it has to be handled with care.

The architecture of PNSVS has been divided into sub-tasks, where :

- the audio and video stockers receive data from the network. Buffers are used to temporarily store the desynchronised received data to solve the jitter problem ;
- the presentation audio and video processes realise the required operations for the sound and picture presentations ;
- the real-time orchestration processes realise the synchronisation scenario modeled by the TStreamPN and control the presentation processes to respect the presentation temporal requirements ;
- the pre-synchronisation processes perform a temporal control of the stockers, and ensure that the time between two transport indications does not overstep a maximum value ; in fact, pre-synchronisation processes realise a temporal control of the stocker while the orchestration processes do the same on the presentation ones.

Intra-stream synchronisation

The proposed synchronisation approach gives a solution different to the ones used in real time systems [1, 2, 10]. In an asynchronous system, upper bounds do not exist. In order to respect its maximum presentation time, the presentation process needs a real-time timer with a maximal priority, thus overcoming the system asynchronism. As the orchestration process computing time and the timer expiration time are known and made equal to their physical lower bounds, it is not possible to overstep the maximum presentation time of multimedia objects.

Inter-stream synchronisation

The inter-stream synchronisation algorithm is based on a rendez-vous between the audio and video orchestration processes, with the semantic of the "and-master" firing rule. The principle of the temporal control is the same as the one described for the intra-stream synchronisation, except that only the audio orchestration process uses a real-time timer (because only the presentation requirements on the audio object must be respected). The audio orchestration process has to wait until the end of the audio presentation process, and until the rendez-vous of the video orchestration process (meaning that the video presentation process is completed) : then, it can fire the inter-stream transition to start the next period. Nevertheless, if the real-time time-out occurs the audio orchestration process has to kill the audio and/or video presentation processes and has to inform the video orchestration process to jump to the first object of the next synchronisation period.

4.4 Performance measurements

4.4.1 Synchronisation mechanisms evaluation

PNSVS is a videoconferencing application that can process 20 images/s (320 x 240 pixels and 24 bits coded colors) in one direction. The minimum end to end presentation delay obtained is around 400 ms and cannot be reduced because of the audio board latency time, of around 250 ms.

It is important to verify that all synchronisation mechanisms respect the temporal presentation requirements, and in particular, the quality of the intra and inter-streams synchronisation. The measurements have been realised in the case of a 10 images/s videoconference application where synchronisation requirements are modeled by the presentation TStreamPN of Figure 1.
Figure 10 shows, for the 100 first images, the jitter that appears on the presentation of each image. The measured jitter is the difference between the effective presentation duration and the nominal presentation time. This figure shows that the maximum jitter of 10 ms is never overstep, but the jitter is always negative. In fact in this experiment there is no network problem (no loss and jitter), and data is available when the presentation process needs it. Thus the anticipation mechanism, that makes it possible to stop the presentation of an object as soon as its minimal presentation time has been reached, always works. Variations are due to the real-time scheduler of Solaris 2.

Figure 11 shows the same experiment applied to the audio stream of PNSVS. As for the video, the intra-stream synchronisation requirements are always respected, but the jitter is always positive. In fact in this case, the firing of the intra-stream transition is caused by the audio port signal. With a time scale expressed in millisecond, this consumption is always equal to 100 ms. Variations are due to the time required by the system to take into account this information.

Figure 12 shows the curve representing the inter-stream drift for the audio and video sequence numbers. Figure 10 shows that the video jitter is always negative and Figure 11 shows the audio jitter is always positive. Thus, the inter-streams drift (difference between the audio and video objects presentation dates) is positive. This drift increases during each period of 5 objects (due to the cumulative effect of jitter) and it is eliminated at each inter-stream synchronisation. The inter-stream synchronisation requirements are perfectly respected because the drift never exceeds 50 ms, the maximum allowed value being less than 100 ms.

