An Intermedia Synchronization Mechanism for Multimedia Distributed Systems

S.E. Pomares Hernandez*,
J. Estudillo Ramirez,
L.A. Morales Rosales and
G. Rodriguez Gomez
Computer Science Department,
National Institute of Astrophysics, Optics and Electronics (INAOE), Mexico
E-mail:{spomares, jestudillo, lamorales, grodrig}inaoep.mx
*Corresponding author

Abstract: Emerging distributed multimedia systems such as Telemedicine, Tele-Conference and IPTV, deal with simultaneous geographically distributed sources by transmitting heterogeneous data, such as text, images, graphics, video and audio. The preservation of temporal relations that consider different data types and simultaneous distributed sources is an open research area. Although several works try to specify and execute at runtime distributed temporal multimedia scenarios, they are far from resolving the problem. This paper proposes a synchronization mechanism to be used at runtime in distributed multimedia systems. One original aspect of the present work is that we avoid the use of a common reference by executing all possible multimedia temporal relations according to their causal dependencies. We emulate the mechanism considering a wide area network environment and using MPEG-4 encoders. The emulation results show that our mechanism is effective in reducing the intermedia synchronization error that can exist. The present work does not require previous knowledge of when, nor for how long, the media involved of a temporal scenario is executed.

Keywords: Multimedia Distributed Systems, Temporal Synchronization, Causal Ordering

Reference

Biographical notes: Saul Eduardo Pomares Hernandez is a Researcher in the Computer Science Department at the National Institute of Astrophysics, Optics and Electronics, in Puebla, Mexico. He completed his PhD degree at the Laboratory for Analysis and Architecture of Systems of CNRS, France in 2002. Since 1998, he has been researching in the field of distributed systems, partial order algorithms and multimedia synchronization.

1 INTRODUCTION

Multimedia systems deal with heterogeneous data, such as text, graphics, images, audio, video and animations. Generally, multimedia data is grouped into two types: *continuous media* (e.g. audio and video) and *discrete media* (e.g. text, data and images) (Geyer et al., 1996). The main difference between these two types is that while continuous media events are considered to be executed during a period of time, discrete media events are considered to be executed at specific timeless points. One open research area in distributed multimedia systems involves intermedia synchronization. Intermedia synchronization concerns the preservation of temporal dependencies among the application data from the time of generation to the time of presentation. For example, in a one-to-many synchronous telemedicine session, a specialist exposes to a group of colleagues his diagnosis, which was obtained from the medical images of a patient (e.g. radiography, magnetic resonance and tomography). In this case, the specialist sends medical images (discrete media) through a teleradiology tool and he makes use of two continuous medias (audio and video) to give comments about them. The intermedia synchronization, in this scenario, must ensure that at any reception participant (*Client*) (see Figure 1), the images must be presented according to the phrases pronounced with its corresponding video. A simple on-site meeting situation such as "We distinguish in the next tomography..." (associated with the correspondent concurrent presentation image action) is not simple to reproduce on a telemedicine session since the communication channels are asynchronous and independent, and the media involved is heterogeneous (Balaouras et al., 2000). The term *next* in the scenario establishes, at an application level, a temporal dependency with the remaining media involved. In such case, if temporal dependencies are not ensured, it is possible that a receiver can associate, without distinction, this phrase with the current, the preceding or the following medical image. Satisfying temporal dependencies is even more complex for cooperative many-to-many multimedia scenarios, where geographically distributed participants simultaneously communicate. The present work mainly focuses on this last kind of scenarios, where in principle there are no global references, such as a wall clock or a shared memory.

Several works attempt to give a possible solution to the intermedia synchronization in distributed systems (Bulterman D., 2008; Geyer et al., 1996; Ishibashi et al., 1999; Plesca et al., 2005; Shimamura et al., 2001); however, as we will see in the related work section, these works are far from resolving the problem. In this paper, we propose an intermedia synchronization mechanism based on the *logical mapping* concept presented by Morales Rosales and Pomares Hernandez (2006). We use the logical mapping to specify, according to causal dependencies, any kind of temporal relationship among the multimedia data involved: continuous-continuous, discrete-discrete, and discrete-continuous relations. Our mechanism takes as base the specification obtained by the logical mapping in order to establish "when" a multimedia data must be delivered, and consequently, executed by a participant.

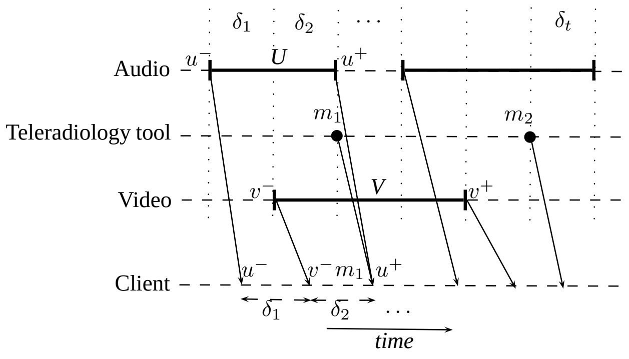


Figure 1: A telemedicine multimedia scenario example

We emulate the mechanism considering some wide network conditions and using MPEG-4 encoders (Mackie et al., 2000). We note that even when the audio and video are considered to be continuous, their transmission using MPEG-4 (and similar encoders) is in fact non-continuous since compression techniques, such as silence compression and remotion of temporal redundancy, are used. We show in this paper how our mechanism takes into consideration this behavior and is able to reduce the intermedia synchronization error without needing to use previous knowledge of the system nor global references.

This paper is structured as follows. Section 2 presents the most relevant related work. Next, in Section 3, the system model is described, and the background information is provided. In Section 4, the temporal model concerning discrete and continuous media is presented. The synchronization mechanism with a detailed description is shown in Section 5. The emulation results are presented in Section 6. Finally, some conclusions are given in Section 7.

2 RELATED WORK

Many works related to multimedia synchronization exist. We classify them according to the mode of execution, namely: offline mode and inline mode. The offline mode refers to when the specification of a temporal scenario is manually made and previous to the media data transmision. The most representative works in this category are: the object composition Petri net model (OCPN) introduced by Little and Ghafoor (1990), the timing and synchronization model of SMIL (Bulterman D., 2008), and the time ontology model of OWL (Bechhofer S., 2006). The main characteristic of the works in this category is that they need complete previous knowledge of the media data, the temporal dependencies and/or the actions sequence order, before carrying out the temporal specification.

