

A General-purpose Multimedia Synchronization Mechanism based on Causal Relations

Jean-Pierre Courtiat, Luiz Fernando Rust da Costa Carmo, and Roberto Cruz de Oliveira

Abstract— Designing distributed multimedia applications raises temporal and spatial synchronization issues related to processing, transport, storage, retrieval and presentation of data, sound, still images and video. Within this framework, the paper aims to define a general-purpose multimedia synchronization mechanism, known as the conditional delivery mechanism capable of addressing both intra- and inter-stream synchronization issues. The proposed mechanism, based on the identification of causal relations among information units of one or several streams, is designed to ensure that these causal relations, expressed at the user's level, are satisfied when delivering the streams. The conditional delivery mechanism is analyzed in depth and both informal and formal specifications of the mechanism are provided. The formal specification refers to an extension of the standard formal description technique LOTOS (RT-LOTOS for Real-Time LOTOS). Validation results of the conditional delivery mechanism are finally presented for a distance and interactive training application.

I. INTRODUCTION

MULTIMEDIA synchronization is the task responsible for the co-ordination, scheduling and presentation of multimedia objects in time and space [1]. This definition poses the problem of synchronization which raises two main issues with respect to temporal synchronization [2], [3] (spatial composition of multimedia objects is not addressed here). These issues are:

- how simple temporal dependencies can be guaranteed when delivering a particular media; this is commonly called *intra-stream synchronization*;
- how structural temporal dependencies among different media can be guaranteed such that temporal links specified by the users are effectively satisfied when presenting, in a co-ordinate manner, these media at one or several remote sites; this is usually called *inter-stream synchronization*.

Numerous papers in the literature have dealt with intra-stream synchronization (see [4], [5], [6], [7] for details). However, as pointed out in [8], inter-stream synchronization is much less mature than intra-stream synchronization. In [9] mechanisms and algorithms have been devised for synchronizing streams during file storage (the file server creates a relative time system) and retrieval (the file server detects and restores synchrony by deleting or duplicating information units) on a multimedia network. In [10] a distributed synchronization algorithm capable of scheduling independent sources for a multimedia teleorchestration has been proposed.

In this paper, it is shown that the expression of causal

relations among information units from one or several streams associated with the definition of implicit intra-stream temporal requirements at the level of each individual stream, may allow complex intra- and/or inter-stream synchronization patterns to be created; these synchronization patterns may then be effectively implemented by a new general-purpose synchronization mechanism, known as the *conditional delivery mechanism*. Additionally the paper shows the advantage of using formal methods, in particular RT-LOTOS [11], [12], for specifying and validating this mechanism and assessing its effectiveness for implementing the synchronization requirements of a distance and interactive training application.

The paper is organized as follows: Section II gives the main intuitive background to the conditional delivery mechanism as well as several examples of application for intra- and inter-stream synchronization. Section III details an application in the area of distance and interactive training illustrating the use of the conditional delivery mechanism and, more particularly, the merging of temporal and causal requirements. Section IV describes the implementation of the conditional delivery mechanism by looking at the so-called *restrictor algorithms*. Section V presents the formal background supporting the validation of the formal specification of the conditional delivery mechanism and introduces some simulation results obtained by applying this mechanism to the interactive training application. Finally, some conclusions are drawn and future research work is outlined in Section VI. For clarity, LOTOS and its temporal extension RT-LOTOS have been briefly described in appendix.

II. INTUITIVE BACKGROUND TO THE CONDITIONAL DELIVERY MECHANISM

A. Introduction

By considering the basic synchronization concepts reported in [13], multimedia information can be modeled as streams made up of a timed sequence of information units, a bundle being a collection of streams which have been grouped in the same temporal range. Both streams and bundles exhibit temporal properties that may be formalized by their temporal signatures i.e. the respective time stamps of the stream information units.

Conditional dependencies have been proposed [14], [15] as a way of taking advantage of the knowledge on semantic relationships among different stream/bundle information units in order to characterize intra- and inter-stream synchronization patterns. It is assumed that the conditional delivery mechanism, designed to enforce these syn-

The authors are with the Laboratoire d'Analyse et d'Architecture des Systèmes, Centre National de la Recherche Scientifique, 7, avenue du Colonel Roche, 31077 Toulouse Cedex, France.

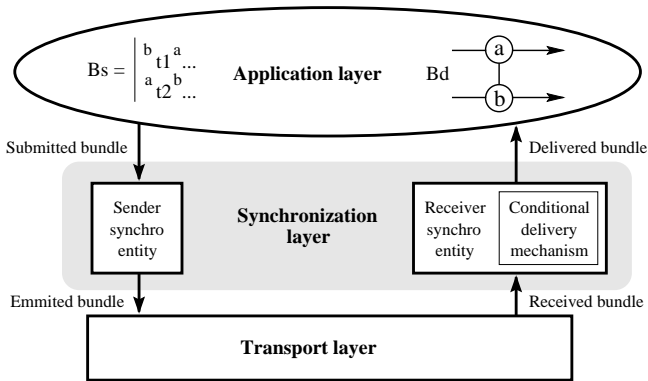


Fig. 1. Functional architecture

ynchronization patterns, can be implemented on top of a transport service (see Fig. 1) providing a basic connection-oriented service with a guaranteed bandwidth, a bounded packet loss and possibly jitter compensation mechanisms for isochronous streams [5]. Dependency expressions will be associated with information units whenever they are submitted to the source synchronization entity; they will then be encoded within the information units transferred across the transport service for a subsequent recovery at the remote peer synchronization entity. There, the dependency expressions will be evaluated by the conditional delivery mechanism so as to determine whether the referenced information units are to be delivered (within the delivered bundle) to the upper layer user.

B. Expressing conditional dependencies

Conditional dependencies are causal relations associated with a stream's information unit aiming to express the delivery constraints of that information unit, relative to the delivery of other information units belonging to either the same stream (intra-stream conditional dependencies) or distinct ones within the same bundle (inter-stream conditional dependencies).

Causal relations may be characterized by Boolean expressions (termed *dependency expressions*). For example, dependency expression $Expr^m$, associated with the delivery of information unit m , may formally be defined as a logical expression in a disjunctive normal form on a (finite) number of Boolean variables. The latter known as information unit identifiers, characterize the delivery status of the information units on which the delivery of m depends. Depending on whether these are prefixed by a *not* logical operator (denoted \neg), they characterize the positive or negative premises defined within a particular conjunction of the dependency expression.

As a simple example of intra-stream conditional dependencies, consider the following stream definition: $S = t_1^{n_1}, t_2^{n_2}, n_2 t_3^{n_3}, \dots$ where t_1, t_2, t_3, \dots are those instants when information units n_1, n_2, n_3, \dots have been submitted. Note that $Expr^{n_1} = Expr^{n_2} = true$ and $Expr^{n_3} = n_2$, thus implying, on the one hand, that no conditional delivery constraint has been defined for information units n_1

and n_2 , and on the other, that delivery of n_3 depends on that of n_2 . Thus, if n_2 cannot be delivered, then n_3 should not be delivered. Note also, that the instant when n_3 may be delivered equally depends not only on meeting the conditional delivery constraints but on the global timing associated with the stream.

Inter-stream conditional dependencies entail a greater level of complexity, as the delivery of an information unit a may depend on that of information unit b , whose delivery may depend itself on that of a . This characterizes the setting of a *synchronization point* among different streams of a bundle, which should normally lead to the simultaneous delivery of information units a and b . However, if for any reason, one information unit cannot be delivered (for instance, because its delivery time fails to match the associated temporal requirement), then none of the information units would be delivered.

