A Time-Efficient Architecture for Multimedia Applications

Philippe Owezarski, Michel Diaz, and Christophe Chassot

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 fulfill the selected synchronization, 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 presynchronization sublayer that makes the partial order transport service match the applicative synchronization requirements. Moreover, measurements on the implementation of a videoconference system show that the requirements of the quality of service are fulfilled.**

Index Terms—**Multimedia communication architecture, multimedia synchronization, partial order transport, real-time presentation, videoconference.**

I. 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, and end-to-end delay. What has been emphasized in this study are the requirements related to the temporal semantics. For this purpose, we have intensively analyzed the multimedia synchronization requirements of applications to first specify and second ensure both intra- and interstream temporal requirements. Controlling the intrastream temporal requirements consists of observing jitters in the stream, and keeping them under an acceptable maximal value. Ensuring time interstream synchronization consists of 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 synchronization, for example).

These requirements prove to be difficult to fulfill because the available distributed systems are asynchronous, from operating systems to wide area networks. As users and systems see synchronization 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 synchronization down to the communication layer.

To specify the perceptual temporal synchronization as seen from the user, i.e., the presentation behavior 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 II, which also shows the difficulties related to the use of asynchronous distributed systems.

Section III 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 IV 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 fulfills at the user interface the adequate temporal synchronization requirements. Finally, some remarks and perspectives conclude this paper.

II. 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 emphasized here because of their importance, greatly impact the amount of resources needed because programmers have

Manuscript received January 10, 1997; revised September 12, 1997.
The authors are with LAAS—CNRS, 31077 Toulouse Cedex 04, France.
Publisher Item Identifier S 0733-8716(98)00645-3.

to design the applications and fulfill their QoS requirements. The main QoS parameters are multimedia synchronization, dynamic presentation quality, and presentation end-to-end delay.

- The multimedia synchronization parameters define the intra- and interstream 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 admissible number of discontinuities during the presentation of the media is a dynamic parameter. In fact, it appears that one lost data packet causes a synchronization discontinuity that can be difficult to handle. To solve this problem, the lost data are usually replaced by other data temporally equivalent (generally, the preceding data are presented twice), but this does not prevent a discontinuity in the video stream presentation. Discontinuities are harmful to 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, such as LAN's, Internet, and operating systems on workstations (UNIX, DOS, etc.), 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 realized.

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 supports and protocols (no upper bound of the transit delay, etc.), nonreal-time operating systems such as UNIX (time-shared scheduling, memory pages swapping, etc), and the application itself.

Model of the Synchronization

Such asynchronous support means that the implementation of synchronized 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 synchronization properties of the multimedia objects themselves, a model allowing the author of a multimedia application to define the synchronization requirements is needed. Several studies have already been realized 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 consider nominal computing times, and do not address the computing time variability of asynchronous systems. Using the TPN (time Petri net) model [7] does not allow an easy modeling of interstream synchronization 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. TStreamPN's 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 nondeterminism 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 synchronization scenarios.

For instance, the TStreamPN model has been used to describe the synchronization requirements of a videoconference application called PNSVS (Petri nets synchronized videoconferencing system). Fig. 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/object (per unit of presentation). It can be seen that

- 10 images/s defines the nominal presentation time of a video object, i.e., 100 ms;
- the maximum intrastream jitter determines the temporal validity intervals [90, 100, 110]; this interval appears in Fig. 1 on all arcs leaving a place (for all objects of both streams).

In TStreamPN's, interstreams temporal drifts can be controlled in a very precise way using nine different interstream 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 synchronization mechanisms driven by the earliest stream ("or" synchronization rules), the latest stream ("and" synchronization rules), or by a given stream ("master" synchronization rules). These synchronization semantics define synchronization 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 return to the example of Fig. 1, and assume: 1) that the audio medium is the most important one, and 2) that the interstream drift must remain under 100 ms, as less than 100 ms [10] of temporal gap between audio and video cannot be perceived by a human being.

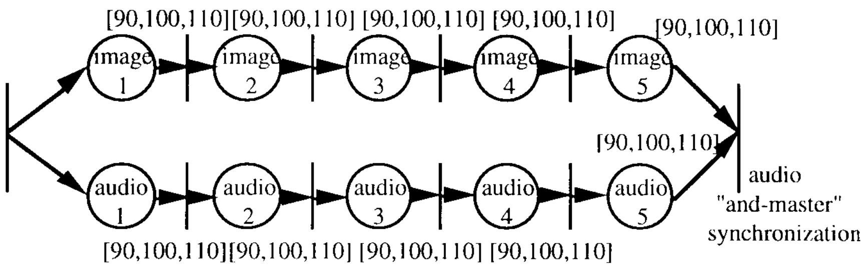

Fig. 1. Videoconference TStreamPN example.

These QoS parameters determine the complete TStreamPN of Fig. 1.

- The interstream synchronization is of "and-master": a) the sound is more important than the video (is the 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 interstream drift must be less than 100 ms; the maximum drift on five objects being 50 ms for each stream, the interstreams drift cannot be greater than 100 ms.