On the other hand, we have the inline mode. In this mode the temporal specification is carried out at runtime. Most works in this category are primarily based on the identification and preservation of physical time constraints by using a common reference (wall clock, shared memory, mixer, etc) (Balaouras et al., 2000; Perkins, 2003). These works usually try to answer the synchronization problem by measuring and ensuring, based on a timeline, the period of physical or virtual time elapsed (δ_t) between certain points. Such points can be, for example, the *begin* (x^-), *end* (x^+) and/or discrete events (m_1) of the media involved (see Figure 1).

Few works address intermedia synchronization at runtime without the use of a global reference, which is desirable when the media data have different sources and the transmission delay is not negligible. These last works are primarily based on the identification of logical dependencies. There are two main works based on logical dependencies: The protocol of Shimamura et al. (2001) and the work of Plesca et al. (2005).

Shimamura et al. (2001) establishes six logical precedence relations at an object level (*top, tail, full, partial, inclusive* and *exclusive)*. These relations are specified based on the causal dependencies of the *begin* (v^-) and *end* (v^+) points of the objects. The objects are represented by intervals composed of messages. In order to obtain a fine level synchronization, Shimamura introduces an interval segmentation mechanism that arbitrarily divides the objects into predetermined fixed length segments. This mechanism uses two logical relations: the *precedes* relation and the *concurrent* interval relation. The *precedes* relation of Shimamura is defined as:

$$U \rightarrow V \text{ if } u^+ \rightarrow v^-$$

while the *concurrent* relation is defined as:

$$U \parallel V \text{ if } \neg(u^+ \rightarrow v^-) \wedge \neg(v^+ \rightarrow u^-)$$

We note that this mechanism can be inaccurate since it does not clearly establish, at a segment level, a translation of the possible timeline temporal interval relations identified by Allen (1983) (*before, meets, overlaps, starts, finishes, includes, equals*). Shimamura's work, as a consequence of an arbitrary segmentation and a broad definition of the concurrent interval relation, determines six of Allen's seven basic relations as "concurrent"(see Table 1). A pair of concurrently related segments (intervals) means that no order can be established between the messages that compose them. In other words, it is not possible to clearly determine when the media data must be executed.

Plesca et al. (2005) have considered a practical approach for intermedia synchronization by using causal dependencies. Plesca's work uses causal messages as synchronization points to satisfy temporal dependencies among continuous media. This work, in a heuristic manner, introduces causal synchronization points and shows that these points can be useful, but it does not resolve the problem of when causal messages must be introduced.

Table 1: Allen's relations with their corresponding Shimamura's relations

Allen's Relations	**Shimamura's Relations**
$U before V$	$U precedes V$
$U equals V$	$U concurrent V$
$U meets V$	
$U overlaps V$	
$U during V$	
$U starts V$	
$U finishes V$	

3 PRELIMINARIES

3.1 The System Model

Participants: The application under consideration is composed of a set of participants $P = \{i, j, \ldots\}$ organized into a group that communicates by reliable broadcast asynchronous message passing. A participant can only send one message at a time.

Messages: We consider a finite set of messages M, where each message $m \in M$ is identified by a tuple $m = (p, x)$, where $p \in P$ is the sender of m, and x is the local logical clock for messages of p, when m is broadcasted. The set of destinations of a message m is always P.

Events: Let m be a message. We denote by $send(m)$ the emission event and by $delivery(p, m)$ the delivery event of m to participant $p \in P$. The set of events associated to M is the set $E = \{send(m) : m \in M\} \cup \{delivery(p, m) : m \in M \wedge p \in P\}$. The participant $p(e)$ of an event $e \in E$ is defined by $p(send(m)) = p$ and $p(delivery(p, m)) = p$. The set of events of a participant p is $E_p = \{e \in E : p(e) = p\}$.

Intervals: We consider a finite set I of intervals, where each interval $A \in I$ is a set of messages $A \subseteq M$ sent by participant $p = Part(A)$, defined by the mapping $Part : I \rightarrow P$. We denote by a^- and a^+ the endpoint messages of A, and due to the sequential order of $Part(A)$, we have that for all $m \in A : a^- \neq m$ and $a^+ \neq m$ implies that $a^- \rightarrow m \rightarrow a^+$. We note that when $|A| = 1$ (discrete media), we have that $a^- = a^+$; in this case, a^- and a^+ are denoted indistinctly by a.

3.2 Background and Definitions

Happened-Before Relation for Discrete Media. The Happened-Before relation is a strict partial order defined by Lamport (1978) (i.e. irreflexive, asymmetric and transitive) denoted by $e \rightarrow e'$ (i.e. e causally precedes e') defined as follows:

Definition 3.1. *The causal relation "$\rightarrow$" is the least partial order relation on E satisfying the two following properties:*

1. *For each participant p, the set of events E_p involving p is totally ordered: $e, e' \in E_p \Rightarrow e \rightarrow e' \vee e' \rightarrow e$*
2. *For each message m and destination $p \in P$ of m, the emission of m precedes its delivery; i.e. $send(m) \rightarrow delivery(p, m)$*

By using "$\rightarrow$", Lamport defines that two events are concurrent as follows:

$$e \parallel e' \text{ if } \neg(e \rightarrow e' \vee e' \rightarrow e)$$

A behavior or a set of behaviors satisfies *causal order delivery* if the diffusion of a message m causally precedes the diffusion of a message m', and the delivery of m causally precedes the delivery of m' for all participants that belong to P. Formally, we have:

Definition 3.2. *Causal Order Delivery (broadcast case): If $send(m) \rightarrow send(m')$, then $\forall p \in P : delivery(p, m) \rightarrow delivery(p, m')$*

Partial Causal Relation (PCR). The PCR relation was introduced by Fanchon et al. (2004) (Definition 3.2). It considers a subset $M' \subseteq M$ of messages. The PCR induced by M' takes into account the subset of events $E' \subseteq E$ that refer to *send* or *delivery* events of the messages belonging to M'. In our work, the PCR relation is a non strict partial order (i.e. *reflexive*, *asymmetric*, and *transitive*).