4.4.2 Overall architecture benefits evaluation

The temporal synchronisation mechanisms having been measured, it remains to evaluate the benefits of the overall architecture that includes the Partial Order Transport, the pre-synchronisation and the temporal synchronisation layers. As previously said, the use of a partial order transport (thanks to its earliest deliveries and losses mechanisms) avoid time losses at the application level. These time losses are really hurtful for QoS because the application has to discard some presentation objects (images or sounds) when it has not enough time to compute them (end to end delay control mechanisms, "and-master" inter-streams synchronisation schemes, etc.). Thus, in this section, the losses created by the synchronisation application will be measured in two cases : (1) when a partial order transport and (2) when a classical connectionless transport service (UDP) are used. To perform this evaluation, a network simulator allowing to simulate losses on the network has been used. This simulator is a Solaris 2 stream module, put on the sending machine (between the sending process and the ATM driver) that discards some packets. The PNSVS application evaluated is the one the temporal constraints expressed by the TSPN of Figure 1. The other parameters are : the maximum end to end delay 1 s, the audio latency 400 ms, and the video latency : 50 ms. With these values, the audio and video buffers can contain respectively 5 audio frames and 10 images. Results of measurements are described on Figure 13. It shows the applicative loss average in the 2 cases (using UDP and POC) depending on the network loss level.

Figure 13 shows that the QoS obtained by using POC is better than the one using UDP. In fact, when the application receives a loss indication from the partial order transport, it does not wait, and fires the TSPN transition as soon as possible (when the minimum presentation time is reached). While using UDP it has to wait the expiration of the timer associated to the maximum presentation duration. Using POC, each loss detection allows the application to "recover" the maximum allowed negative jitter ($n^s - x^s$), while using UDP it provokes a new time loss (equal to $y^s - n^s$).
Thus, the curve presenting the results of PNSVS using UDP grows linearly. The more network losses, the more time losses there are, and the more the application has to discard data to reduce the end to end delay.
Four comments can be made about the POC curve :
1) if the network is reliable, results using POC or UDP are equivalent ;
2) the applicative loss level decreases between 0 and 10 % of network losses. To explain this phenomenom, let us remember the results on audio and video jitter that are respectively 1 ms and -4 ms (on average). With such jitter values and a network loss level less than 10 %, the drift in a synchronisation period is positive (there is not enough network losses to make the application recover from the audio drift), and some applicative losses are created by the end to end delay control mechanism ;

3) the applicative loss level grows between 10 and 50 % of network losses. In this case, there are enough network losses to make the drift on a synchronisation period negative. However losses appear because of a famine problem : each time a network loss appears, using POC, the end to end delay decreases, and this progressively frees buffers. When buffers are empty, even if the application is waiting as much as possible for the next object, this will not arrive before the application replaces it by a substitution one (the last image for video, or an empty sound for audio stream) ;
4) finally, if the network loss level is greater than 50 %, long sequences (with more than 5 objects consecutively lost, 5 being the number of audio packets that can be stored in audio buffer) can appear. In this case, POC cannot detect all losses early enough, and the pre-synchronisation layer detects them. The principle of the pre-synchronisation mechanism is the same as the applicative synchronisation one. Pre-synchronisation is nevertheless more effective because it computes data earlier than the synchronisation layer. However, the drift on a synchronisation period can be positive and applicative losses are then provoked by the end to end delay control scheme.

Another test for the effect of network jitter on PNSVS has also been made (thanks to the network simulator). This test proves that PNSVS can always recover from jitter less than 100 ms. This result is quite interesting because, using ATM networks, such jitter is impossible.

5 Conclusion

This paper has presented adequate architectures and mechanisms to fulfil important synchronisation requirements for multimedia applications in asynchronous environments. To obtain the best possible QoS, the synchronisation architecture consists of two extreme layers : an applicative synchronisation layer that ensures the multimedia objects temporal requirements and a new multimedia advanced partial order transport layer. However, to interface the partial order transport service with the application needs, a pre-synchronisation level has been located between the application and the transport layers.
This architecture has been derived after considering a formal representation of the multimedia information. It has been shown that it is possible to model the behaviour of all layers of such a synchronisation architecture. The model that has been used here is a time Petri nets based model, the TStreamPN. A presentation TStreamPN, deduced from the user QoS, is used to model the presentation level multimedia synchronisation scenarios at the interface between the application and the user. Then, in the general case, a modified applicative representation (for the sender and the receiver) is deduced from it to model the synchronisation application behaviour. Finally, the receiver applicative model leads to the design of the partial order transport.