The TStreamPN of Fig. 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 fulfill the QoS requirements, and shows the original contribution of the paper in terms of layering, design, and implementation. In particular, it emphasizes 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 SDU's 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 LAN's and WAN's 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 TStreamPN's the different layers and sublayers. As a consequence, it makes explicit for the first time the formal temporal semantics of a global layered architecture.

III. A New Distributed Multimedia Architecture

Section II has shown that the purpose of the architecture currently under study is to implement a distributed multimedia application having well-defined synchronization requirements. It appeared that formal appropriate models as TStreamPN's 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, many studies have been performed around the design of lightweight 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 constraint management mechanisms remain implementation dependent: a new generation of high-speed, multimedia, and cooperative architectures then has 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

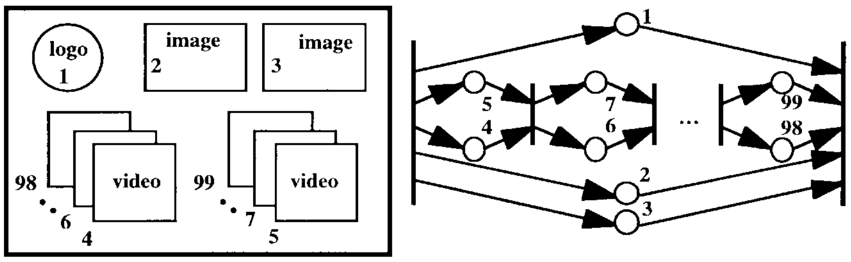

Fig. 2. Partial order deduced from a TStreamPN model of a multimedia object.

applications. This approach, of course more complex, leads to network support software that is 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 synchronization 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.

A. A Partial-Order-Based Transport Service and Protocol

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 the 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 PDU's 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 setup. 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 (SDU's) 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 II that the TStreamPN model formally describes multimedia synchronization 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 synchronization requirements. Furthermore, a Petri net represents a partial order.

Consider, for instance, the *multimedia object* pictured on the left part of Fig. 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 transitions. For objects 4–99, the logical synchronization 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 SDU's) may be delivered to the service user in any sequence consistent with both spatial and temporal synchronization requirements; these different delivery sequences lead to transfer speed-up and save resources at both the sending and receiving sides. References [27]–[30] illustrate this last point through multiple examples and two different theoretical analyses.

Reference [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

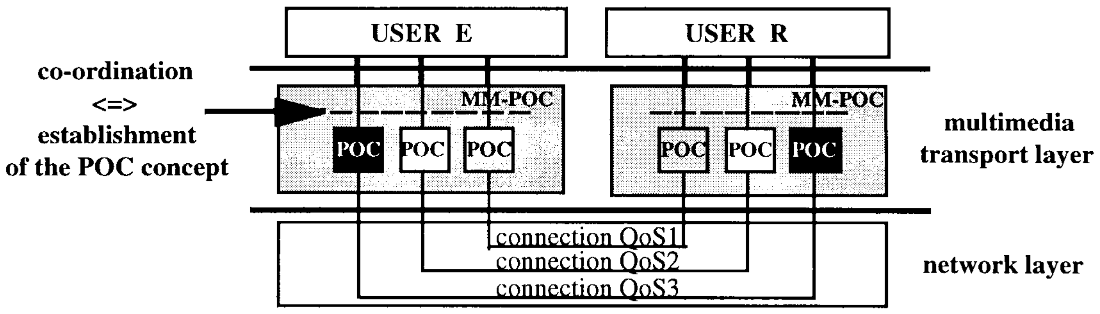

Fig. 3. Multimedia partial order transport architecture.

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. Reference [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.

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 synchronized 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 *nonreal 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 works use investigations around the QoS concept of [18].

Although pursuing the AAN philosophy, these two approaches do not tackle interflow synchronization 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 III-A2a].

Let us note that interflow 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 setup; as a result, the receiving entity has to deal with synchronization control according to a great deal 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.

a) Design principles of an MM-POC: In order to tackle the "multiflow" 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 (called MM-POC for *multimedia partial order connection*) is based on a multimedia connection concept, implying the setup and then a specific coordination, *at the transport level*, of several monomedia connections, each of them providing a specific QoS. As an example, consider the MM-POC given Fig. 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 coordination of the different POC's (illustrated in the figure by a dotted line) is based on the management at the transport level of "order" and "reliability" between the connections.

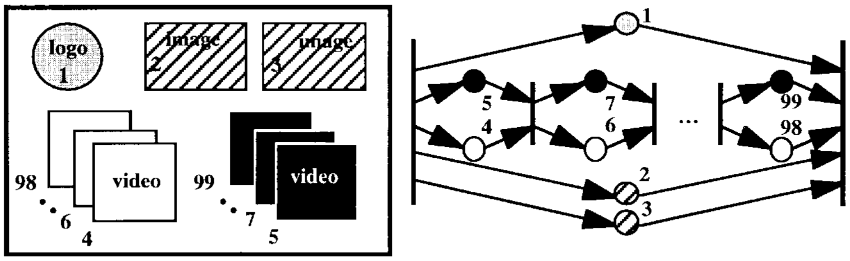

Fig. 4. Multimedia partial order.