Definition 3.3. *The partial causal relation "$\rightarrow_{E'}$" is the least partial order relation satisfying the two following properties:*

1. *For each participant $p \in P$, the local restrictions of $\rightarrow_{E'}$ and $\rightarrow$ to the events of E'_p coincide: $\forall e, e' \in E'_p : e \rightarrow e' \Leftrightarrow e \rightarrow_{E'} e'$*
2. *For each message $m \in M'$ and $j \in P$, the emission of m precedes its delivery to j : $j \in P \Rightarrow send(m) \rightarrow_{E'} delivery(j, m)$*

Happened-Before Relation for Intervals. Lamport (1986) establishes that an interval A happens before another interval B if all elements that compose interval A causally precede all elements of interval B. This definition is used in the model presented in Section 4. However, according to the specification of $Intervals$ presented in Section 3.1, the causal interval relation "$\rightarrow_I$" can be expressed only in terms of the endpoints as follows:

Property 3.4. *The relation "$\rightarrow_I$" is accomplished if the two following conditions are satisfied:*

1. *$A \rightarrow_I B$ if $a^+ \rightarrow_{E'} b^-$*
2. *$A \rightarrow_I B$ if $\exists C \mid (a^+ \rightarrow_{E'} c^- \wedge c^+ \rightarrow_{E'} b^-)$*

where a^+ and b^- are the endpoints of A and B, respectively, c^- and c^+ are the endpoints of C, and $\rightarrow_{E'}$ is the partial causal order (Definition 3.2) induced on $E' \subseteq E$, where E', in this case, is the subset composed by the endpoint events of the intervals in I. When $|A| = 1$, we have that $a^- = a^+ = a$; and in this case, we consider $a \rightarrow a$.

Next, we present the simultaneous relation for intervals, defined as follows:

Definition 3.5. *Two intervals, A and B, are said to be simultaneous "$|||$" if the following condition is satisfied:*

$$A \mid\mid\mid B \Rightarrow a^- \parallel b^- \wedge a^+ \parallel b^+$$

The definition above means that one interval A can take place at the "same time" as another interval B.

Finally, we present the definition of causal delivery for intervals based on their endpoints as follows:

Definition 3.6. *Causal Broadcast Delivery for Intervals*
If $(a^+, b^-) \in A \times B, send(a^+) \rightarrow_{E'} send(b^-) \Rightarrow$
$\forall p \in P, delivery(p, a^+) \rightarrow_{E'} delivery(p, b^-)$, then
$\forall p \in P, delivery(p, A) \rightarrow_I delivery(p, B)$

4 THE LOGICAL MAPPING MODEL

The logical mapping model, introduced by Morales Rosales and Pomares Hernandez (2006) for continuous media (interval-interval relations), specifies a temporal scenario based on the identification of logical precedences among the media involved. Specifically, a logical mapping translates a temporal relation (see Table 2) to be expressed in terms of the causal relation and simultaneous relation for intervals previously presented (Definition 3.3 and Property 3.4). In order to fully work with temporal relations among multimedia data (continuous and discrete data), we consider in this paper, in addition to the interval-interval relations (Allen, 1983), the point-point relations (Vilain, 1982) for discrete media and the point-interval relations (Vilain, 1982) when discrete and continuous media coexist.

The logical mapping translation (see Table 2) involves every pair of intervals of a temporal relation. For every pair, each interval is labeled as X or Y, such that $x^- \rightarrow y^-$ or $x^- \parallel y^-$. Once the X and Y intervals are identified, they are segmented into four subintervals [1]: $A(X,Y)$, $C(X,Y)$, $D(X,Y)$, and $B(X,Y)$. These data segments, along with the interval definition in Section 3.1, become new intervals[2]. Finally, the general causal structure $S(X,Y) = A(X,Y) \rightarrow_I W(X,Y) \rightarrow_I B(X,Y)$, is constructed, where $W(X,Y)$ determines if overlaps exist between the present pair. We note that the data segments can be constructed at runtime as the events occurs in the system, based on the *happened-before* dependencies of their endpoints.

To summarize, five logical mappings are identified: *precedes*, *simultaneous*, *ends*, *starts*, and *overlaps*. These five logical mappings, as we will show, are sufficient to represent all possible intermedia temporal relations. Notice that the resulting logical mappings represent synchronization specification units, which are automatic processable descriptions of a given temporal relation.

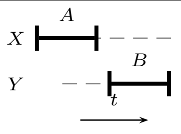
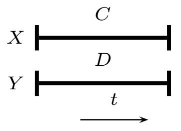
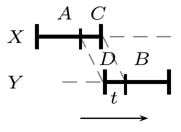
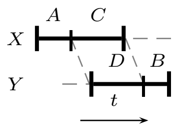
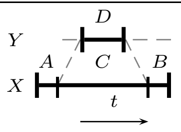
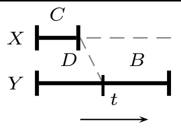
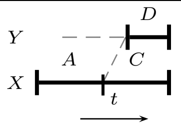
[1]We consider in our model that an interval can be empty. In such case, the following properties apply:
– $\emptyset \rightarrow_I A \vee A \rightarrow_I \emptyset = A$ and $\emptyset \mid\mid\mid A \vee A \mid\mid\mid \emptyset = A$

[2]Henceforth, we can refer them only as A, C, D, and B when there is no ambiguity in the context

Table 2: Logical Mapping

$\forall(X,Y)$	$\in$	$I \times I$
$A(X,Y)$	$\leftarrow$	- if $x^- \to y^-$, $\{x \in X : \mathit{delivery}(\mathit{Part}(Y), x) \to \mathit{send}(y^-)\}$ - otherwise, $\emptyset$
$C(X,Y)$	$\leftarrow$	- if $y^+ \to x^+$, $\{x \in X : \mathit{send}(x) \to \mathit{delivery}(\mathit{Part}(X), y^+)\} - A(X,Y)$ - otherwise, $X - A(X,Y)$
$D(X,Y)$	$\leftarrow$	- if $x^+ \to y^+$, $Y - \{y \in Y : \mathit{delivery}(\mathit{Part}(Y), x^+) \to \mathit{send}(y)\}$ - otherwise, Y
$B(X,Y)$	$\leftarrow$	- if $y^+ \to x^+$, $X - \{A(X,Y) \cup C(X,Y)\}$ - otherwise, $Y - D(X,Y)$
$W(X,Y)$	$\equiv$	$C(X,Y) \mid\mid\mid D(X,Y)$
$S(X,Y)$	$\equiv$	$A(X,Y) \to_I W(X,Y) \to_I B(X,Y)$

Table 3: Allen's interval-interval relations and their logical mapping

Allen's Relations	**Logical Mapping**	**Scenario Example**	**Logical Mapping Expressed on Endpoints**
$U before V$	*precedes*: $A \to_I B$		$a^+ \to b^-$
$U equals V$	*simultaneous*: $(C \mid\mid\mid D)$		$c^- \| d^-, c^+ \| d^+$
$U meets V$	*overlaps*:		
$U overlaps V$	$A \to_I (C \mid\mid\mid D) \to_I B$		$a^+ \to c^-, a^+ \to d^-$ $c^- \| d^-, c^+ \| d^+$ $c^+ \to b^-, d^+ \to b^-$
$U during V$			
$U starts V$	*starts*: $(C \mid\mid\mid D) \to_I B$		$c^- \| d^-, c^+ \| d^+$ $c^+ \to b^-, d^+ \to b^-$
$U finishes V$	*ends*: $A \to_I (C \mid\mid\mid D)$		$a^+ \to c^-, a^+ \to d^-$ $c^- \| d^-, c^+ \| d^+$

We will now present the establishment of the logical mappings for point-point and point-continuous relations. The interval-interval relations have been presented by Morales Rosales and Pomares Hernandez (2006) (see Table 3).