This concept has been used to build a videoconferencing system, PNSVS. Its implementation has been based on advanced system mechanisms, such as the real-time scheduling class of Solaris 2, and the threads mechanisms. The new transport architecture, service and protocol based on partial orders, running on the Solaris 2 streams mechanisms, provides the basis of the global architecture. PNSVS has been implemented on Sun Workstations (Sun SparcStation 10) with Parallax video boards, on top of an Ethernet and a 155 Mbits/s ATM network. This videoconferencing tool allows users to create conferences having up to 20 images/s, an audio stream at 64 Kbits/s, and perfectly fulfils the synchronisation requirements.
Future work concerns the theoretical and practical extension of PNSVS towards a multi-user videoconference application allowing many users to communicate with a guaranteed QoS.

Acknowledgements

The authors would like to thank Michael Fry for having greatly contributed to improve this text.

References

1. G. Coulson, G. S. Blair, P. Robin. "Micro-kernel support for continuous media in distributed systems", Computer networks and ISDN systems 26, 1994.
2. K. Jeffay, D.L. Stone, F.D. Smith. "Kernel support for live digital audio and video", Computer communications, Vol. 15, No. 6, July / August 1992.
3. Kang G. Shin, Parameswaran Ramanathan. "Real time computing : a new discipline of computer science engineering", Proceedings of the IEEE, Vol. 82, No. 1, January 1994.
4. T.D.C. Little, A. Ghafoor. "Synchronisation and storage models for multimedia objects", IEEE journal on selected areas in communications, Vol. 18, No. 3, april 1990.
5. C.C. Yang, J.H. Huang. "A multimedia Synchronization Model and Its Implementation in Transport Protocol", IEEE Journal on Selected Areas in Communications, vol.14, n°1, January 1996.
6 M. Woo, N. U. Qazi, A. Ghafoor, "A Synchronisation Framework for Communication of Pre-orchestrated Multimedia Information", IEEE Networks, January/February 1984, pp 52-61.
7. B. Berthomieu, M. Diaz. "Modelling and verification of time dependant systems using time Petri nets", IEEE transactions on software engineering, Vol. 17, No. 3, March 1991.
8. M. Diaz, P. Sénac, "Time stream Petri nets, a model for multimedia streams synchronisation", proceedings of MultiMedia modeling'93, Singapore, November 1993.
9. P. Sénac, M. Diaz, P. De Saqui-Sannes. "Toward a formal specification of multimedia synchronisation", Annals of telecommunications, May / June 1994.

10. K. Jeffay, D.L. Stone, F.D. Smith. "Transport and display mechanisms for multimedia conferencing across packet-switched networks", Computer networks and ISDN systems 26, 1994.
11. D.D. Clark, M.L. Lambert, L. Zhang. "NETBLT : a bulk data transfer protocol". RFC 998, March 1987.
12. Cheriton (D.R.), Williamson (C.L.). "VMTP as the Transport layer for high-performance distributed systems". IEEE Communications Magazine, pages 37-44, June 1989.
13. G. Chesson. "XTP/PE Design Considerations". IFIP International Workshop on Protocols for High-Speed Networks, Rüschlikon, May 9-11, 1989.
14. XTP protocol definition, Revision 3.6. Protocol Engines Incorporated, January 1992.
15. D.D. Clark, D.L. Tennenhouse. "Architectural Considerations for a New Generation of Protocols", Proceedings of ACM SIGCOMM, 1990.
16. C. Diot, C. Huitema, T. Turletti. "Network conscious Applications", HPCS workshop, Mystic, August 1995.
17. C. Diot, R. De Simone, C. Huitema. "Automated design of communication protocols using ESTEREL, Journal for high speed networks, vol.5, n°2, 1996.
18. A. Danthine. Ed. "The OSI95 Transport Sevice with Multimedia Support", Research Reports ESPRIT, Project 5341, OSI95, vol.1, Springer Verlag, 1994.
19. L. Henckel. "Multimedia Communication Platform : Specification of the Enhanced Broadcast Transport Service", RACE CIO Project 2060, Sept. 1993.
20. L. Delgrossi, J. Sandvoss. "The BERKOM-II Multimedia Transport System", Aug. 1993.
21. TENET group. "Recent and Current Research", University of California, Berkeley, April, USA, 1994.
22. A. Campbell, G. Coulson, D. Hutchinson. "A Quality of Service Architecture", ACM Computer Communications Review, April 1994.
23. K. Nahrstedt, J.Smith. "Design, Implementation and Experiences of the OMEGA End-Point Architecture", IEEE Journal on Selected Areas in Communications, vol.14, n°7, september 1996.
24. M. Zitterbart, B. Stiller, A.N. Tantawy. "A Model for Flexible High-Performance Communication Subsystems", IEEE Journal on Selected Areas in Communications, vol.11, n°4, May 1993.
25. G. Gopalakrishna and G. Parulkar. "A Framework for QoS Guarantees for Multimedia Applications within an Endsystem", GI Jahrestagung, Zurich, Switzerland, september 1995.
26. M. Diaz, G. Pays. "The Cesame Project : Formal design of high speed multimedia cooperative systems", Annals of telecommunications, Tome 49, n° 5-6, May / June 1994.