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 interflow *logical* synchronization requirements, which can be deduced from a TStreamPN model of the application.

Consider, for instance, the partial order given on the right part of Fig. 4, deduced from a TStreamPN model of the object illustrated on the left part. If one color is used to indicate a specific QoS requirement, the associated MM-POC will be made of four monomedia POC's, each of them providing a transport support of the corresponding colored objects (SDU's). In this example:

- "SDU 6 has to be delivered after SDU 4" is an example of an intraflow 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 interflow 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 synchronization requirements expressed by the application, which 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 PDU losses when the resulting lost SDU's do not generate a *degradation* of the selected reliability (*degradation* meaning that the required reliability is out of the requested boundaries). Reference [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 all SDU's 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 SDU's 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 analyzed.

First, assume a multimedia partial order connection to be composed of n monomedia POC's, each of them being identified by an index 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 PDU's made obsolete on POC$_i$ still fulfill reliability requirements, that is, with our reliability definition assumption, if the number of consecutive SDU's made obsolete does not exceed k_i. However, it cannot deliver it if this delivery would need the loss declaration of one or more SDU's on any of the other POC's. A *media per media* reliability is thus defined.

"Per group of media" reliability management: Generalizing 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 SDU's on *any* POC, that is, still with our reliability definition, when the number of valid consecutive SDU's 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 POC's.

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 optimized 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 optimized), but independence between POC's is preserved.

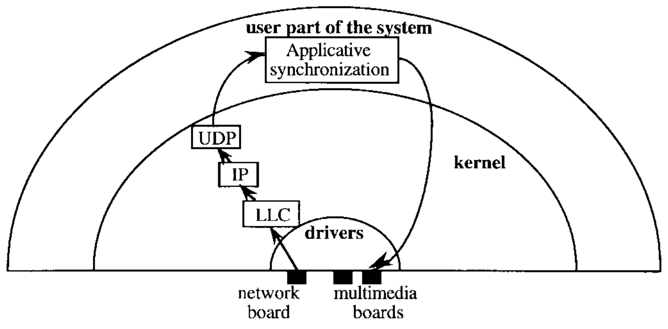

Fig. 5. Transit of multimedia data in the operating system.

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 synchronization requirements at the application layer. This point is developed in Section III-B and then illustrated through an example in Section IV.

b) Implementation of an 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].

B. 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 synchronization requirements.

1) The Temporal Synchronization Task in the Multimedia Architecture: In asynchronous distributed systems, temporal synchronization 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 synchronization operations at a lower level, for instance in the communication layer, is not useful. This is because such synchronized data will be desynchronized 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 synchronized, data are sent to the multimedia subsystems through the kernel, and are again impacted by asynchronism (Fig. 5).

In fact, it is impossible to control the behavior of data in the kernel and in multimedia subsystems: data are 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 synchronization will depend on the truth of this hypothesis.

This assumption about multimedia subsystems latency reduces temporal synchronization 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 synchronization mechanisms enforcing the temporal requirements of multimedia applications. This point will be detailed in the implementation part (Section IV-C) of the case study discussed in Section IV.

As a result of these asynchronous system characteristics, the behavior of the applicative synchronization task can be different from the presentation one modeled in Section II. Indeed, if latency times are different, the synchronization 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 interstream synchronization can change, etc. It follows that the presentation synchronization scenario modeled by a presentation TStreamPN will induce, at the application level, two different applicative synchronizations, and so two TStreamPN models. One models the synchronization 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 are missing, the normal protocol behavior consists of 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 are considered as lost, and exception handling is started, which consists of 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 interstream synchronizations 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 fulfill this requirement.

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 Fig. 6. Note that it does not directly interface the synchronization 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 presynchronization

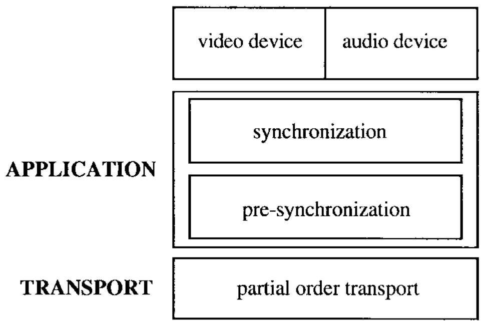

Fig. 6. Architecture of a multimedia application on top of a partial order transport.

layer is required to compensate for the temporal deficiencies of the partial order transport.

The architecture has three conceptual layers: transport, presynchonization, and synchronization.

- The partial order transport opens high-speed connections for communicating multimedia objects, and provides earliest delivery, as has been explained in Section III-A. 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 presynchronization layer has been added between the transport and the synchronization 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 synchronization 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 synchronization 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 nonsynchronizable data are 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 minimize time and ensure a maximum QoS; and 3) storage is reduced: only the presynchronization 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.