4.0.1 Logical Mapping for Discrete Media

Vilain establishes three possible basic point-point temporal relations based on a timeline, which are *before* ($<$), *simultaneous* ($=$) and *after* ($>$) (See Table 4 left column). After applying Table 2 to each possible temporal relation, we identify two logical mappings, which are *precedes* and *simultaneous* (See Table 4, right column). We can see that these logical mappings are the same as defined at an interval level for the continuous media. They differ in the representation endpoints. For example, when $x < y$ (*before* relation), we have $A = \{x\}$ and $B = \{y\}$, and therefore, we obtain the logical mapping $A \rightarrow_I B$. We have for this logical mapping an endpoint representation $a \rightarrow b$.

Table 4: Vilain's point-point relations and their logical mapping

Basic Point Relations	**Logical Mapping**	**Logical Mapping Expressed on Endpoints**
$x < y$	*precedes*: $A \rightarrow_I B$	$a \rightarrow b$
$y > x$		
$x = y$	*simultaneous*: $C \parallel\!\mid D$	$c \parallel d$

4.0.2 Logical Mapping for Discrete and Continuous Media

Vilain's point-interval algebra (Vilain, 1982) establishes five basic temporal relations (See Table 5, left column). We note that our model considers an additional temporal relation called the *simultaneous* relation. This relation is not considered by Vilain because by using a timeline, a single discrete element (a point) cannot occur with more than one element of an interval at the same time (See Table 5, v *during* U point-interval relation). By translating each temporal relation, we establish five possible logical mappings (*precedes, overlaps, ends, starts,* and *simultaneous*). We can see in Table 5 that these logical mappings are sufficient to represent all possible point-interval temporal relations. It is interesting to note that by considering the continuous media and the discrete media as intervals, the model can indistinctly deal with both of them without a need for any particular considerations.

5 SYNCHRONIZATION MECHANISM

Our synchronization mechanism fulfills at runtime three main tasks: first, it performs the logical mapping translation of a temporal scenario according to the model previously presented; secondly, it carries out the delivery of discrete and continuous media according to the resultant logical mapping specification; and finally, it takes corrective actions in order to reduce the possible synchronization error among the media data.

5.1 Logical Mapping and Causal Delivery Algorithm

The algorithm uses four different causal messages: *begin*, *end*, *cut* and *discrete*, and one type of FIFO messages (*fifo_p*). The first three causal messages and the FIFO message are used for the transmission of continuous media. The algorithm initially defines a *macro* that loads the *send_continuous* or the *send_discrete* function according to the media type handled (see Table 6). Only one reception function is defined since a participant is able to receive continuous media or discrete media data. Next, we describe the main components of the algorithm.

Messages. Formally, a message m in our algorithm is a tuple $m = (k, t, TP, H(m), data)$, where:

- k is a participant identifier.
- $t = VT(p)[k]$ is the local participant clock value with the identifier k when a causal message m (*begin*, *end*, *cut* and *discrete*) is sent. The value of t indicates the sequential number to which a causal message belongs.
- $H(m)$ contains identifiers of messages (k, t) causally preceding causal message m. The information in $H(m)$ ensures the causal delivery of message m. For ($fifo\_p$) messages, structure $H(m)$ is always $H(m) = \emptyset$.
- TP is the type of message (*begin*, *end*, *cut*, *discrete* and $fifo\_p$).
- *data* is the structure that carries the media data.

Data Structures. The state of a participant p is defined by three data structures: $VT(p)$, $CI(p)$ and $last\_fifo(p)$.

- $VT(p)$ is the vector clock established by Mattern (1989). The size of $VT(p)$ is equal to the number of participants in the group. Each participant p has an element $VT(p)[j]$, where j is a participant identifier.
- $CI(p)$ is a set composed by entries (k, t), which is a message identifier (the message diffused by the participant identifier k with the logical clock value t).
- $last\_fifo(p)$ has information about the last ($fifo\_p$) messages received by p. This structure is important because it represents potential causal messages.

Table 5: Vilain's point-interval relations and their logical mapping

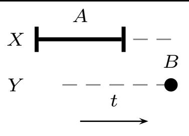
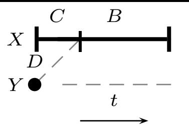
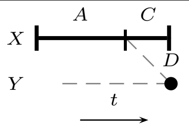
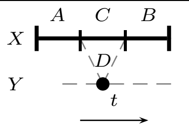
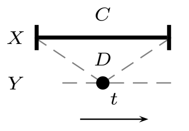
Vilain's Relations	**Logical Mapping**	**Scenario Example**	**Logical Mapping Expressed on Endpoints**
U before v / *v after U*	*precedes*: $A \rightarrow_I B$		$a^+ \rightarrow b$
U starts v	*starts*: $(C \mid\mid\mid D) \rightarrow_I B$		$c^- \parallel d$, $c^+ \rightarrow b^-$ $d \rightarrow b^-$
U finishes v	*ends*: $A \rightarrow_I (C \mid\mid\mid D)$		$a^+ \rightarrow c^-$, $a^+ \rightarrow d$ $c^- \parallel d$
v during U	*overlaps*: $A \rightarrow_I (C \mid\mid\mid D) \rightarrow_I B$		$a^+ \rightarrow c^-$, $a^+ \rightarrow d$ $c^- \parallel d$, $c^+ \parallel d$ $c^+ \rightarrow b^-$, $d \rightarrow b^-$
Not defined in Vilain's relations *U simultaneous v*	*simultaneous*: $C \mid\mid\mid D$		$c^- \parallel d$, $c^+ \parallel d$