27. P.D.Amer, C.Chassot, T.Connolly, M.Diaz. "Partial order transport service for multimedia applications : reliable service". Proc. 2nd High Performance Distributed Computing Conference, July 1993.
28. P.D.Amer, C.Chassot, T.Connolly, M.Diaz. "Partial order transport service for multimedia applications : unreliable service". Proc. 3rd International Networking Conference, INET'93, Aug. 1993.
29. P.D. Amer, C. Chassot, T. Connolly, P. Conrad, M. Diaz. "Partial order transport service for multimedia and other applications", IEEE/ACM transactions on networking, Vol. 2, No. 5, October 1994.
30. M. Diaz, A. Lozes, C. Chassot, P.D. Amer. Partial Order Connections. "A new concept for High Speed and Multimedia Services and Protocols", Annals of telecommunications, Tome 49, n° 5-6, May / June 1994.
31. C. Chassot, M. Diaz, A. Lozes. "From the Partial Order Concept to Partial Order Multimedia Connections". Journal for high speed networks, vol.5, n°2, 1996.
32. P. Conrad, E. Golden, P. Amer, R. Marasli. "A multimedia document retrieval system using partially-ordered/partially-reliable transport service", Proc. Multimedia and Computing Networking 96, San Jose, January 1996.
33. R. Marasli, P.D. Amer, P. Conrad. "Retransmission-based partially reliable transport service : An analytic model", Proc. IEEE INFOCOM 96, San Fransico, March 1996.
34. M. Diaz, K. Drira, A. Lozes, C. Chassot. "Definition and representation of the quality of service for multimedia systems", 6th International conference on high speed networking, HPN'95, Palma de Mallorca (Balearic Islands), Spain, September 11-15, 1995.
35. M. Fournier, C. Chassot, A. Lozes, M. Diaz. "Multimedia Partial Order Transport Architecture : Design and Implementation", Protocol for high speed network'96, Sophia Antipolis, Nov. 1996.
36. P. Owezarski, M. Diaz. "Models for enforcing multimedia synchronisation in videoconference applications", proceedings of the 3rd MultiMedia Modeling conference, Towards the information superhighway (MMM'96), pp 85-100, World scientific editor, Toulouse, France, November 12-15, 1996.

FIGURES :