IV. Case Study: The PNSVS Videoconference Application

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

The application TStreamPN model has been given in Fig. 1 for 10 images/s, 10 ms maximum of jitter, and an interstream drift of less than 100 ms.

A. Behavior Modeling of the Synchronization Layer

In this TStreamPN, for a normal interstream synchronization, any audio object i must be synchronized 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 Fig. 1, the final presentation would not be synchronized: the audio part would be 200 ms late with respect to the video part, i.e., audio object i would be synchronized with image $i + 2$.

Latency Times: To solve this problem, an artificial drift has to be introduced in the rendezvous. The difference between the audio and the video board latency times being 200 ms, it is sufficient to synchronize the audio object i with the image $i - 2$: after having been handled by the presentation boards, the audio object i will be synchronized with the image i. In this example, the applicative TStreamPN modeling the synchronization of the application, with the drifted rendezvous, is given in Fig. 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 rendezvous 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 that the maximum interstream drift remains under 100 ms (cf. Fig. 1), the TStreamPN interstream synchronization period has to be modified. If the interstream synchronization is enforced each five images, because of the 20 ms remaining drift, the interstream drift of the audio/video streams would be in the interval [−80 ms, 120 ms], outside the allowed 100 ms. The interstream synchronization must be enforced every four images in order to keep the interstream drift in the interval [−60 ms, 100 ms].

The presentation TStreamPN modeling the multimedia synchronization scenario and the applicative TStreamPN modeling the applicative processes behavior have different shapes because of the drifted rendezvous. Moreover, the interstream synchronization 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 does not overrun a maximum bound, in order to control the end-to-end presentation delay.

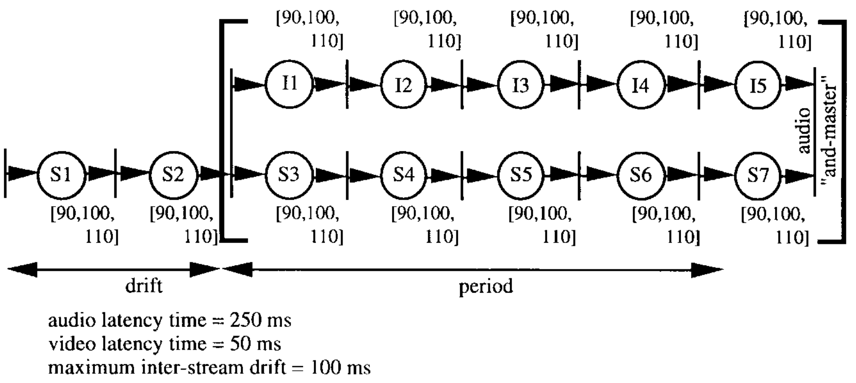
[90,100, 110]
I1
I2
I3
I4
I5
S1
S2
S3
S4
S5
S6
S7
audio "and-master"
drift
period
audio latency time = 250 ms
video latency time = 50 ms
maximum inter-stream drift = 100 ms

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

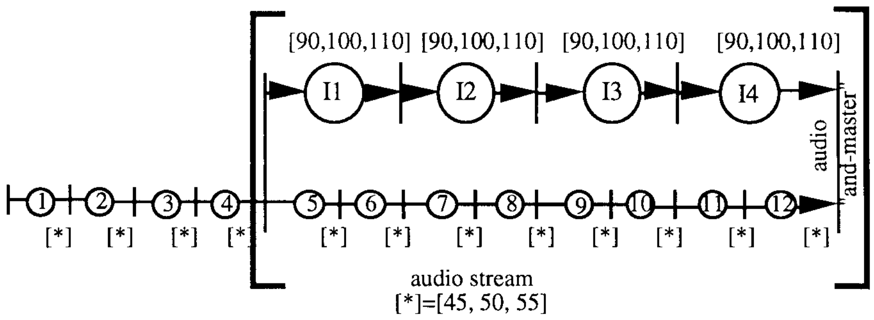

Fig. 8. Applicative TStreamPN taking into account the end-to-end delay improvements.

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 shown 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 possible, however, 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 Fig. 7 for which an interstream synchronization period modification has been applied, the 100 ms audio packets (800 bytes) require 100 ms to be produced and 2 or 3 ms to be packetized and sent. On the other hand, if the computing time of one image (grabbing, compression, packetization, and sending) requires only 55 ms, the delay comes from the audio stream. As audio packets are nonatomic, they can be divided. Dividing the 100 ms audio packet by two 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 (Fig. 8). Note that as this delay is induced by the atomic video stream, it cannot be reduced any further.

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

To define the partial order transport behavior, 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 synchronization between the two media. The reliability requirements are associated with the connections in the same way as the required throughput.

B. 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 presynchronization sublayer between the synchronization and transport layers.

Given this synchronization architecture, the modeling TStreamPN architecture has been extended in order to take into account the functionalities of these new layers.