As a simple example of inter-stream conditional dependencies, consider the following bundle definition comprising two streams:

$$B = \begin{bmatrix} t_1^{n_1}, t_3^{n_2}, m_2 t_5^{n_3}, \dots \\ t_2^{m_1}, n_3 t_4^{m_2}, \dots \end{bmatrix}$$

where t_1, t_3, t_5, \dots and t_2, t_4, \dots characterize respectively the instants when the information units of both streams have been submitted. Here, the delivery of information unit n_3 depends on the delivery of m_2 and vice-versa. This coupled inter-stream conditional dependency relation between n_3 and m_2 characterizes a synchronization point, which implies that n_3 and m_2 are to be delivered at the same time. In other words, they are to be mutually synchronized when delivered, although they may have been submitted at a different time.

As a further example of inter-stream conditional dependencies, consider the delivery configurations shown in Fig. 2:

- Configuration $\{a, b, c\}$, where $Expr^a = b$ (i.e., a depends on b), $Expr^b = c$ and $Expr^c = a$, represents a single synchronization point;
- Configurations $\{d, e\}$ and $\{e, f\}$, where $Expr^d = e$, $Expr^e = d \wedge f$ (i.e., e depends on d and f) and $Expr^f = e$, constitute together one single synchronization point obtained by merging synchronization points $\{d, e\}$ and $\{e, f\}$ (see operator \wedge in $Expr^e$);
- Configurations $\{g, h, i, j\}$ and $\{i, k\}$, where $Expr^g = h$, $Expr^h = i$, $Expr^i = k \vee j$, $Expr^j = g$ and $Expr^k = i$, make up two distinct synchronization points that cannot be merged (see operator \vee in $Expr^i$).

C. Merging dependency and temporal requirements

The effective delivery of a stream's information units is dependent upon the conditional delivery constraints being met as discussed above, as well as additional time constraints related to the nature of the considered stream. Let $\Delta m = [td_{min}^m, td_{max}^m]$ be the so-called *delivery time interval* associated with some information unit m ; this interval defines the maximum time ($\delta d = td_{max}^m - td_{min}^m$) during

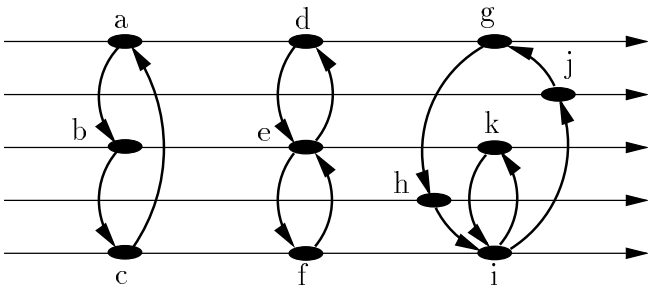


Fig. 2. Different synchronization point configurations

which m can be stored within the receiver entity before delivery. Basically, δd may be defined in two ways, depending on the kind of temporal requirement associated with its related stream:

1. if no jitter compensation mechanism is applied to the stream, then δd can be defined as a specific QoS parameter of the conditional delivery mechanism (delivery_time_interval QoS parameter);
2. otherwise, δd may be derived from the residual jitter allowed (if any) by the jitter compensation mechanism.

The main difference between the two assumptions lies in the time when the delivery time interval Δm starts. According to assumption 1), Δm starts at time tr^m when m becomes available from the transport service, i.e. $\Delta m = [tr^m, tr^m + \delta d]$. According to assumption 2), and to take account of the additional temporal requirement, three cases may occur, viz.:

1. if $tr^m < td_{min}^m$, then $\Delta m = [td_{min}^m, td_{max}^m]$;
2. if $td_{min}^m \leq tr^m \leq td_{max}^m$, then $\Delta m = [tr^m, td_{max}^m]$;
3. if $tr^m > td_{max}^m$, then m cannot be delivered due to a late reception, and must therefore be discarded. As a consequence there is no delivery time interval associated with this situation.

D. Brief summary

The general-purpose synchronization mechanism proposed here follows a hybrid approach, relying on both causal relations (essentially at the level of inter-stream dependency relations) and timing constraints (at the level of the intra-stream synchronization of each medium taken separately). The notion of delivery time interval plays a central role for merging these different kinds of requirements.

This approach is fairly different from previous approaches reported in the literature, and solely based on global timing requirements [16]. It is the authors' belief that many inter-stream synchronization requirements, normally expressed as global temporal requirements, may be translated into inter-stream dependency relations, the global timing required for the presentation of the multimedia document being then ensured by a particular "master" stream with which the other streams may synchronize through use of synchronization points [17]. In this light synchronization points appear as powerful synchro-

nization tools, generalizing the marker concept initially introduced in [3]. Note that this hybrid approach is particularly well-suited for the remote presentation of multimedia documents with distinct sources, a synchronization issue being recognized as particularly challenging [10]. Finally, causal relations can also be applied for implementing intra-stream synchronization patterns, with a possible direct application to a MPEG coding stream, where one wants to recover from the possible loss of Intra-coded pictures.

III. ILLUSTRATIVE APPLICATION

A. Presentation of the application

In this section, a simple example in the area of distance and interactive training is described. The application is assumed to be distributed over three nodes, namely a synchronous server (SS), an asynchronous server (AS) and a student's workstation (S) (see Fig. 3). SS provides the student with audio information, whereas AS provides him with text and slide information. The transfer of these multimedia information may be characterized by a type II bundle transfer [13] (from distinct sources - SS and AS -, to a single destination - S).

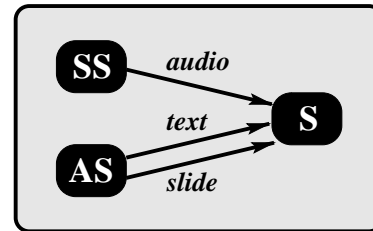


Fig. 3. Application nodes

The part of the training application considered here comprises the following three phases : an introduction (the servers send various pieces of information to the student in order to introduce a particular topic), a question (the servers send various information to the student so as to ask him a question), and finally answer assessment (the servers send various information to the student in order to comment his answer). Fig. 4 shows the high-level temporal scenario related to the delivered bundle, including the presentation duration of each information unit (slide, audio or text) sent by the servers. Finally, it is assumed that the student answers the asked question by means of a control connection between him and SS.

Consider the synchronization requirements that may be associated with the different phases of the application. In the introduction, several information units are transferred from the servers to the student, i.e.: one slide information unit, four audio information units (each audio information unit corresponding to some encoded audio segment) and two text information units. In this phase, there exists a strong requirement for the delivery of the audio and slide information units. In others words, the delivery of only one type of information (audio or slide) is regarded as useless from an application point of view. Such a strong require-

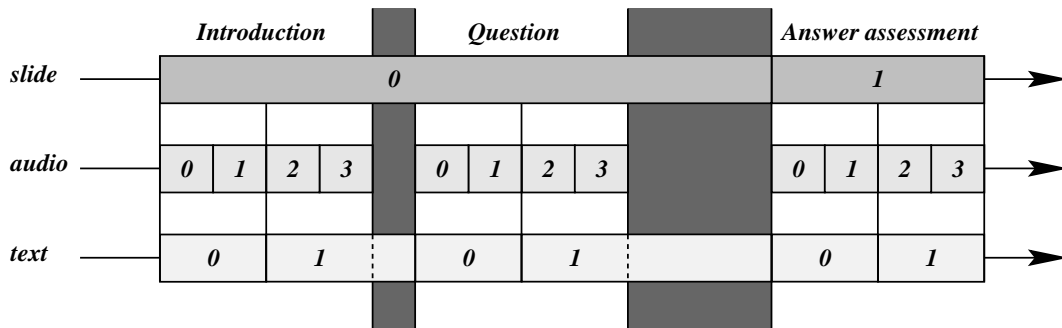


Fig. 4. Temporal presentation requirements

ment no longer holds for the text information units, as it is assumed that this application phase can proceed, even if some text information units cannot be presented on time to the student. The question phase includes the continuous presentation of the previous slide, as well as that of four new audio information units and two new text information units. In this phase, there exists a strong requirement for delivery of the text information units, as these describe the various options available to the student for answering the question asked through use of audio information units. Finally, the question assessment phase includes the transfer of one slide information unit, four audio information units and one additional text information unit. The information transferred during this phase depends on whether the student's previous answer is right. In this phase, delivery of audio information is not considered mandatory, since it is assumed that the student will be sufficiently aware of the validity of his answer thanks to the other information units delivered to him (text and/or slide information units).

