Broadcast with Time and Causality Constraints for Multimedia Applications

Roberto Baldoni, Ravi Prakash, Michel Raynal, Mukesh Singhal

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Broadcast with Time and Causality Constraints for Multimedia Applications

R. Baldoni , R. Prakash , M. Raynal , M. Singhal

N ° 2976

Septembre 1996

THÈME 1

ISSN 0249-6399

INRIA
RENNES

Broadcast with Time and Causality Constraints for Multimedia Applications

R. Baldoni*, R. Prakash†, M. Raynal*, M. Singhal†

Thème 1 — Réseaux et systèmes
Projet Adp

Rapport de recherche n ° 2976 — Septembre 1996 — 12 pages

Abstract: Δ-causal ordering is a communication abstraction designed for distributed applications whose messages (i) have to be delivered according to causal ordering and (ii) have a limited lifetime after which their data can no longer be used by the application. Example of such applications are: *multimedia real-time collaborative applications* and *groupware real-time applications*. For such applications, the broadcasting of information is of primary importance. In this paper, we propose a simple and efficient Δ-causal ordering protocol in the context of broadcast communication. By taking into account transitive dependencies on message sends, this algorithm gets a significant reduction in the control information piggybacked on application messages, compared to previous algorithms.

Key-words: distributed computing, causal ordering, multimedia, real-time communication.

(Résumé : tsvp)

Appeared in the Proceedings of the 22nd EUROMICRO Conference, Pragua, September 1996.

* IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France. {baldoni,raynal}@irisa.fr
† Department of Computer and Information Science, The Ohio State University, Columbus, Ohio, 43210. {prakash,singhal}@cis.ohio-state.edu. Mukesh Singhal was supported by an INRIA grant during a stay at IRISA.

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : (33) 99 84 71 00 – Télécopie : (33) 99 84 71 71

Diffusion pour applications multimedia avec des contraintes de temps et de causalité

Résumé : La notion de Δ-causalité a été introduite pour faciliter la conception d'applications réparties dans lesquelles (1) les messages ont une durée de vie (Δ) après laquelle leur contenu n'a plus d'utilité pour le destinataire et (2) leurs livraisons doivent respecter l'ordre causal. Dans cet article, nous proposons un protocole simple de diffusion qui respecte ces deux contraintes.

Mots-clé : Applications multimedia, calculs répartis, communication temps-réel, ordre causal

1 Introduction

The notion of Δ-*causal ordering* was introduced in [Yav92], and later refined and formalized in [BMR95]. This communication mode has been defined in order to reduce the *asynchrony* of communication channels in distributed systems with the following characteristics: (i) messages can be lost and (ii) messages have a *lifetime*, Δ, after which their data can no longer be used. In other words, messages whose transmission delays are greater than Δ are considered to be lost. Δ-causal order strives to deliver as many messages as possible before their deadlines in such a way that these deliveries respect causal order. Causal ordering means that if two message sends are causally related [Lam78] and have the same destination, then the corresponding messages are delivered in their sending order. Indeed, when we consider a reliable distributed system and messages whose contents have no delivery constraints, Δ-causal ordering boils down to the original definition of causal ordering given by Birman and Joseph in [BJ87].

In the context of broadcast communication, the Δ-causal ordering abstraction matches the requirements of an important class of multimedia applications, namely, distributed multimedia real-time applications, where participants are required to exchange real-time *audio* and *video* information over a communication network. This flow of information must preserve the causal dependency even though part of the information can be lost or can be discarded when it violates the timing constraints imposed by a real-time interaction.

Several protocols implementing pure causal ordering [BJ87, RST91] and Δ-causal ordering [BMR95, AS95] appeared in the literature; however, they suffer from the typical pitfall of the timestamping (logical or physical) technique: to ensure causal order, in the context of broadcasting, messages have to carry a vector of integers whose dimension is given by the number of processes [Mat89].

In this paper we introduce a new and efficient Δ-causal ordering protocol, reducing the amount of information carried by messages. Only in the worst case, this information has the same size as previous algorithms. This reduction is obtained by taking into account transitive dependencies on the message sends.