The presynchronization sublayer has to detect late or lost data and control end-to-end delay. To model the presynchronization sublayer, a new TStreamPN is required: the presynchronization TStreamPN. This new TStreamPN has

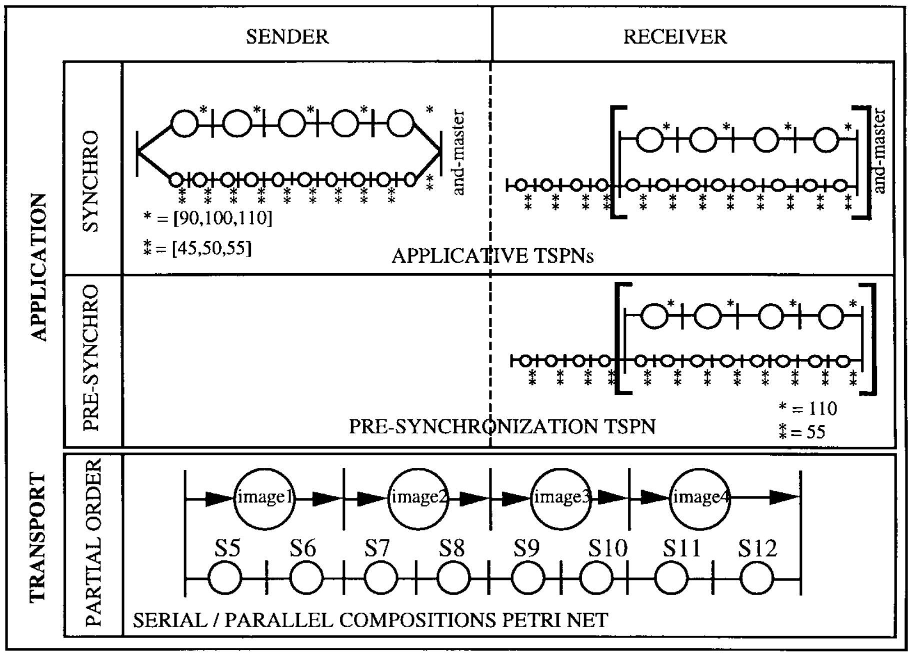

Fig. 9. TStreamPN architecture.

the same shape as the receiving applicative TStreamPN, and contains the required parameters to be able to detect the late data and control the end-to-end delay.

To model the partial order transport behavior, 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 presynchronization TStreamPN is used. Reliability constraints are associated with the connections in the same way as the required throughput.

Fig. 9 shows an entire example of the PNSVS modeling with the related TStreamPN's: applicative, presynchronization, and transport. For readability, not all parameters are given in the figure.

C. 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-Mbit/s ATM network.

The synchronization mechanisms aim to present audio and video objects while fulfilling their synchronization requirements. In fact, the goal consists of respecting the QoS requirements defined by the presentation TStreamPN, which is the same as the one given in Fig. 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 preemptive 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 an RT process runs. RT processing must therefore be kept short. For instance, if the workstation is overload by RT tasks, communications (in system class) will not be processed. Also, if an 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 synchronization requirements, but it has to be handled with care.

The architecture of PNSVS has been divided into subtasks, where the following hold.

- The audio and video stockers receive data from the network. Buffers are used to temporarily store the desynchronized received data to solve the jitter problem.
- The presentation audio and video processes realize the required operations for the sound and picture presentations.
- The real-time orchestration processes realize the synchronization scenario modeled by the TStreamPN, and control the presentation processes to respect the presentation temporal requirements.
- The presynchronization 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, presynchronization processes realize a temporal control of the stocker, while the orchestration processes do the same on the presentation ones.

Intrastream Synchronization: The proposed synchronization approach gives a solution different from 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

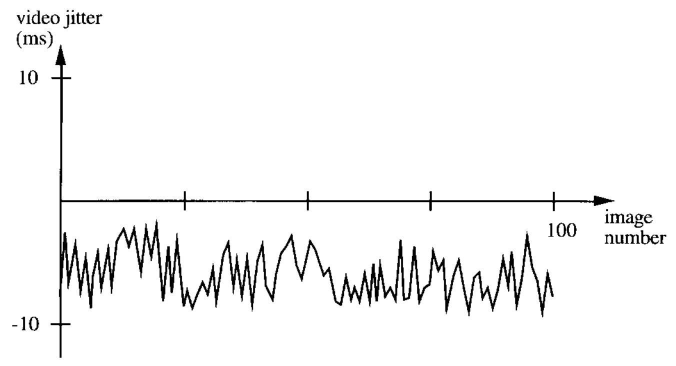

Fig. 10. Measurement of the jitter on the PNSVS images.

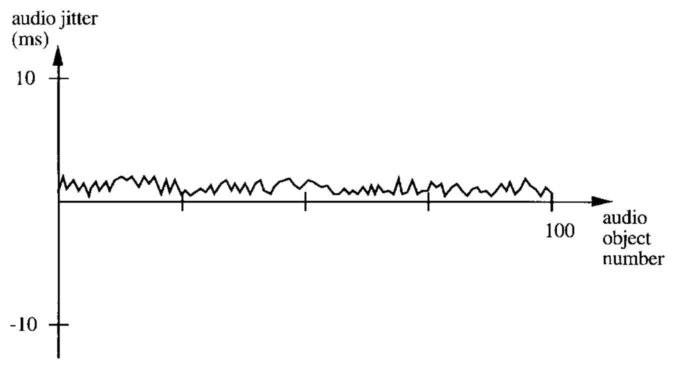

Fig. 11. Measurement of the jitter on the PNSVS audio objects.