The three phases of the application are mandatory; consequently, if one phase is not successful (e.g., due to mandatory information units not presented at the right time), then it will abort and the subsequent phase(s) (if any) will no longer occur.

B. Expressing causal requirements

It is worth analyzing the causal relations that may be expressed for this application, starting from the relationship existing between the first audio information unit (ai_0) and the first text information unit (ti_0) of the introduction phase. As previously stated, if, (due to a possible loss of information units in the transport service), ai_0 cannot be delivered, then the application cannot start; ti_0 may then be delivered if and only if ai_0 has previously been delivered. In other words, from the application viewpoint, the delivery of ti_0 is useless if, for some reason, ai_0 cannot be delivered. The following dependency expression specifies the conditional dependency requirement associated with the delivery of ti_0 : $Dep_{pr}^{ti_0} = ai_0$.

Let us now analyze the relationship between the first audio information unit (ai_0) and the slide information unit (si_0) of the introduction phase. It has been stated that the application cannot start if either information unit cannot be delivered in time. In other words, the delivery of ai_0 de-

pends on the delivery of si_0 and vice-versa. Expressing such a coupled dependency relation characterizes the setting of a synchronization point which should result in ai_0 and si_0 being delivered simultaneously.

Another interesting feature of causal relations is the possibility of submitting multiple information units, and then delivering only one of them (or a subset) by evaluating some pre-determined dependency expressions. In the proposed training application, this may be useful during the answer assessment phase so as to avoid a specific interaction between the workstation on which the student is logged and the asynchronous server. Proceeding thus, the student's answer is only sent to SS, which assesses the correctness of the answer by issuing positive or negative audio comments. Slide and text information units corresponding to both cases (the student's answer is right or wrong) may be independently sent by AS, the delivery of these information units depending on the corresponding, positive or negative, audio comment sent by SS. Depending on which information unit, originating from SS, is received from the transport service, only the synchronization point associated with the current situation (the student is right or wrong) will be enabled, permitting thereby to deliver to the student only the relevant information units from both SS and AS.

Causal relations among information units may be expressed graphically as follows: an arrow from a to b means that the delivery of a depends on that of b . A circuit in this graph characterizes the presence of a synchronization point among the respective information units. This leads to the *dependency graph* depicted in Fig. 5, which characterizes the causal relations identified in the training application.

With respect to the graph, it may be pointed out that the causal relations between two consecutive audio information units from the introduction and question phases, mean that these information units cannot be lost. On the other hand, in the answer assessment phase, information units aw_1, aw_2, aw_3 only depend on aw_0 , this dependency being required for starting the answer assessment phase. Finally note, that the setting of synchronization points $\{aq_0, tq_0\}$ and $\{aq_1, tq_1\}$ enforces the simultaneous presentation of the text and audio information units during the question phase.

The causal relations formalized so far have expressed the

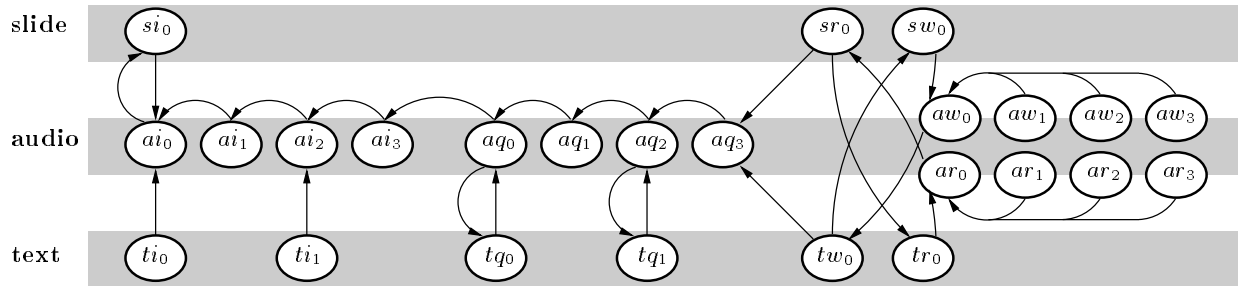


Fig. 5. Dependency graph of the training application

dependency constraints that have to be fulfilled when delivering the bundle information units so as to meet the application requirements.

C. Expressing temporal requirements

Consider the specific temporal requirements that may be expressed for the training application. To do this, the audio stream is first analyzed because of its isochronous nature. The presentation of such a stream motivates the use of a temporal signature preserving mechanism in order to compensate for the jitter that may be introduced by the lower transport layer [5]. The delivery time interval, to be associated with any information unit of the audio stream, accounts for some residual jitter which itself depends on the audio quality expected by the users.

As far as the other media (text and slides) are concerned, no jitter compensation mechanism is needed. The causal relations suffice to match the temporal requirements stated for the application, as soon as the following two conditions are met:

- the text and slide information units should be received by the remote synchronization entity before the end of the delivery time interval associated with the relevant audio information unit, whenever there exists a synchronization point involving an audio information unit and a text/slide information unit;
- the value of δd (for delivery of the text/slide information units) must be large enough to avoid discarding the text and slide information units before their predicted time of delivery.

The first condition allows the synchronization points to be enabled (provided of course no information unit has been lost in the transport service). The second condition defines a minimum value for the upper bound of the delivery time interval associated with the text and slide information units.

IV. RESTRICTER ALGORITHMS

A. Introduction

Restrictor algorithms have been devised for implementing the conditional delivery mechanism. The *restrictor* is an object manipulated by these algorithms expressing the relevant information for delivering the information units. Two restrictor algorithms, resp. *simple restrictor algorithm*

and *general restrictor algorithm*, have been developed. The former was initially defined for the implementation of the intra-stream conditional delivery mechanism [18]. The latter is a generalization of the previous one accounting for the coupled dependency expressions among information units belonging to distinct streams. This generalization, the inter-stream conditional delivery mechanism [14], [15], is required for setting synchronization points among distinct streams of a bundle.

B. The simple restrictor algorithm

For clarity, the simple restrictor algorithm is briefly introduced to identify what has then to be added in the general restrictor algorithm to cope with coupled conditional dependencies. Basically, this algorithm works as follows: for every information unit m , received from the transport service, a restrictor is created and m is temporarily stored in a buffer; a restrictor is (initially) defined as a tuple $(m, \Delta m, Dexpr^m)$, where Δm is the delivery time interval of m and $Dexpr^m$ is the dependency expression associated with m ; the dependency expression is evaluated when the restrictor is created and re-evaluated whenever another information unit is delivered; if both temporal and dependency conditions are satisfied, the information unit is delivered, otherwise it is not.