Table 6: Synchronization Algorithm Part I

Initialization

```
1: procedure INITIALLY
2:     ♯define {$continuous$ | $discrete$}                    ▷ setup the media type handled
3:     ♯if defined ($continuous$)                     ▷ In order to take the adequate function
4:         send_continuous(input: $TP = \{begin \mid end \mid cut \mid fifo\_p\}$)
5:     ♯elif defined ($discrete$)
6:         send_discrete(input: $TP = discrete$)
7:     ♯endif
8:     $VT(p)[j] = 0 \quad \forall j : 1 \ldots n, CI(p) \leftarrow \emptyset, last\_fifo(p) \leftarrow \emptyset, Act = 0$
9: end procedure
```

5.1.1 Algorithm Description

Construction of logical mappings. In order to explain how our algorithm creates a logical mapping (Table 2), we take the pair (V, m_1) depicted in Figure 2 from the previous telemedicine scenario. The referenced lines correspond to the algorithm shown in Tables 6 to 9. The process of creating a logical mapping in our work is made by identifying the causal boundaries of the concerned segment(s) from left to right. In this example (See Figure 2), we first identify $Y = \{m_1\}$ and $X = V$. Then, we proceed to determine segment $A(V, m_1)$. To achieve this, we identify the left causal boundary a^- as equal to $x^- = x_1$, and the right causal boundary as equal to $a^+ = x_k$. The right endpoint a^+ is determined by the last *(fifo_p)* message received by participant i (Lines 42-44

Table 7: Synchronization Algorithm Part II

Send Continuous Media

```
10: For each continuous packet $m$ diffused by $p$ with participant identifier $i$
11: procedure SendContinuous(input: $TP = \{begin \mid end \mid cut \mid fifo\_p\}$)
12:     $VT(p)[i] = VT(p)[i] + 1$
13:     if Not($TP = fifo\_p$) then
14:         if Not ($TP = begin$) then        ▷ construction of the $H(m)$ for $end$ and $cut$ packets
15:             $H(m) \leftarrow CI(p)$
16:             if ($TP = end$) then        ▷ determines that participant $p$ is inactive
17:                 $Act = 0$
18:             end if
19:         else        ▷ construction of the $H(m)$ for $begin$ packets
20:             $Act = 1$        ▷ Determines that participant $p$ is sending an interval
21:             for all $(s, r) \in CI(p)$ do
22:                 if $\exists(x, f) \in last\_fifo(p) \mid s = x$ then        ▷ adds $fifo\_p$ packets to $CI(p)$
23:                     if Not($(s, r) = \max((x, f), (s, r))$) then
24:                         $CI(p) \leftarrow (CI(p) \setminus (s, r)) \cup (x, f)$        ▷ replaces $(s, r)$ from $CI(p)$ by $(x, f)$
25:                     end if
26:                 end if
27:             end for
28:             $H(m) \leftarrow CI(p)$
29:             $last\_fifo(p) \leftarrow \emptyset$
30:         end if
31:         $CI(p) \leftarrow \emptyset$        ▷ erases $CI(p)$ on each causal packet sent
32:     else        ▷ construction of the $H(m)$ for $fifo\_p$ packets
33:         $H(m) \leftarrow \emptyset$
34:     end if
35:     $m = (i, t = VT(p)[i], TP, H(m), data)$
36:     $sending(m)$
37: end procedure
```

Table 8: Synchronization Algorithm Part III

Send Discrete Media

```
38: For each discrete packet $m$ diffused by $p$ with participant identifier $i$
39: procedure SendDiscrete(input: $TP = discrete$)
40:     $VT(p)[i] = VT(p)[i] + 1$
41:     for all $(s, r) \in CI(p)$ do
42:         if $\exists(x, f) \in last\_fifo(p) \mid s = x$ then        ▷ adds $fifo\_p$ packets data to $CI(p)$
43:             if Not($(s, r) = \max((x, f), (s, r))$) then
44:                 $CI(p) \leftarrow (CI(p) \setminus (s, r)) \cup (x, f)$        ▷ replaces $(s, r)$ from $CI(p)$ by $(x, f)$
45:             end if
46:         end if
47:     end for
48:     $H(m) \leftarrow CI(p)$
49:     $last\_fifo(p) \leftarrow \emptyset$
50:     $CI(p) \leftarrow \emptyset$        ▷ erases $CI(p)$ on each causal packet sent
51:     $m = (i, t = VT(p)[i], TP, H(m), data)$
52:     $sending(m)$
53: end procedure
```

Table 9: Synchronization Algorithm Part IV

Reception process

```
54: For each packet $m$ received by participant $p$ with identifier $j$
55: procedure Receive($i \neq j, m = (i, t, TP, H(m), data)$)
56:     if ($t = VT(p)[i] + 1$) then                                    ▷ FIFO delivery condition
57:         if Not($TP = fifo\_p$) then
58:             if Not($t' \leq VT(p)[l]\ \forall (l, t') \in H(m)$) then      ▷ causal delivery condition
59:                 $wait()$
60:             else                                                    ▷ causal delivery procedure
61:                 $delivery(m)$
62:                 $VT(p)[i] = VT(p)[i] + 1$
63:                 if $\exists (s, r) \in CI(p) \mid i = s$ then
64:                     $CI(p) \leftarrow CI(p) \setminus (s, r)$                ▷ erases $(s, r)$ from $CI(p)$
65:                 end if
66:                 $CI(p) \leftarrow CI(p) \cup \{(i, t)\}$                        ▷ adds $(i, t)$ to $CI(p)$
67:                 for all $(l, t') \in H(m)$ do                     ▷ updates $CI(p)$ and $last\_fifo(p)$
68:                     if $\exists (s, r) \in CI(p) \mid l = s \land r \leq t'$ then
69:                         $CI(p) \leftarrow CI(p) \setminus (s, r)$
70:                     end if
71:                     if $\exists (x, f) \in last\_fifo(p) \mid l = x \land f \leq t'$ then
72:                         $last\_fifo(p) \leftarrow last\_fifo(p) \setminus (x, f)$
73:                     end if
74:                 end for
75:                 if $Act = 1$ and not ($TP = cut$) and not ($TP = begin$) then
76:                     SendContinuous ($cut$)                              ▷ sends a $cut$ packet
77:                 end if
78:             end if
79:         else                                                        ▷ FIFO delivery procedure
80:             $delivery(m)$
81:             $VT(p)[i] = VT(p)[i] + 1$
82:             if $\exists (x, f) \in last\_fifo(p) \mid i = x$ then
83:                 $last\_fifo(p) \leftarrow last\_fifo(p) \setminus (x, f)$
84:             end if
85:             $last\_fifo(p) \leftarrow last\_fifo(p) \cup (i, t)$      ▷ updates $last\_fifo(p)$ with a more
                                                                              recent packet
86:         end if
87:     else
88:         $wait\_FIFO()$
89:     end if
90: end procedure
```