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.

Interstream Synchronization: The interstream synchronization algorithm is based on a rendezvous 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 intrastream synchronization, 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 rendezvous of the video orchestration process (meaning that the video presentation process is completed): then, it can fire the interstream 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 synchronization period.

D. Performance Measurements

1) Synchronization Mechanisms Evaluation: PNSVS is a videoconferencing application that can process 20 images/s (320 $\times$ 240 pixels and 24-bit 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 synchronization mechanisms respect the temporal presentation requirements, and in particular, the quality of the intra- and interstreams synchronization. The measurements have been realized in the case of a 10 image/s videoconference application where synchronization requirements are modeled by the presentation TStreamPN of Fig. 1. Fig. 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 overstepped, but the jitter is always negative. In fact, in this experiment, there is no network problem (no loss and jitter), and data are available when the presentation process needs them. Thus, the anticipation mechanism, which makes it possible to stop the presentation

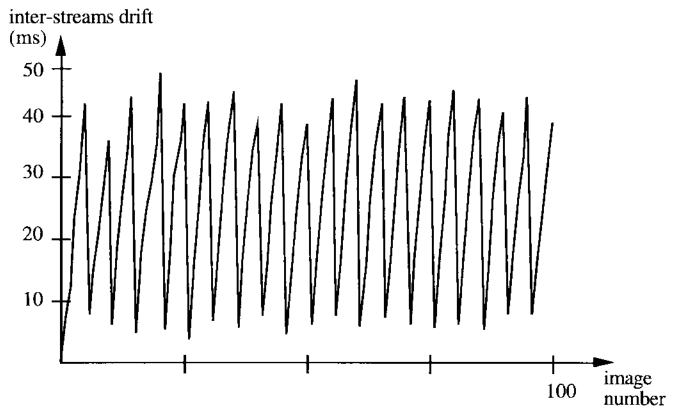

Fig. 12. Measurement of the PNSVS interstream drift.

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.

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

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

2) Overall Architecture Benefits Evaluation: The temporal synchronization mechanisms having been measured, it remains to evaluate the benefits of the overall architecture that includes the partial order transport, the presynchronization, and the temporal synchronization layers. As previously stated, the use

% 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

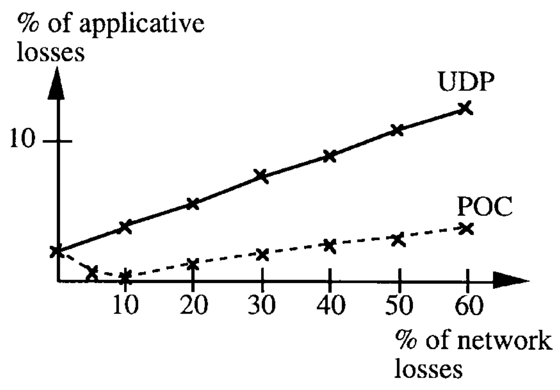

Fig. 13. Overall architecture benefits evaluation.

of a partial order transport (thanks to its earliest deliveries and losses mechanisms) avoids time losses at the application level. These time losses are really harmful to QoS because the application has to discard some presentation objects (images or sounds) when it does not have enough time to compute them (end-to-end delay control mechanisms, "and-master" interstream synchronization schemes, etc.). Thus, in this section, the losses created by the synchronization application will be measured in two cases: 1) when a partial order transport is used, and 2) when a classical connectionless transport service (UDP) is used. To perform this evaluation, a network simulator allowing us 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 with the temporal constraints expressed by the TSPN of Fig. 1. The other parameters are: maximum end-to-end delay 1 s, audio latency 400 ms, and video latency 50 ms. With these values, the audio and video buffers can contain, respectively, five audio frames and ten images. Results of measurements are described in Fig. 13. It shows the applicative loss average in the two cases (using UDP and POC), depending on the network loss level.

Fig. 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 it fires the TSPN transition as soon as possible (when the minimum presentation time is reached). While using UDP, it has to wait for the expiration of the timer associated with 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 phenomenon, 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 synchronization period is positive (there are 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 synchronization 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 five objects consecutively lost, five 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 presynchronization layer detects them. The principle of the presynchronization mechanism is the same as the applicative synchronization one. Presynchronization is, nevertheless, more effective because it computes data earlier than the synchronization layer. However, the drift on a synchronization 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.

V. Conclusion

This paper has presented adequate architectures and mechanisms to fulfill important synchronization requirements for multimedia applications in asynchronous environments. To obtain the best possible QoS, the synchronization architecture consists of two extreme layers: an applicative synchronization 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 presynchronization 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 behavior of all layers of such a synchronization 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 synchronization 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 synchronization application behavior. 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-Mbit/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 fulfills the synchronization requirements.