By way of example, consider the following two restrictors $(m, \Delta m, Dexpr^m)$ and $(n, \Delta n, Dexpr^n)$, such that $Dexpr^m$ and $Dexpr^n$ are coupled dependency expressions defined as $Dexpr^m = n$ (m depends on n) and $Dexpr^n = m$ (n depends on m). The simple restrictor algorithm states that $Dexpr^m$ and $Dexpr^n$ are re-evaluated whenever an information unit is delivered. Thus, using the simple restrictor algorithm, neither n nor m would be delivered because there is no other delivery of information unit which could induce $Dexpr^m = true$ or $Dexpr^n = true$. This issue, which is more complex when synchronization points have to be set among more than two information units, is addressed by the general restrictor algorithm.

C. The general restrictor algorithm

When creating a new restrictor, the general restrictor algorithm looks at the current set of restrictors, so as to determine whether one of the following cases arises:

1. the dependency expression of a restrictor depends on information units whose delivery is already con-

ditioned by other restricters;

2. the dependency expression of a restricter is coupled with that of other restricters.

The following example illustrates both situations and shows the type of restricter transformation that can be applied to enable the delivery of all the information units involved in a synchronization point. Let information units $\{a, b, c\}$, with their associated dependency expressions defined below, be a synchronization point: $Depr^a = b$ $Depr^b = c$ $Depr^c = a$

Assume that the restricters associated with information units a , b and c have been created in this order, and that the restricter temporal requirements are met. The creation of a b restricter illustrates case 1) for the restricter associated with information unit a . The resulting configuration may be dealt with as follows: since a is conditioned by information unit b and b by c , the transformation $Depr^a = b \Rightarrow Depr^a = c$ may be performed. Now, when a c restricter is created, $Depr^a$ and $Depr^c$ become coupled dependency expressions illustrating case 2) : $Depr^a = c$ $Depr^c = a$. The resulting configuration may be dealt with as follows: as soon as a c restricter is created, a and c must immediately be delivered (recall that the restricter temporal requirements were assumed to be met); thereby enabling the delivery of b because $Depr^b = c$.

In practice, this is usually more complex, as the dependency expressions may include several arguments that can be combined using different logical operators and temporal requirements have to be taken into account as well. Two restricter transformations have been defined in the general restricter algorithm which have to be applied as many times as required in order to evaluate a synchronization point. The first transformation (called transformation A) deals with case 1) and the second (transformation B) addresses case 2). The underlying idea consists in modifying dynamically the dependency and the temporal characteristics of the restricters in order to avoid delivery conflicts as those pointed out above, without altering the initial delivery constraints expressed by the synchronization point.

D. Merging delivery time intervals in the restricter transformations

Let us now investigate in more details the restricter transformations from the point of view of the temporal requirements. To do this, consider restricters $(a, \Delta a, b \wedge c)$ and $(b, \Delta b, d)$, which are assumed to have been created some time before the beginning of intervals Δa and Δb . These restricters fulfill the condition associated with transformation A; as a consequence of this transformation, b in restricter $(a, \Delta a, b \wedge c)$ may be replaced by d .

The difficult point now consists in determining the delivery time intervals to be defined within the restricters, once the transformations are performed (only transformation A is considered here). Different alternative solutions will progressively be introduced and carefully analyzed before characterizing the correct one:

1. Using restricter $(a, \Delta a, d \wedge c)$ is not correct, as illustrated by temporal configuration (i) of Fig. 6: a might

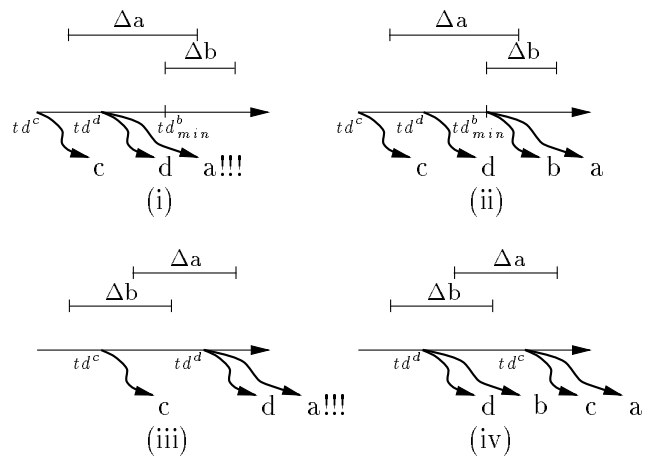


Fig. 6. Temporal configurations

be delivered at time td^d , before the start of interval Δb leading therefore to a contradiction with the initial restricter $(a, \Delta a, b \wedge c)$;

2. Using restricter $(a, \Delta a \cap [td_{min}^b, \infty[, d \wedge c)$ enforces that the delivery of a is based on its own delivery time interval (Δa) and on a time constraint inherited from b ($[td_{min}^b, \infty[$) (note that the update of the delivery time interval must be performed for each elementary conjunction of the dependency expression, and not at the global level of the restricter as done here for simplification purpose): as a consequence, in temporal configuration (ii), information unit a , if already received from the lower service, may be delivered at time td_{min}^b just after the delivery of b ; this way to proceed is however incorrect for temporal configuration (iii): information unit a is delivered at time td^d as a consequence of the delivery of d ; this is incorrect since b has not been delivered during Δb ;
3. Using restricter $(a, \Delta a \cap [td_{min}^b, \infty[, d_{Ld} \wedge c_{Lc})$, where Ld (respectively Lc) represents a maximum latency information associated with d (respectively c) in order to express that the delivery of d (respectively c) must occur before Ld (respectively Lc) for enabling the delivery of a . Latency information (for example Lc) is associated with every Boolean variable of a dependency expression and is initially equal to the maximum delivery time of the information unit whose delivery is controlled by the dependency expression (therefore $Lc = td_{max}^c$); for information unit d the situation is slightly different, since d is the result of transformation A (d serves as a substitute for b); looking at b restricter, the latency information associated with d is $Ld = td_{max}^b$, which is inherited in the restricter of a (more precisely Ld is equal to $\min(td_{max}^a, td_{max}^b)$). Doing in this way implies that the delivery of a in configuration (iii) is not anymore possible; configuration (iv) illustrates a possible delivery of a .

Let us now consider another example, with the following two restricters: $(a, \Delta a, b \vee c)$ and $(b, \Delta b, \neg d \wedge e)$. Applying transformation A to the first restricter generates the

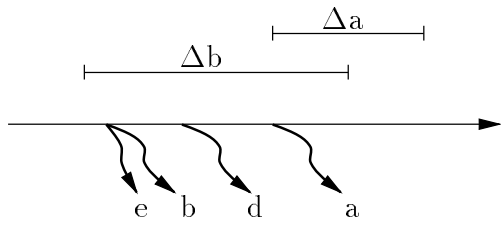


Fig. 7. Temporal configuration

following restricter:

$$(a, \underbrace{\neg d_{Ld} \wedge e_{Le}}_{\Delta a^1} \vee \underbrace{c_{Lc}}_{\Delta a^2})$$

with $\Delta a^1 = \Delta a \cap [td_{min}^b, \infty[$, $\Delta a^2 = \Delta a$ and $Ld = Le = \min(td_{max}^a, td_{max}^b)$ and $Lc = td_{max}^a$.