Table 8) before the *discrete* send event ($send(m_1)$) (Lines 51-52, Table 8). Once we know the causal boundaries of $A(V, m_1)$, we determine the set of messages that compose it ($A = \{x_1, x_2, ..., x_k\}$). After interval segment $A(V, m_1)$ is identified, we proceed to recognize the causal boundaries of the interval segments $C(V, m_1)$ and $D(V, m_1)$. At this point, we can identify the left causal boundary $c^- = x_{k+1}$. With the discrete send event $send(m_1)$, we establish that $D(V, m_1) = \{m_1\}$. However, it is only until the delivery event of m_1 at j ($delivery(j, m_1)$) that we can identify the right endpoint of $C(V, m_1)$. At the reception of m_1 by participant j, our algorithm sends a *cut* message (Lines 75-76, Table 9) that establishes the endpoint of interval $C(V, m_1)$ ($c^+ = x_{k+l}$) and the beginning of interval $B(V, m_1)$ ($cut = b^- = x_{k+l+1}$). As a result, we have $C(V, m_1) = \{x_{k+1}, x_{k+2}, ..., x_{k+l}\}$. Finally, with the send event of x^+, we have $b^+ = x^+ = x_n$, and consequently, $B(V, m_1) = \{x_{k+l+1}, x_{k+l+2}, ..., x_n\}$.

For our purpose, interval $A(V, m_1)$ identifies the messages of V that must be executed before discrete point m_1. Interval $C(V, m_1)$ identifies the messages of V that are concurrent to m_1 and that can be executed in any order. Interval $B(V, m_1)$ identifies the messages of interval V that must be executed after message m_1. In other words, at a message level, the resulting logical mapping establishes that message m_1 will be executed by

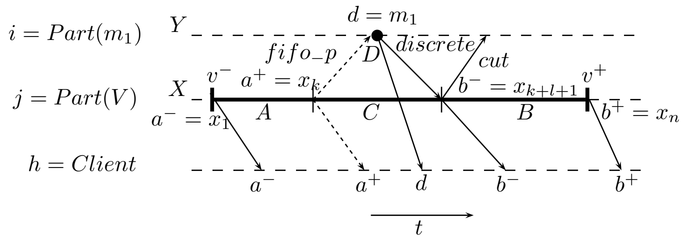


Figure 2: Logical mapping for the pair (V, m_1)

all participants after message $a^+ = x_k$ and before message $b^- = x_{k+l+1}$.

Performing causal order delivery. The resultant logical mapping for the example scenario is $A(V, m_1) \rightarrow_I (C(V, m_1)|||D(V, m_1)) \rightarrow_I B(V, m_1)$. To carry out the interval causal delivery at a $h = Client$ in terms of their endpoints we need to ensure that:

- $delivery(h, a^+) \rightarrow_{E'} delivery(h, c^-)$,
- $delivery(h, a^+) \rightarrow_{E'} delivery(h, d)$,
- $delivery(h, c^+) \rightarrow_{E'} delivery(h, b^-)$ and
- $delivery(h, d) \rightarrow_{E'} delivery(h, b^-)$

Since $a^+ = x_k$, $c^- = x_{k+1}$, $c^+ = x_{k+l}$ and $b^- = x_{k+l+1}$, the procedure of $delivery(h, a^+) \rightarrow_{E'} delivery(h, c^-)$ and $delivery(h, c^+) \rightarrow_{E'} delivery(h, b^-)$ is accomplished by the FIFO property implemented by lines 56 and 81 of Table 9. The procedure of $delivery(h, a^+) \rightarrow_{E'} delivery(h, d)$ is achieved in the following way. Initially, message $a^+ = x_k$ is sent as $fifo\_p$. To consider it as a causal message, participant j includes information concerning x_k in its causal history (Lines 42-44 Table 8), and attaches this information to structure $H(m_1)$ of the message $d = m_1$ before its send event (Line 48, Table 8). The causal delivery condition (Line 58) detects that the interval $D(V, m_1)$ ($D(V, m_1) = \{m_1\}$) must be delivered after the delivery at h of x_k, and the FIFO condition (Line 56, Table 9) establishes that x_k must be delivered after the delivery of messages $x_{k'} \in A(V, m_1) \subseteq X$, such that $k' < k$. For the requirement of $delivery(h, d) \rightarrow_{E'} delivery(h, b^-)$, message $b^- = x_{k+l+1}$ has attached information on its structure $H(b^-)$ about the message m_1 (Lines 14-15, Table 7). The causal delivery condition (Line 58, Table 9) establishes that b^- should be delivered at h after the delivery of m_1.

5.2 Correction of the Synchronization Error

The main objective of the present section is to show how it is possible to reduce the synchronization error among the media data by using the resulting logical mapping specification of a temporal scenario. First of all, we note that we work only with the interval endpoints of the logical mapping specification. We take the causal messages (endpoints) as synchronization points. Each time that an endpoint is received by a participant, we take two corrective actions. First, we delay the delivery of messages when they do not satisfy the causal dependencies, and secondly, we discard messages when the maximum waiting time (Δ_{il}) is exceeded. In our case, we establish the value of Δ_{il} according to the maximum synchronization error tolerated between the type of media involved (Hac and Xue, 1997).

Waiting time period. In order to determine how much time a message m must wait from its reception to its delivery, we define the following function:

$$\begin{aligned}
&wait(m) : i = Part(m) \\
&\{\ \forall m' = (l, t') \in H(m), \\
&\quad a) t' \leq VT(p)[l] \text{ or} \\
&\quad b) receive\_time_p(m) + remainder\_time_{m'}(i, l) \\
&\qquad\qquad \leq current\_time(p)\ \}
\end{aligned}$$

Where $receive\_time_p(m)$ returns the reception time of m at participant p, the function $current\_time(p)$ has the existing time at p and the $remainder\_time_{m'}(i, l)$ returns the possible wait time for the reception of m'. In general, the function $remainder\_time_{m'}(i, l)$ for a message m' is defined as follows:

$$remainder\_time_{m'}(i, l) =$$

$$\begin{cases} \Delta_{il} - (last\_rcv_p(i) - last\_rcv_p(l)), & \text{if } > 0, \\ 0 & \text{otherwise.} \end{cases}$$

The Δ_{il} is the maximum waiting time established between the media data with sources i and l, and the $last\_rcv_p(k)$ gives the time when the participant p has received the last message by a participant k.