Future work concerns the theoretical and practical extension of PNSVS toward a multiuser videoconference application allowing many users to communicate with a guaranteed QoS.

ACKNOWLEDGMENT

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

REFERENCES

[1] G. Coulson, G. S. Blair, and P. Robin, “Micro-kernel support for continuous media in distributed systems,” *Comput. Networks ISDN Syst.*, vol. 26, 1994.
[2] K. Jeffay, D. L. Stone, and F. D. Smith, “Kernel support for live digital audio and video,” *Comput. Commun.*, vol. 15, July/Aug. 1992.
[3] K. G. Shin and P. Ramanathan, “Real time computing: A new discipline of computer science engineering,” *Proc. IEEE*, vol. 82, Jan. 1990.
[4] T. D. C. Little and A. Ghafoor, “Synchronization and storage models for multimedia objects,” *IEEE J. Select. Areas Commun.*, vol. 18, Apr. 1990.
[5] C. C. Yang and J. H. Huang, “A multimedia synchronization model and its implementation in transport protocol,” *IEEE J. Select. Areas Commun.*, vol. 14, Jan. 1996.
[6] M. Woo, N. U. Qazi, and A. Ghafoor, “A synchronization framework for communication of pre-orchestrated multimedia information,” *IEEE Networks*, pp. 52–61, Jan./Feb. 1984.
[7] B. Berthomieu and M. Diaz, “Modeling and verification of time dependent systems using time Petri nets,” *IEEE Trans. Software Eng.*, vol. 17, Mar. 1991.
[8] M. Diaz and P. Sénac, “Time stream Petri nets, A model for multimedia streams synchronization,” in *Proc. Multimedia Modeling'93*, Singapore, Nov. 1993.
[9] P. Sénac, M. Diaz, and P. De Saqui-Sannes, “Toward a formal specification of multimedia synchronization,” *Ann. Telecommun.*, May/June 1994.
[10] K. Jeffay, D. L. Stone, and F. D. Smith, “Transport and display mechanisms for multimedia conferencing across packet-switched networks,” *Comput. Networks ISDN Syst.*, vol. 26, 1994.
[11] D. D. Clark, M. L. Lambert, and L. Zhang, “NETBLT: A bulk data transfer protocol,” RFC 998, Mar. 1987.
[12] D. R. Cheriton and C. L. Williamson, “VMTP as the transport layer for high-performance distributed systems,” *IEEE Commun. Mag.*, pp. 37–44, June 1989.
[13] G. Chesson, “XTP/PE design considerations,” presented at the IFIP International Workshop Protocols for High-Speed Networks, Rüschlikon, Switzerland, May 1989.
[14] XTP protocol definition, revision 3.6, Protocol Engines Inc., Jan. 1992.
[15] D. D. Clark and D. L. Tennenhouse, “Architectural considerations for a new generation of protocols,” in *Proc. ACM SIGCOMM*, 1990.
[16] C. Diot, C. Huitema, and T. Turletti, “Network conscious applications,” presented at the HPCS Workshop, Mystic, CT, Aug. 1995.
[17] C. Diot, R. De Simone, and C. Huitema, “Automated design of communication protocols using ESTEREL,” *J. High Speed Networks*, vol. 5, no. 2, 1996.
[18] A. Danthine, Ed., “The OSI95 transport sevice with multimedia support,” *Res. Rep. ESPRIT*, Project 5341, OSI95, vol. 1. Berlin, Germany: 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 and J. Sandvoss, “The BERKOM-II multimedia transport system,” Aug. 1993.
[21] TENET Group, “Recent and current research,” Univ. California, Berkeley, Apr. 1994.
[22] A. Campbell, G. Coulson, and D. Hutchinson, “A quality of service architecture,” *ACM Comput. Commun. Rev.*, Apr. 1994.
[23] K. Nahrstedt and J. Smith. “Design, implementation and experiences of the OMEGA end-point architecture,” *IEEE J. Select. Areas Commun.*, vol. 14, Sept. 1996.
[24] M. Zitterbart, B. Stiller, and A. N. Tantawy, “A model for flexible high-performance communication subsystems,” *IEEE J. Select. Areas Commun.*, vol. 11, May 1993.
[25] G. Gopalakrishna and G. Parulkar, “A framework for QoS guarantees for multimedia applications within an endsystem,” GI Jahrestagung, Zurich, Switzerland, Sept. 1995.
[26] M. Diaz and G. Pays, “The cesame project: Formal design of high speed multimedia cooperative systems,” *Ann. Telecommun.*, vol. 49, May/June 1994.
[27] P. D. Amer, C. Chassot, T. Connolly, and M. Diaz, “Partial order transport service for multimedia applications: Reliable service,” in *Proc. 2nd High Performance Distributed Computing Conf.*, July 1993.
[28] ——, “Partial order transport service for multimedia applications: Unreliable service,” in *Proc. 3rd Int. Networking Conf., INET'93*, Aug. 1993.
[29] P. D. Amer, C. Chassot, T. Connolly, P. Conrad, and M. Diaz, “Partial order transport service for multimedia and other applications,” *IEEE/ACM Trans. Networking*, vol. 2, Oct. 1994.
[30] M. Diaz, A. Lozes, C. Chassot, and P. D. Amer, “Partial order connections, A new concept for high speed and multimedia services and protocols,” *Ann. Telecommun.*, vol. 49, May/June 1994.
[31] C. Chassot, M. Diaz, and A. Lozes, “From the partial order concept to partial order multimedia connections,” *J. High Speed Networks*, vol. 5, no. 2, 1996.
[32] P. Conrad, E. Golden, P. Amer, and R. Marasli, “A multimedia document retrieval system using partially-ordered/partially-reliable transport service,” in *Proc. Multimedia Computing Networking 96*, San Jose, CA, Jan. 1996.
[33] R. Marasli, P. D. Amer, and P. Conrad, “Retransmission-based partially reliable transport service: An analytic model,” in *Proc. IEEE INFOCOM 96*, San Fransico, CA, Mar. 1996.
[34] M. Diaz, K. Drira, A. Lozes, and C. Chassot, “Definition and representation of the quality of service for multimedia systems,” presented at the *6th Int. Conf. High Speed Networking*, HPN'95, Palma de Mallorca (Balearic Islands), Spain, Sept. 1995.
[35] M. Fournier, C. Chassot, A. Lozes, and M. Diaz, “Multimedia partial order transport architecture: Design and implementation,” presented at the Protocols for High Speed Network'96, Sophia Antipolis, France, Nov. 1996.