Here, the maximal latency information Ld is no longer sufficient to cope with negative premises. The configuration, depicted in Fig. 7, states that e and d are delivered before their maximal latencies (both equal to td_{max}^b). However, d being delivered before td_{min}^a , information unit a cannot be delivered as a result of the delivery of e , although a could be delivered as a result of the delivery of b in the initial restricter $(a, \Delta a, b \vee c)$. To address this issue, the maximal latency information Ld has been replaced with a set of preconditions (i.e., $\{e\}$) and a latency interval ($\Delta L = \Delta b$); the restricter may then be rewritten as:

$$(a, \underbrace{PC(\neg d)_{\Delta L} \wedge e_{Le}}_{\Delta a^1} \vee \underbrace{c_{Lc}}_{\Delta a^2}) \text{ with } \Delta L = \Delta b \text{ and } PC = \{e\}$$

where timed negative premise $PC(\neg d)_{\Delta L}$ is satisfied if one of the following cases arises:

- information unit d has not been delivered;
- information unit d has been delivered after ΔL ;
- information unit d has been delivered at $td^d \in \Delta L$ and all information units in PC have been delivered before td^d .

In conclusion, one can stress that the restricter algorithms refer to timed dependency expressions, which feature the following characteristics:

- delivery time intervals are associated with each elementary conjunction of the dependency expression;
- a maximal latency information is associated with each positive premise;
- a set of preconditions and a latency interval are associated with each negative premise.

Here simple examples have been utilized to present informally the details of the transformations performed by the general restricter algorithm. It is therefore not surprising that the description of these transformations appears slightly ambiguous as numerous cases have to be accounted for. This is primarily why we proposed a complete formalization of the conditional delivery mechanism and formally assessed its utilization for implementing the synchronization requirements of the interactive training application.

This is based on the RT-LOTOS Formal Description Technique. Basic background information on RT-LOTOS, its associated tool environment and the validation of the conditional delivery mechanism will now be given in the next section.

V. FORMALIZATION

Formal description techniques (FDTs for short) are increasingly recognized as particularly important for the successful design of large distributed and time-critical systems. They present several advantages relative to conventional design methods. In particular, they allow for:

- the expression of unambiguous specifications, understanding these specifications relying solely on the FDT's formal semantics; frequently, these specifications are also concise and make it possible to capture the essential features of the system without entering into specific implementation-oriented details;
- the analysis of the specifications with the purpose of proving properties of the system under design.

RT-LOTOS is a temporal extension of the standard formal description technique LOTOS [19], which is part of the family of process algebras [20] (see [21], [22], [23] for other temporal extensions of LOTOS). The latter have recently received a great deal of attention for two reasons: they permit formal specifications to be expressed at different levels of abstraction, and a general theory of behavioral equivalences has been developed, providing therefore mathematical tools for formally comparing the behavior of different specifications. This is a major advantage compared to established formalisms such as Petri nets [24]. A brief introduction to LOTOS and RT-LOTOS is given in appendix, and detailed tutorial papers are available in [11], [12], [25].

A. General architecture of the whole specification

A complete RT-LOTOS specification of the interactive training application has been developed. Fig. 8 describes the specification architecture in terms of the RT-LOTOS processes involved.

The stream submissions by SS and AS are specified by the *Synchronous_Server* and *Asynchronous_Server* processes respectively. The *Synchronous_Medium* and *Asynchronous_Medium* processes characterize the stream transfer procedures performed by the transport service towards the student's multimedia workstation from SS and AS respectively. Medium processes may be parameterized to provide a reliable or a non reliable service as well as to select the minimum and maximum transfer delays. The *Jitter_Control* process supports a conventional jitter compensation mechanism. The application located in the student's workstation is represented by a simple demultiplex process intended to deliver the information media to the respective audio, text and slide presentation devices. A transport connection, modeled through a *Control_Medium* process, is assumed to be established between the student's workstation and SS to enable the student to interact with the training application (i.e. to answer the questions).

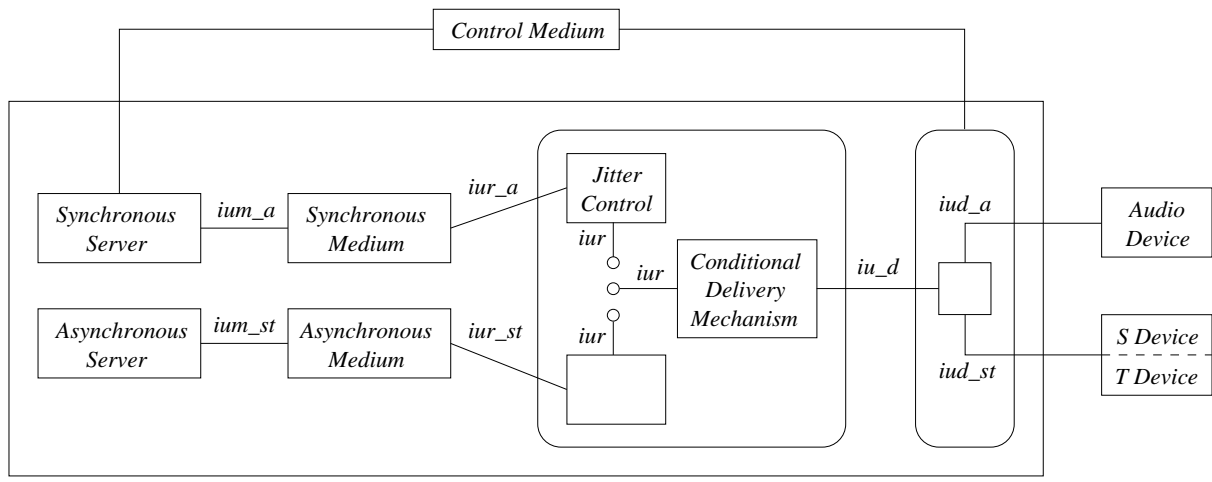


Fig. 8. Training application specification architecture

The server processes (i.e. synchronous and asynchronous) handle the submission of their respective streams. Process *Synchronous_Server* characterizes the submission of the audio stream, whereas *Asynchronous_Server* characterizes that of the text and the slide information units. The *Synchronous_Medium* and *Asynchronous_Medium* processes characterize the communication media (i.e. the transport service) used for interconnecting both servers to the student's workstation. These processes may be parameterized by different QoS parameters, among them the minimum (D_{min}) and the maximum (D_{max}) transfer delays. For each information unit received from gate ium , a non-deterministic delay is selected, between D_{min} and D_{max} , before offering the information unit to gate iur . Different processes have been implemented to characterize different transport services as a function of the expected quality of service. Process *Jitter_Control* implements a buffering technique to compensate for the jitter introduced by the transport service. For each information unit received at gate iur_x , a delivery time interval is calculated by taking into account: (i) the time stamp associated with the current information unit (embedded in $IU_{receiver}$), (ii) the residual jitter (RJ), (iii) the medium jitter (MJ) and finally (iv) the temporal reference associated with the local reception of the first information unit ($T0$). Note that information units do not need to be stored by this process, as buffering is already achieved by the conditional delivery mechanism.