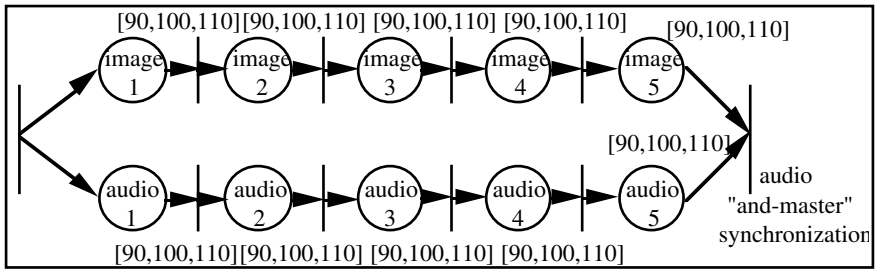

Figure 1. Videoconference TStreamPN example

Figure 2. : Partial order deduced from a TStreamPN model of a multimedia object

Figure 3. : Multimedia Partial Order Transport Architecture

Figure 4. : Multimedia partial order

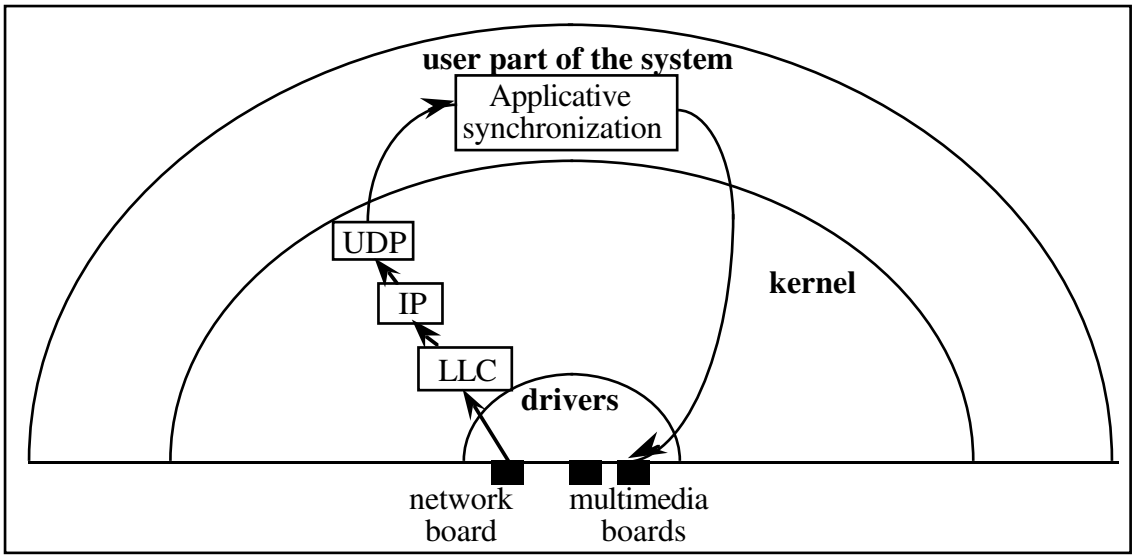

Figure 5. Transit of multimedia data in the operating system

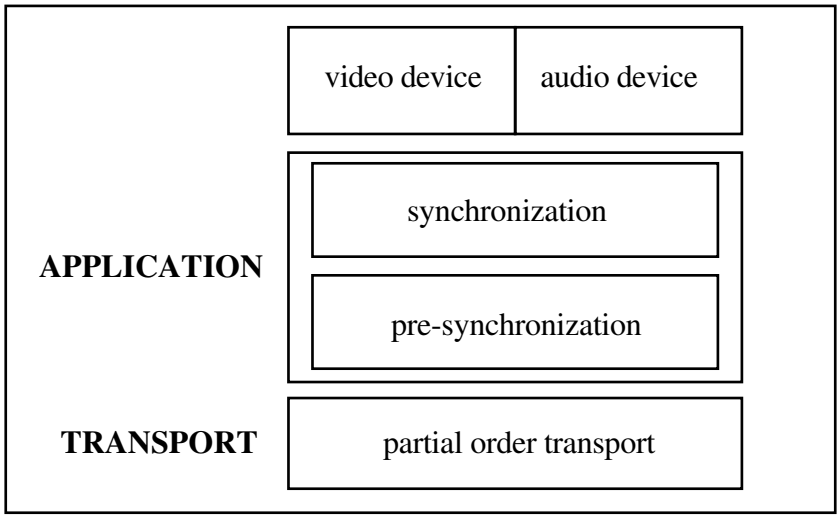

Figure 6. Architecture of a multimedia application on top of a partial order transport