[36] P. Owezarski and M. Diaz, “Models for enforcing multimedia synchronization in videoconference applications,” in *Proc. 3rd MultiMedia Modeling Conf., Toward the Inform. Superhighway (MMM’96)*. Toulouse, France: World Scientific, Nov. 1996, pp. 85–100.

Philippe Owezarski received the Ph.D. degree in computer science from the Université Paul Sabatier, Toulouse, France.

He is now working as a Researcher (“chargé de recherche”) at Laboratoire d’Analyze et d’Architecture des Systèmes (LAAS) of the CNRS (the French center for scientific research), in the research group Software and Tools for Communications. His main interest concerns high-speed multimedia and cooperative systems, and in particular, the modeling, design, and implementation of distributed applications—including multimedia synchronization and cooperation mechanisms, like videoconferences or application-sharing tools—on top of high-speed networks such as ATM. He manages, for LAAS, the CANET European project that aims to assess new information technologies (ATM networks, CAD application-sharing tools, videoconferences, and other CSCW applications) for automotive teledesign between a car manufacturer (Renault) and its supplier (Siemens Automotive), and also the MIRIHADE project that is the first French public experiment of ATM WAN.

Michel Diaz is Directeur de Recherche at the CNRS, and leads the Research Group Communications Software at the LAAS—CNRS, Toulouse. In 1989–1990, he spent one year as a Visiting Professor and staff at the University of Delaware, Newark, and the University of California, Berkeley. He has been working on the development of methodologies, architectures, and tools for the design of high-speed multimedia cooperative distributed systems.

He is now the Co-head of the French AIRBUS-CNRS project TOPASE on distributed multimedia professional teleteaching. He has written one book, more than 120 technical publications, and is the Editor of six volumes on protocol specification, testing, and verification, on the application and theory of Petri nets, and on multimedia modeling. He is also presently Director of the French Research on Parallelism, Networks and Systems (GDR Parallélisme, Réseaux et Systèmes), and a member of the CEC Advisory Group on the future of the Internet.

Dr. Diaz was/is a member of many Program Committees, and served as the Chairman for the IFIP Conference on Protocol Specification, Testing and Verification, the European Workshop on Application and Theory of Petri Nets, the International Conference on Distributed Computing Systems, the IFIP Conference on Formal Description Techniques, and the International Conference on Multimedia Modeling. He was the Prime Manager of the SEDOS-ESPRIT project, which developed the Formal Techniques Estelle and LOTOS. He received the Silver Core of the IFIP, is a member of the New York Academy of Sciences, and is listed in *Who’s Who in Science and Engineering.*

Christophe Chassot received the Dipléme d’Ingénieur and DEA in computer science from ENSEEIHT, Toulouse, in 1992 and the Ph.D. degree in computer science in 1995 from the Institut National Polytechnique of Toulouse (INPT).

He is a Maître de Conference at the Institut National des Sciences Appliquées (INSA), Toulouse, France. He also works in the Communications Software Research Group at the Laboratoire d’Analyze et d’Architecture des Systemes (LAAS) of France’s Centre National de la Recherche Scientifique (CNRS). During his Ph.D., his main interests involved multimedia transport services/protocols and formal specification using Estelle. Currently, he is working on the specification of an end-to-end communication architecture, more specifically dedicated to the transport of distributed interactive simulation (DIS) applications over new-generation networks such as ATM and IPnG.