B. Formal specification of the conditional delivery mechanism

The conditional delivery mechanism is implemented by means of a specific protocol entity (see Fig. 9), whose behavior is formalized by process *Receiver_Synchro_Entity*. This process corresponds to an instance of the *Receiver_Control* process, in which the set of the delivered information units and the set of restricters are initialized with empty, and the global time with zero.

```

process Receiver_Synchro_Entity [iur,iud] : noexit :=
  Receiver_Control [iur,iud]
  (Empty_Restrictor_Set, Empty_Delivered_IU_Set, 0)
endproc

```

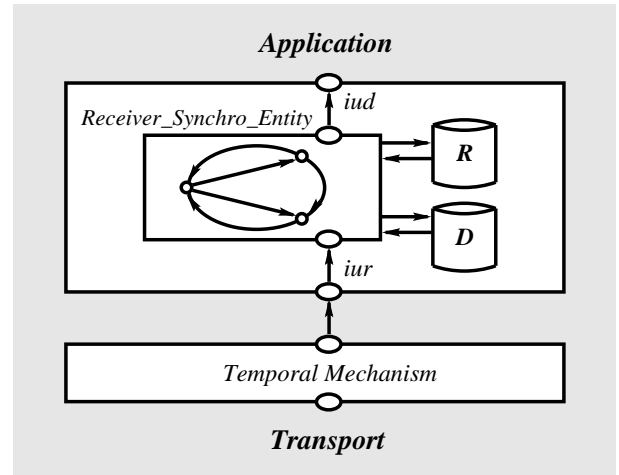


Fig. 9. Architecture of process *Receiver_Synchro_Entity*

Receiver_Control is further refined into two processes, namely *IU_Receiver* and *IU_Delivery*, dealing respectively with the reception of new information units from the transport service, and the delivery of information units to the user's application.

```

process Receiver_Control [iur,iud]
  (R:Restrictor_Set_type,D:Delivery_IU_Set_type,GT:Time) :
  noexit :=
  let instantaneous_delivery:bool = May_Deliver (R,D,GT) in
  [instantaneous_delivery]->
  i: IU_Delivery [iur,iud] (R,D,GT)
  [] [not (instantaneous_delivery)]->
  i: IU_Receiver [iur,iud] (R,D,GT)
endproc

```

Process *Receiver_Control* expresses a non deterministic choice which is resolved by evaluating predicate *May_Deliver* in the current configuration of the protocol entity which includes the current set of restricters (R), the

current set of previously delivered information units (D) and the current global time (GT): if true, there exists an information unit to be delivered (the subsequent behavior then corresponds to process $IU_Delivery$), and false, there exists, at the current time, no information to be delivered (the subsequent behavior then corresponds to process $IU_Receiver$).

```

process  $IU\_Delivery$  [ $iur, iud$ ]
( $R:Restrictor\_Set\_type, D:Delivery\_IU\_Set\_type, GT:Time$ ) :
noexit :=
let  $M : IU = IU\_to\_Deliver$  ( $R, D, GT$ ) in
   $iud ! M; Receiver\_control$  [ $iur, iud$ ]
  ( $Release\_from\_Restrictor\_Set$ ( $R, M$ ),
    $Update\_Delivered\_IU\_Set$ ( $D, M, GT$ ),  $GT$ )
endproc

```

Process $IU_Delivery$ characterizes the delivery of some information unit M to the upper layer; M to be delivered is identified by means of function $IU_to_Deliver$ and further offered at gate iud which formalizes the interface between the synchronization layer and the upper user layer. Once M has been delivered, the process transforms itself recursively into process $Receiver_Control$, the configuration of the synchronization entity being updated by means of functions $Release_from_Restrictor_Set$ and $Update_Delivered_IU$ whose purpose is to:

- release the restricter associated with the delivery of M ;
- update the set of restricters to take into account the delivery of M ;
- append M to the set of delivered information units.

Process $IU_Receiver$ characterizes the behavior of the synchronization entity when no further information units may be delivered; two situations have to be accounted for:

- time is progressing until the temporal constraints associated with information units previously received and currently stored in the synchronization entity become satisfied; this time value is determined by the $Delivery_Wait_time$ function and time progression is formalized by the delay operator;
- a new information unit (N) is received from the transport service, resulting in the creation of the new restricter associated with N and in the transformation of the updated set of restricters to check whether some information units can be delivered at the current time. Note finally, that as soon as a new information unit is received from the transport service, the behavior branch corresponding to the delay alternative just disappears (see the semantics of the choice operator).

Functions $Release_from_Restrictor_Set$ and $Add_to_Restrictor_Set$ perform the restricter transformations which have informally been discussed in the previous section; they have completely been formalized and implemented, and the interested reader can refer to [26] for details.

```

process  $IU\_Receiver$  [ $iur, iud$ ]
( $R:Restrictor\_Set\_type, D:Delivery\_IU\_Set\_type, GT:Time$ ) :
noexit :=
(let  $WT : Time = Delivery\_Wait\_Time$  ( $R, GT$ ) in
  delay( $WT$ ) i;  $Receiver\_control$  [ $iur, iud$ ] ( $R, D, GT+WT$ ) )
)
   $iur @ LT ? N:IU; Receiver\_control$  [ $iur, iud$ ]
  ( $Add\_to\_Restrictor\_Set$ ( $R, N$ ),  $D, GT+LT$ )
endproc

```

This specification is a good illustration of operator $@$; using this operator, the (relative) time at which an information unit is received from the transport service is recovered and may then be added to the current global time.

C. Validation results

An RT-LOTOS tool environment (called RTL for RT-LOTOS Laboratory) is being developed at LAAS-CNRS to validate the correctness of a specification. In particular, it provides a simulation capability. By generating several execution scenarios of the complete RT-LOTOS specification whose global architecture has been depicted in Fig. 8, the high level synchronization requirements of the interactive training application and the correctness of the formal specification of the conditional delivery mechanism can both be assessed with different assumptions made on the QoS parameters of the underlying transport service (see the medium process parameters). In case of non-determinism (behavior and/or time non-determinism), random decisions are made according to probabilistic laws (the uniform distribution law is being implemented in the tool). This tool uses a graphical interface for display of the simulation results: the user specifies the RT-LOTOS specification gates he wants to observe in some execution scenario, and action occurrences at these gates are featured (possibly with their associated data parameters) on temporal axes. Fig. 10 gives an example of execution scenario. By examining these time-lines the execution scenario can be analyzed and it is possible to check whether the conditional dependency relations and their associated temporal constraints have been satisfied. Note that the labels associated with the time-lines correspond to the gates defined in Fig. 8 (e.g., the emission of audio segments is represented by action occurrences on gate iur_a , whereas their remote reception from the transport service is represented by action occurrences on gate iud_a , in which clearly the transport jitter has been compensated for and the delivery of the audio segments is synchronized with the delivery of the slide and text information units).

VI. CONCLUSIONS

In this paper multimedia synchronization issues related to the co-ordination, scheduling and presentation of multimedia objects within a distributed framework have been presented. Special emphasis has been placed on how to meet specific dependency and temporal requirements, providing thus a new approach to intra- and inter-stream synchronization issues.