Therefore, the $wait(m)$ function establishes that a message m sent by the participant i ($i = Part(m)$) and received by a participant p will be immediately delivered after each message $m' = (l, t')$ that belongs to $H(m)$ has been either delivered at p (first condition) or the maximum waiting time Δ_{il} has elapsed (second condition). If the second condition is satisfied, then message m' is discarded by participant p, and every message $m'' = (l, t'')$ such that $t'' > VT(p)[l]$ and $t'' < t'$ is also discarded .

Once the function $wait(m)$ is defined, it can be inserted at line 59 of the algorithm (Table 9). In Figure 3, we show an example of the behavior of the $wait(m)$ function. In this case, the delivery of message m_1 is delayed until the delivery of a^+ since a^+ immediate precedes the sending of m_1 and consequently, it belongs to $H(m_1)$. Since a^+ has not been delivered to p, vector $VT(p)$ does not satisfy the first condition of the $wait()$ funtion. In this scenario, message a^+ is delivered because it arrives before the expiration of the maximum waiting time Δ_{ij}; otherwise, this message will be discarded.

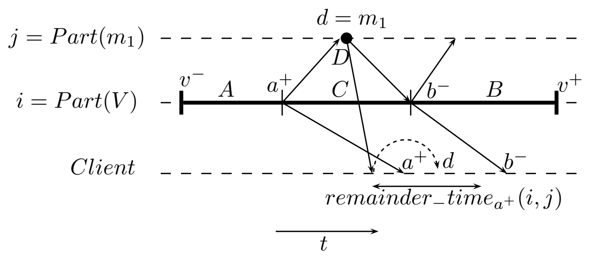


Figure 3: Example of synchronization error correction

5.3 Measuring the Synchronization Error

In order to evaluate the impact of the synchronization mechanism proposed in this paper, we define two equations, the $rcv\_synch\_error_p(m)$ and $dlv\_synch\_error_p(m)$. The function $rcv\_synch\_error_p(m)$ estimates the synchronization error by p at the reception of a causal message m, and $dlv\_synch\_error_p(m)$ estimates the synchronization error at the moment of the delivery of m, which is measured after applying the correction mechanism.

The $rcv\_synch\_error_p(m)$ is defined as follows:

$$rcv\_synch\_error_p(m) = \frac{\sum_{(l,t')\in H(m)} receive\_time_p(m) - last\_rcv_p(l)}{|H(m)|} \quad (1)$$

The $dlv\_synch\_error_p(m)$ is defined as follows:

$$dlv\_synch\_error_p(m) = \frac{\sum_{(l,t')\in H(m)} delivery\_time_p(m) - last\_dlv_p(l)}{|H(m)|} \quad (2)$$

Where $delivery\_time_p(m)$ returns the delivery time of m at participant p, and $last\_dlv_p(l)$ gives the time when participant p has delivered the last message saw from participant l.

6 EMULATION RESULTS

We have tested our synchronization mechanism considering some WAN network characteristics such as transmission delay and partial order delivery. To emulate a WAN network behavior, we use the NIST Net emulation tool (Carson and Santay, 2003). In order emphasize the potential situation of unphased media data, we construct an emulation environment, see Figure 4, that consists of a group of four distributed hosts, three of them transmitting live multimedia data, while the other functions only as *Client* (Host Z). Each host has two input and one output communication channels. The sending hosts only transmit one media (audio, video or images), see Figure 4. For the audio and video, we use MPEG-4 encoders of the MPEG4IP project (Mackie et al., 2000). For the case of audio, the silence compression MPEG-4 CELP mode is used. In this mode, we send a *begin* message each time that the `TX_Flag` equals to 1 (indicates voice activity), and a frame is send as an *end* message each time that `TX_Flag` $\neq 1$ (indicates no or low voice activity). For the remaining audio frames, we send them as $fifo\_p$ messages. For the case of video, the video was encoded as a single video object into a single video object layer with a rate of 25 frames/s, the group of pictures (GoP) pattern is set to IBBPBBPBBPBB. The I frames are sent as *begin* messages, and the last B frames of the patterns are send as *end* messages. The rest of the video frames are sent as $fifo\_p$ messages. To emulate the sending of images we randomly generate and transmit discrete events, approximately each 1200 ms based on a random variable with normal distribution. These data are sent as *discrete* messages. In this scenario, each host has the synchronization mechanism running and only the *Client* measures the synchronization error by using equations 1 and 2. For simplicity, we consider only one maximum waiting time for every pair of media data $\Delta = 240$ms, which is the maximum synchronization error established for image-audio in real time in Hac and Xue (1997).

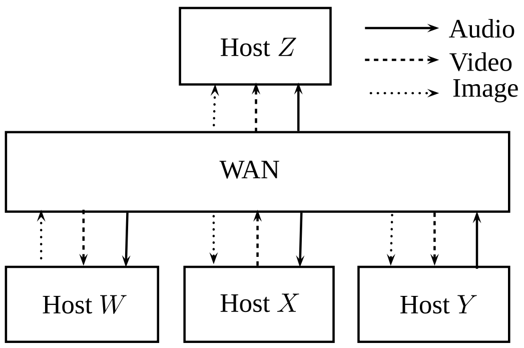


Figure 4: Emulation architecture

With this configuration, we present two tests; one with optimal conditions, which means that the transmission delay considered among the media data does not surpass the synchronization error tolerate; while in the other, the transmission delay can be greater than the the synchronization error, resulting in not accomplishing of the required QoS. The traffic conditions are presented in Table 10.

Test 1. Under optimal conditions, we can see in Figure 5 that the estimated error at the reception time remains acceptable; nevertheless, our mechanism, at the moment of the media delivery, reduces the synchronization error that in this case is close to zero during almost the entire transmission.

Test 2. For the second test, we can see in Figure 6 that at the reception time, the synchronization error surpasses in several points the maximum error tolerated. This effect can produce, at application level, the loss of coherency among the media played. In this case, after applying our mechanism, the error is decreased, making it acceptable in nearly all cases.