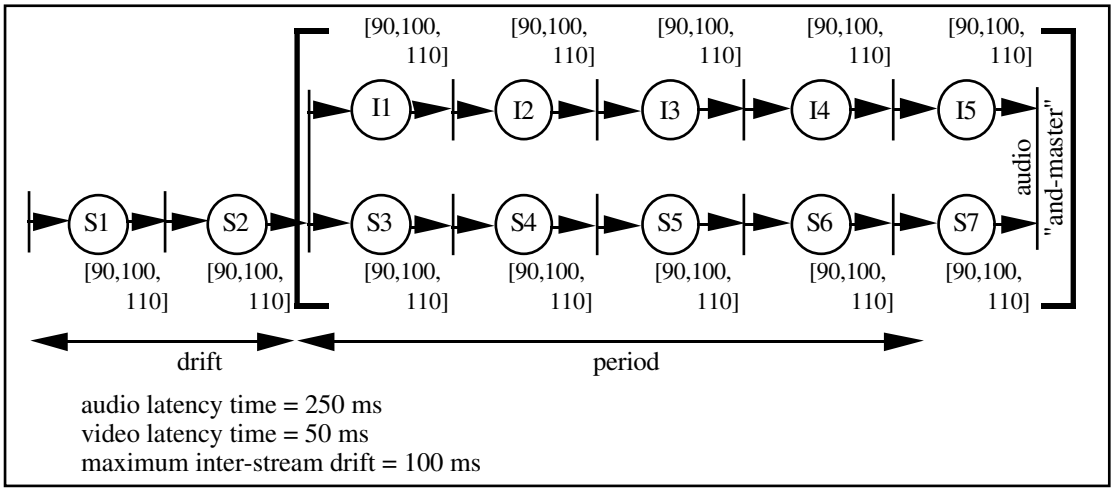

Figure 7. Applicative TStreamPN taking into account the hardware latency times

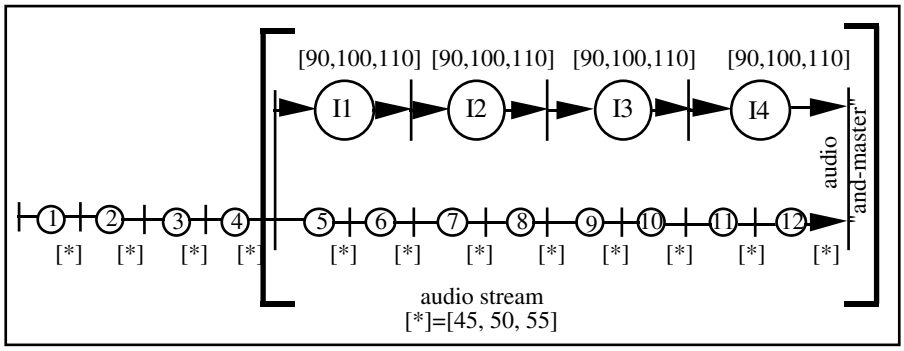

Figure 8. Applicative TStreamPN taking into account the end to end delay improvements