The so-called conditional delivery mechanism, whose usefulness has been demonstrated for a simple distributed

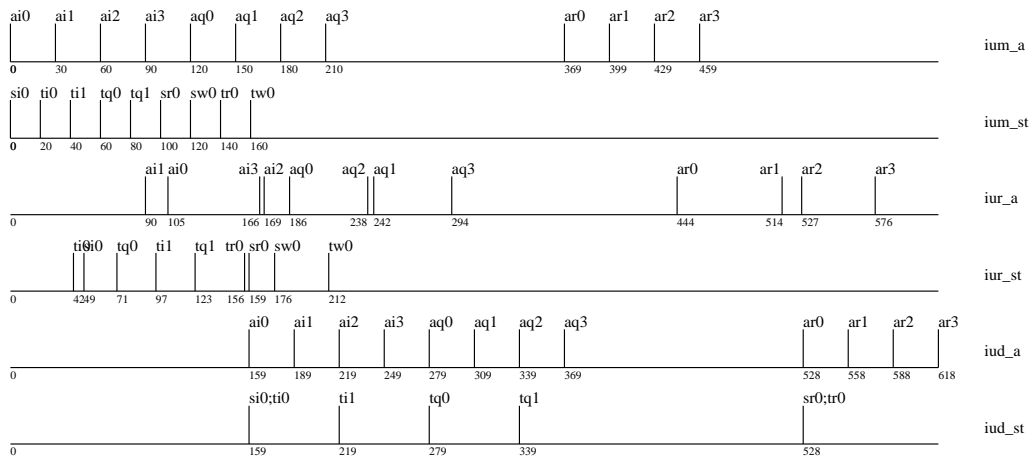


Fig. 10. Execution scenario

training application, has been put forward as the main communication facility. Dependency relations among information units from a single stream or several streams of the same bundle have been considered explicitly specified by the upper layer. These causal relations, as defined at the encoded media level offer a finer level of granularity than high level requirements expressed for example by Petri nets-like models such as OCPNs [24], which address the whole media. It is the authors' opinion however that high-level temporal requirements can be translated into the proposed causal relations by taking into account the specific encoding scheme for the (continuous) media. Ongoing research addresses this by showing how these dependency relations among information units can be derived from more abstract and user-oriented pre-negotiated synchronization scenarios, which could then be mapped onto the proposed conditional dependency scheme [17]. Finally, it has been shown that the strength of the proposed synchronization mechanism, the conditional delivery mechanism, directly results from the level of granularity at which the causal relations are expressed.

We are currently focusing on how to use pre-negotiated synchronization scenarios based on the MHEG standard [27], which defines spatio-temporal synchronization schemes by composing "child" objects within a "parent object". A composite object is further defined for encapsulating the spatio-temporal links among its components (i.e. the individual media). Communication functionality introduced in the paper could be used for implementing the conditional synchronization scheme of a MHEG document, when transferring it across a network. Thus a composite MHEG object could be split into several component objects which would individually be submitted to the communication service; synchronization among the component objects would be ensured by the conditional delivery facility based on a synchronization scheme derived from the synchronization actions of the original composite object.

Finally the importance of formal description techniques for specifying complex synchronization mechanisms has been highlighted and the proposed temporal extension of

the LOTOS formal description technique (RT-LOTOS) has fulfilled its expected goal [28]. The availability of a design environment based on RT-LOTOS (the RTL tool) has made it possible to achieve initial validation results of the proposed conditional delivery mechanism. Further work in this direction is being carried out to upgrade the environment by including reachability analysis capabilities [29].

ACKNOWLEDGMENTS

This work has been supported by CNET (Grant 92-1B-178) as part of CESAME, a CNET-CNRS collaborative project on High Speed Multimedia Systems. L.F.R.C. Carmo and R.C. de Oliveira were partly supported by a grant from CNPq/Brazil.

REFERENCES

- [1] B. Furht, "Multimedia Systems: an Overview", *IEEE Multimedia*, vol. 1, no. 1, pp. 47-59, 1994.
- [2] D.P. Anderson and G. Homsy, "A Continuous Media I/O Server and its Synchronization Mechanisms", *IEEE Computer*, pp. 51-57, Oct. 1991.
- [3] R. Steinmetz, "Synchronization Properties in Multimedia Systems", *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 3, pp. 401-412, Apr. 1990.
- [4] D. Ferrari, "Client Requirements for Real-Time Communication Services", *IEEE Communication Magazine*, vol. 28, no. 11, pp. 65-72, Nov. 1990.
- [5] C. Partridge, "Isochronous applications do not require jitter-controlled networks", RFC 1257, 1991.
- [6] H. Santoso, L. Dairaine, S. Fdida, and E. Horlait, "Preserving temporal signature: a way to convey time constrained flows", in *Proc. IEEE GLOBECOM'93*, Houston, 1993.
- [7] W. Yen and I.F. Akyildiz, "On the synchronization mechanisms for multimedia integrated services networks", in *Multimedia Transport and Teleservices*, LNCS 882, pp. 168-184. Springer-Verlag, 1994.
- [8] G.S. Blair, L. Blair, H. Bowman, and A. Chetwynd, "Formal Support for the Specification and Construction of Distributed Multimedia Systems (The Tempo Project)", 1993.
- [9] P.V. Rangan, S. Ramanathan, H.M. Vin, and T. Kaepfner, "Techniques for multimedia synchronization in network file systems", *Computer Communications*, vol. 16, no. 3, pp. 168-176, 1993.
- [10] L. Li, A. Karmouch, and N.D. Georganas, "Multimedia Teleorchestra with Independent Sources: Part 1 - Temporal Modelling of Collaborative Multimedia Scenarios & Part 2 - Synchronization Algorithms", *Multimedia Systems*, vol. 1, no. 4, pp. 143-153, 154-165, 1994.

- [11] T. Bolognesi and E. Brinksma, "Introduction to the ISO Specification Language LOTOS", *Computer Networks and ISDN Systems*, vol. 14, no. 1, pp. 25–59, 1987.
- [12] J.-P. Courtiat and R.C. de Oliveira, "About time nondeterminism and exception handling in a temporal extension of LOTOS", in *Protocol Specification, Testing and Verification XIV*, Vancouver, Canada, June 1994, pp. 37–52, Chapman & Hall.
- [13] L.F.R.C. Carmo, P. de Saqui-Sannes, and J.-P. Courtiat, "Basic synchronization concepts in multimedia systems", in *Proc. 3rd IEEE Int. Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, LNCS 712, pp. 94–105. Springer-Verlag, San Diego, 1993.
- [14] J.-P. Courtiat, L.F.R.C. Carmo, and R.C. de Oliveira, "A new mechanism for achieving inter-stream synchronization in multimedia communication systems", in *Proc. IEEE Int. Conf. on Multimedia Computing and Systems*, Boston, May 1994, pp. 173–182.
- [15] J.-P. Courtiat, R.C. de Oliveira, and L.F.R.C. Carmo, "Towards a new multimedia synchronization mechanism and its formal specification", in *Proc. ACM Multimedia'94*, San Francisco, Oct. 1994, pp. 133–140.
- [16] L. Dairaine, *Techniques de synchronisation pour les communications dans les systèmes haut-débit multimédia*, PhD thesis, Université Pierre et Marie Curie, Paris, 1994.
- [17] L.F.R.C. Carmo, *Méthodes de synchronisation dans les systèmes répartis multimédia: une approche intégrant relations de causalité et contraintes temporelles*, PhD thesis, Université Paul Sabatier, Toulouse, France, 1994.
- [18] L.F.R.C. Carmo and J.-P. Courtiat, "Implementing intra-stream synchronization by means of conditional dependency expressions", in *Proc. 5th Conf. on High Performance Networking (HPN'94)*, Grenoble, France, June 1994, pp. 187–200, North-Holland.
- [19] ISO Standard 8807, *LOTOS, a Formal Description Technique based on Temporal Ordering of Observational Behavior*, 1988.
- [20] R. Milner, *Communication and Concurrency*, Prentice Hall, 1989.
- [21] L. Léonard and G. Leduc, "An enhanced version of timed LOTOS and its application to a case study", in *Proc. 6th Intern. Confer. on Formal Description Techniques (FORTE'93)*, Boston, Massachusetts, Oct. 1993.
- [22] M.A. Marsan, A. Bianco, L. Ciminiera, R. Sisto, and A. Valenzano, "A LOTOS Extension for the Performance Analysis of Distributed Systems", *IEEE/ACM Transactions on Networking*, vol. 2, no. 2, pp. 151–165, Apr. 1994.
- [23] T. Bolognesi and F. Lucidi, "LOTOS-like process algebras with urgent or timed interactions", in *FORTE'91*, pp. 249–264. IFIP, North-Holland, 1992.
- [24] T.D.C. Little and A. Ghafoor, "Synchronization and Storage Models for Multimedia Objects", *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 3, pp. 413–427, Apr. 1990.
- [25] J.-P. Courtiat and R.C. de Oliveira, "On RT-LOTOS and its application to the formal design of multimedia protocols", Submitted for publication.
- [26] J.-P. Courtiat, R.C. de Oliveira, and L.F.R.C. Carmo, "The RT-LOTOS formal specification of the conditional delivery mechanism", Information Systems, special issue on Multimedia, Springer-Verlag, to appear.
- [27] ISO/IEC DIS 13522-1, *Coding of Multimedia and Hypermedia Information (MHG)*, Oct. 1994.
- [28] J.-P. Courtiat, R.C. de Oliveira, and Laurent Andriantsiferana, "Specification and Validation of Multimedia Protocols using RT-LOTOS", in *Proc. 5th Workshop on Future Trends in Distributed Computing Systems*, Cheju Island, Republic of Korea, Aug. 1995.
- [29] J.-P. Courtiat and R.C. de Oliveira, "A Reachability Analysis of RT-LOTOS Specifications", in *Proc. 8th Intern. Confer. on Formal Description Techniques (FORTE'95)*, Montreal, Quebec, Canada, Oct. 1995.