Table 10: Transmission delay established for emulation

Data Type	**Source**	**Target**	**Test 1 (ms)**	**Test 2 (ms)**
Video/Audio	Host X/Y	Host Z	300 ± 120	300 ± 180
Still images	Host W	Host Z	200 ± 50	200 ± 100

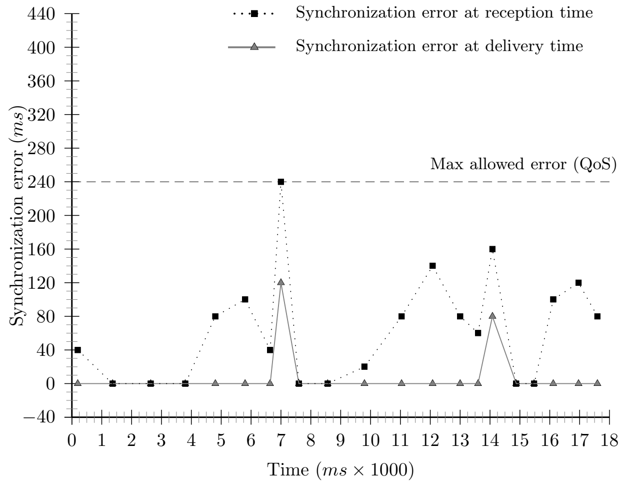


Figure 5: Test 1 graphics results: Audio and video (300 ± 120 ms); Still Images (200 ± 50 ms)

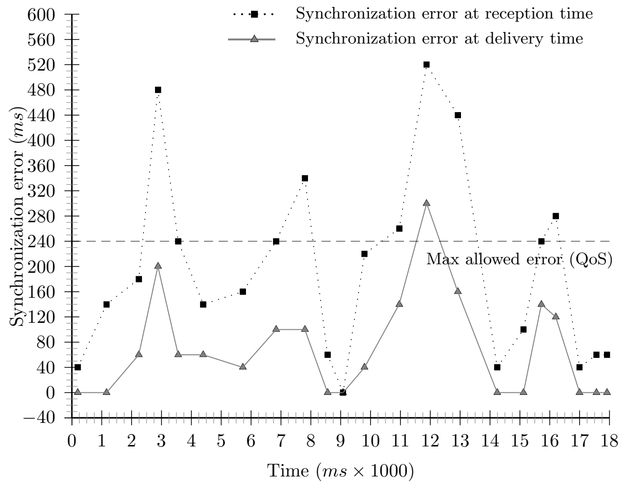


Figure 6: Test 2 graphics results: Audio and video (300 ± 180 ms); Image (200 ± 100 ms)

7 CONCLUSIONS

We have presented an intermedia synchronization mechanism that mainly focuses on distributed multimedia data. The mechanism addresses the problem of satisfying any temporal dependencies for continuous-continuous, discrete-continuous, and discrete-discrete relations. Its core is the use of logical mappings, which specify temporal relations without using global references based on the causal dependencies of the media involved. In order to be efficient, the proposed mechanism uses the specification of logical mappings based on the endpoints. In our work, these endpoints are sent as causal messages which determine synchronization points for any reception process. To demonstrate the viability and effectiveness of the synchronization mechanism, we carry out the emulation of the mechanism considering some aspects of a WAN network environment and using MPEG-4 encoders. The emulation results show the benefits of our mechanism by reducing in all cases the synchronization error of the system.

We note that further work is needed in order to consider more network and media conditions, such as loss of messages and intra-stream lifetime constraints. Our attention is focused in this direction, and we expect to have some interesting contributions shortly.

REFERENCES

Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM 26,* 11, 832–843.

Balaouras, P., Stavrakakis, I., and Merakos, L. 2000. Potential and limitations of a teleteaching environment based on h.323. *Audio-Visual. Communication Systems, Computer Networks 34,* 6, 945–958.

Bechhofer S., et al., E. W. W. W. C. 2006. Time ontology in owl. http://www.w3.org/TR/2006/WD-owl-time-20060927/.

Bulterman D., et al., E. W. W. W. C. 2008. Synchronized multimedia integration language (smil 3.0). http://www.w3.org/TR/2007/WD-SMIL3-20070713/.

Carson, M. and Santay, D. 2003. Nist net - a linux-based network emulation tool. *ACM SIGCOMM Computer Communication Review 33,* 3 (June), 111–126.

Fanchon, J., Drira, K., and Pomares, S. E. 2004. Abstract channels as connectors for software components in group communication services. In *Fifth Mexican International Conference on Computer Science (ENC'04)*, IEEE, Ed. Mexico, 20–24.

Geyer, W., Bernhardt, C., and Biersack, E. 1996. A synchronization scheme for stored multimedia streams. In *IDMS'96, European Workshop on Interactive Distributed Multimedia Systems and Services*, Heidelberg, Ed. Springer-Verlag, 277–295.

Hac, A. and Xue, C. X. 1997. Synchronization in multimedia data retrieval. *International Journal of Network Management 7*, 33–62.

Ishibashi, Y., Tasaka, S., and Tachibana, Y. 1999. A media synchronization scheme with causality control in network environment. In *IEEE LCN'99*, IEEE, Ed. 232–241.

Lamport, L. 1978. Time clocks and the ordering of events in a distributed system. *Communications of the ACM 21,* 7, 558–565.

Lamport, L. 1986. On interprocess communications: I. basic formalism. *Distributed Computing 1,* 2, 77–85.

Little, T. and Ghafoor, A. 1990. Synchonization and storage models for multimedia objects. *IEEE Journal on Selected Areas in Communications 8,* 3 (April), 413–427.

Mackie, D., May, B., and Franquet, A. M. 2000. Mpeg4ip open source project. http://mpeg4ip.sourceforge.net.

Mattern, F. 1989. Virtual time and global states of distributed systems. In *Proc. International Workshop on Parallel and Distributed Algorithms*, R. Y. Q. P. Cosnard, M. and M. Raynal, Eds. Elsevier, North-Holland, 215–226.

Morales Rosales, L. A. and Pomares Hernandez, S. E. 2006. A temporal synchronization mechanism for real-time distributed continuous media. In *International Conference on Signal Processing and Multimedia Applications (SIGMAP).* Setubal, Portugal, 302–309.

Perkins, C. 2003. *RTP Audio and Video for Internet.* Addison Wesley, USA.

Plesca, C., Grigoras, R., Quinnec, P., and Padiou, G. 2005. Streaming with causality: a practical approach. In *MULTIMEDIA'05: Proceeding of the 13th annual ACM International Conference on Multimedia*, ACM, Ed. Hilton, Singapore, 283–286.

Shimamura, K., Tanaka, K., and Takizawa, M. 2001. Group communication protocol for multimedia applications. In *IEEE ICCNMC'01*, IEEE, Ed. 303.

Vilain, M. 1982. A system for reasoning about time. In *2nd (US) National Conference on Artificial Intelligence (AAAI-82).* Pittsburgh, PA, USA, 197–201.