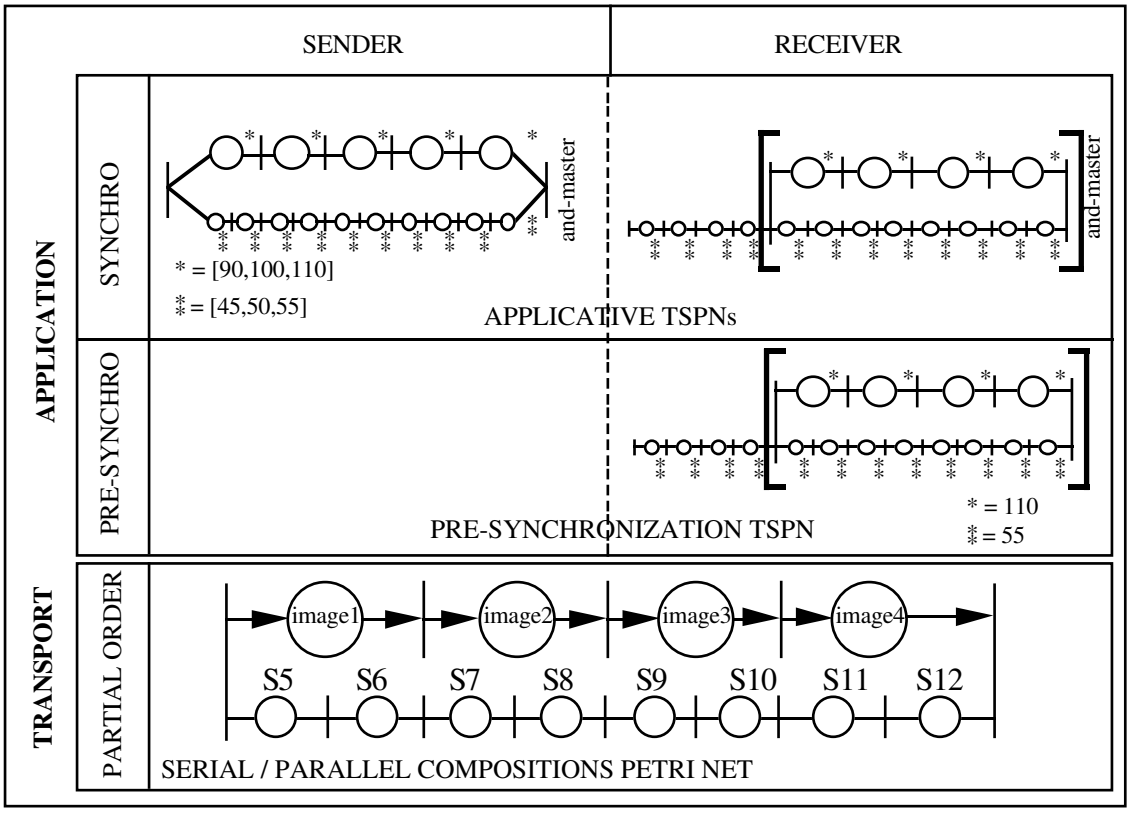

Figure 9. TStreamPN architecture

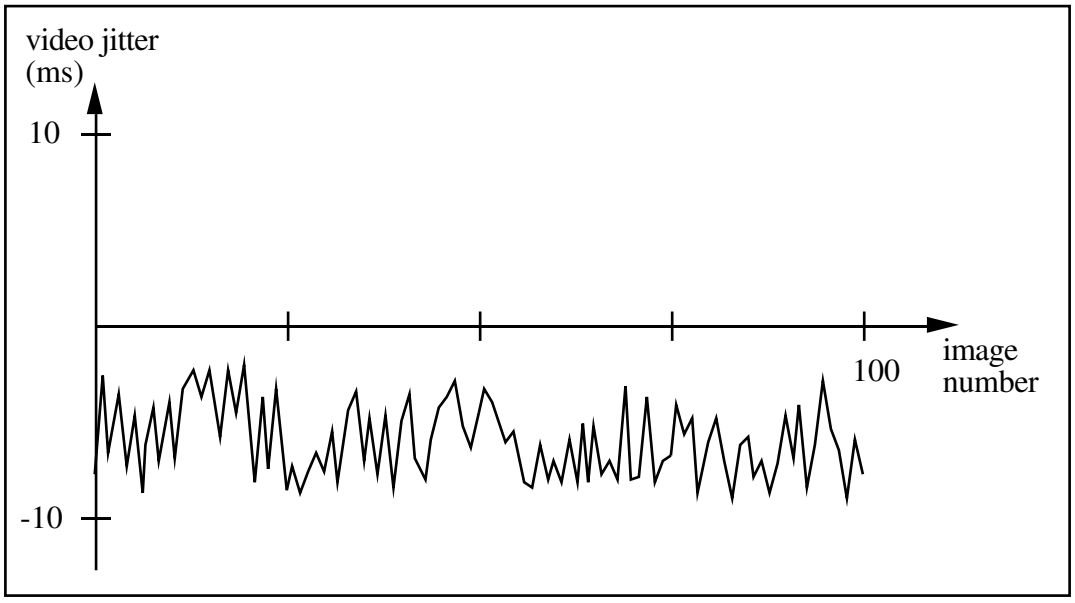

Figure 10. Measurement of the jitter on the PNSVS images

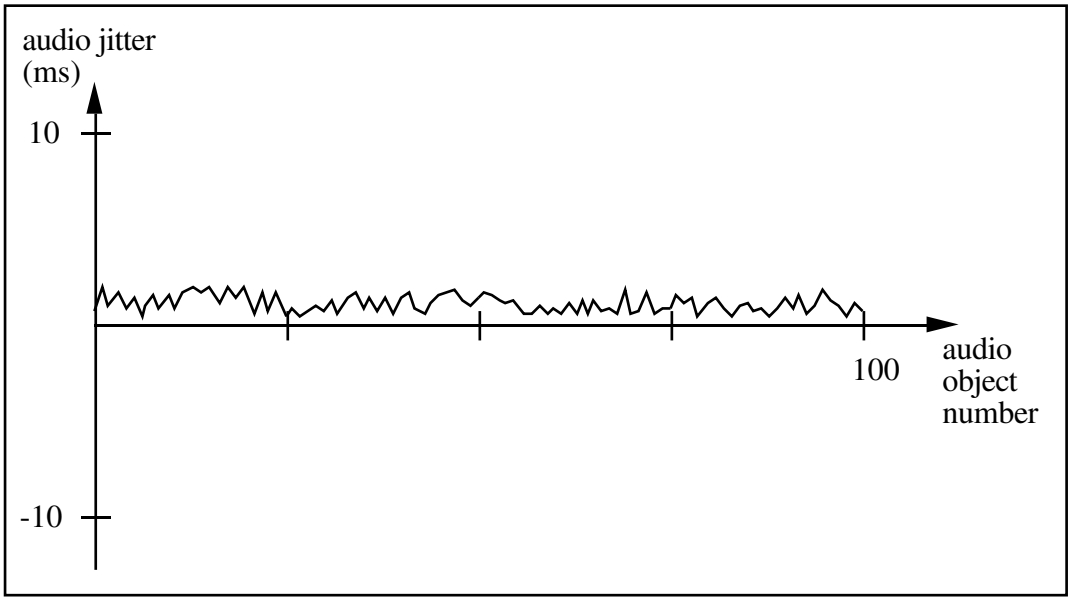

Figure 11. Measurement of the jitter on the PNSVS audio objects

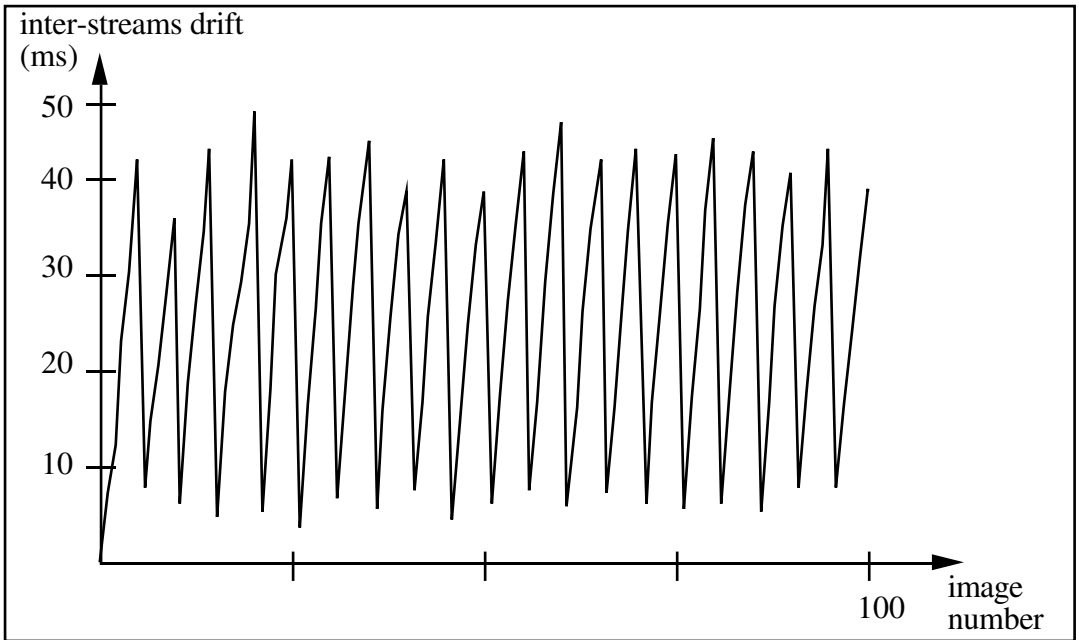

Figure 12. Measurement of the PNSVS inter-streams drift

% of network losses / % of applicative losses	0	5	10	20	30	40	50	60
POC	2	0,4	0,1	0,9	1,7	2,3	2,9	3,7
UDP	2	2,4	3,7	5,5	7,4	9,1	10,9	12,7

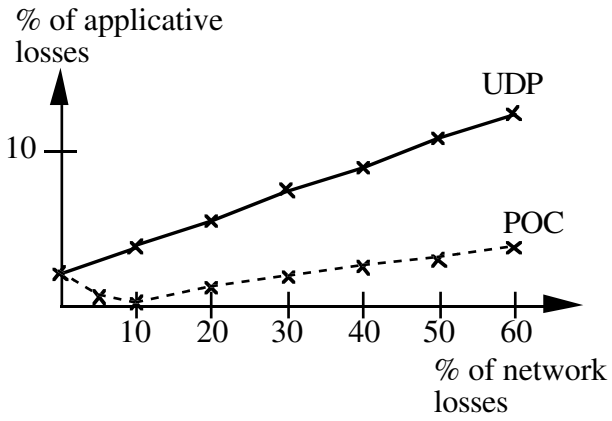

Figure 13. Overall architecture benefits evaluation