APPENDIX

I. BRIEF INTRODUCTION TO LOTOS AND RT-LOTOS

In LOTOS a system is seen as a process that may include several sub-processes. In turn, a sub-process is also a process, so that a LOTOS specification describes a sys-

tem via a hierarchy of process definitions. A process is an entity capable of performing internal, unobservable actions and interacting with other processes, which form its environment. Complex interactions among processes are built up out of elementary synchronization units called events or (atomic) interactions or simply actions. Events imply process synchronization, because the processes interacting on an event (two or more) participate in its execution at the same time. Such synchronization may involve an exchange of data. Events are atomic in that they occur instantaneously, without consuming time. An event is thought of as occurring at an interaction point, or *gate*, and in the case of synchronization without data exchange, the event name and the gate name coincide. The syntax for process definition in LOTOS is as follows:

```

process Process_Identifier [Formal_Gate_List] (Parameter_List)
  := Process_Behavior
endproc

```

where *Parameter_List* is a list of variable declarations, and *Process_Behavior* is a *behavior expression*. A behavior expression is built up by applying an operator to other behavior expressions. A behavior expression may also include *instantiations* of other processes.

The instantiation of a LOTOS process looks like the invocation of a procedure in a programming language. Of course a process instantiation refers to a process definition which must exist somewhere in the specification. The following syntax is used for process instantiation:

```

Process_Identifier [Actual_Gate_List] (Value_Expression_List)

```

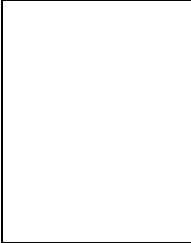
A partial list of LOTOS behavior expressions is given in the following table, which includes all LOTOS operators used in the specification of the conditional delivery mechanism.

process instantiation	$P[g_1, \dots, g_m](v_1, \dots, v_n)$
internal action prefix	$i; B$
observable action prefix	$g O_1 \dots O_n; B$ where $O_i = ?x:s$ or $!v$
choice	$B_1 [] B_2$
parallel	$B_1 [g_1, \dots, g_m] B_2$ or $B_1 B_2$
hiding	hide g_1, \dots, g_m in B
variable declaration	let $x:s=v$ in B
guard predicate	$[v] \rightarrow B$

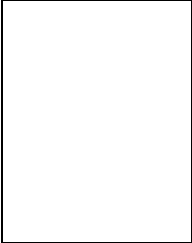
Symbols B , B_1 , B_2 in the table stand for any behavior expression, P refers to any process identifier, g is a gate name, i the internal action, x a variable name, t a data type, and v a value expression.

Many programming languages such as Ada, Esterel and Occam, feature delay operators to suspend activities. For example, Ada has a construct, $delay(t); P$ which means "wait for t units of time and then execute P ". Similarly, we write $delay(t)P$ to denote a process which idles for t units of time and then behaves as P . This construct is fairly classical and its use for modeling real-time systems does not need further emphasis. Construction $a@t; P$ indicates that the relative time at which action a will occur

is stored in variable t (t can obviously be replaced by any other variable name). This time then corresponds to a relative time starting from the instant when action a to be offered. In other words, t characterizes the time period during which action a has remained offered before actually occurring. Finally, construction $i\{t\};P$ indicates that the internal action i will occur in the interval $[0, t]$ at an arbitrary time instant which is not chosen by the environment but by the process itself. This construction has been used here to express time non determinism (for details on time non determinism, other time-related features of RT-LOTOS and its formal semantics see [12], [25]).

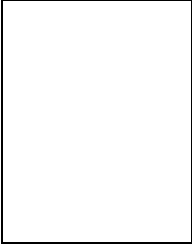


Roberto Cruz de Oliveira received the B.S. and M.S. degrees in electrical engineering from the Federal University of Rio de Janeiro, Brazil, in 1988 and 1991, respectively. He is presently completing his Ph.D. studies at LAAS/CNRS, Toulouse, France. His current research interests include formal methods, multimedia communications, and real-time distributed systems. His email address is: cruz@laas.fr.



Jean-Pierre Courtiat graduated in computer science from ENSEEIHT in 1973. He received the Ph.D. and Doctorat d'Etat degrees in computer science from the University of Toulouse, France, in 1976 and 1986, respectively.

After having been a researcher at LAAS/CNRS from 1973 to 1976, he has been an expert of the French technical cooperation from 1976 to 1980 with an appointment at the Federal University of Rio de Janeiro, as Professor of Computer Science. In 1980, he came back to LAAS, as "charge de recherche au CNRS" (a research position of the French National Council of Scientific Research). At LAAS, Jean-Pierre Courtiat works in the OLC (Software and Communication) research group, where he leads the SFP (Formal Specification of Protocols) research team. His research interests include the design of protocols, as well as the definition and application of formal methods for the specification, verification and testing of protocols and distributed systems, areas in which he has authored or co-authored more than 70 international publications. Currently, he participates to several researches dealing with the semantics of concurrency and the expression of time-constraints in the formal description techniques, as well as the application of these techniques to the formal design of co-operative high speed multimedia distributed systems. In carrying out these researches, he has taken several responsibilities in different European research projects, and has participated to standardization activities, as an expert of AFNOR and ISO. He is currently one of the co-managers of CESAME, a CNET-CNRS collaborative project on High Speed Multimedia Systems sponsored by France-Telecom. He is a member of ACM and IEEE. His email address is courtiat@laas.fr.



Luiz Fernando Rust da Costa Carmo received the B.S. degree in electronic engineer in 1984, and the M.Sc. degree in computer sciences in 1988, both from the Federal University of Rio de Janeiro, Brazil, and the "Doctor es Sciences" degree in 1994, from the University Paul Sabatier, Toulouse, France. Presently, he is a member of research staff of the Computer Center of Federal University of Rio de Janeiro (NCE/UFRJ, C.P. 2324, CEP 20001-970, Rio de Janeiro, Brazil). His research interests are

in the area of formal specification, modeling of high speed network, and multimedia computing and communication. His email address is: rust@nce.ufrj.br.