The paper is structured into two main sections. Section 2 points out the problem of messages deliveries in multimedia real-time applications, presents the model of asynchronous distributed executions, and recalls the formal definition of Δ-causal order. Section 3 presents the Δ-causal ordering protocol with its proof of correctness.

2 Δ-Causal Ordering

2.1 Distributed System

A distributed program is a finite set P of n sequential processes $\{P_1, P_2, \ldots, P_n\}$ that communicate and synchronize only by exchanging messages. The underlying system, on which distributed programs execute, is composed of n processors (for simplicity, we assume one process per processor) that can exchange messages. When a message arrives on a channel, it can be delivered as soon as the delivery condition becomes true; in an asynchronous system with no special constraints on deliveries[1] this condition is always true. We assume that each pair of processes is connected by an asynchronous and unreliable logical channel (messages can be lost and transmission delays of non lost messages are unpredictable). Processors do not have a shared memory, there is no bound on their relative speeds, and there are no hidden channels.

[1] Examples of special constraints in deliveries are fifo order, Causal Order and Δ-causal order.

2.2 Distributed Executions

At the application level, execution of a process produces a sequence of events which can be classified as: *send* events, *delivery* events, and *internal* events. An internal event may change only local variables; send or delivery events involve communication. The causal ordering of events in a distributed execution is based on Lamport's *happened-before* relation [Lam78] denoted by $\rightarrow$. If a and b are two events then $a \rightarrow b$ iff one of these conditions is true:

i. a and b occur at the same process with a before b

ii. $a = send(m)$ is the send event of a message m and $b = delivery(m)$ is the delivery event of the same message

iii. there exists an event c such that $a \rightarrow c$ and $c \rightarrow b$.

Such a relation allows us to represent a distributed executions as a partial order of events, called $\widehat{E} = (E, \rightarrow)$ where E is the set of all events. Hereafter, we call $M_{\widehat{E}}$ the set of all messages exchanged in $\widehat{E}$ and we do not consider internal events since they do not affect the causal ordering of events.

2.3 The Lifetime of a Message

The lifetime, Δ, of a message is the physical time duration, after its sending time, during which its content is meaningful and consequently can be used by its destination process(es). A message that arrives at its destination after its lifetime is useless and must be discarded; as far as the destination process is concerned, the message is lost. A message that arrives at its destination within its lifetime must be delivered at the target process within the expiration of its lifetime. For simplicity, we assume an identical lifetime for all the messages (in case of distinct types of messages with different lifetimes, we can assume the minimum among them as the lifetime to be respected by all the messages). As indicated in the introduction, for a message m the instant "$sending\ time\ of\ m + \Delta$" is the deadline after which this message is useless for the destination process.

In multimedia systems, the lifetime of a message is the maximum delay, after its sending, that can be tolerated before delivering it without degrading the quality of the service. This delay is dependent on the type of multimedia applications. For example, in teleconferencing, the maximum lifetime for audio and video messages is 250 msec.[JSS94] (an acceptable one is 100 msec. [RB93]), videophones systems may tolerate a lifetime of 350 msec., non-interactive video systems have a lifetime of 1 sec. and catalog browsing services have a lifetime of 2 sec. [RB93, Fer90].

2.4 Global Clock

In order to enforce the real-time delivery constraints, processes are endowed with a global clock value whose drift with respect to physical time is bounded, and whose granularity and precision [KO87, Ver94] are such that all the causally dependent events are produced at different times. Many clock synchronization protocols have been described in the literature. Some currently used protocols provide a global clock synchronization that bounds the clock drift value, ϵ, within $5 - 10$ milliseconds [GZ89]. Without loss of generality, we assume that ϵ is incorporated in the lifetime Δ of the message. The relation between the value Δ and the granularity and precision of a global clock is thoroughly discussed in [BMR95].

2.5 Definition

The notion of Δ-causal ordering was introduced in [Yav92]; and was later formalized in [BMR95]. This notion is very general as it assumes an unreliable communication system where messages contain

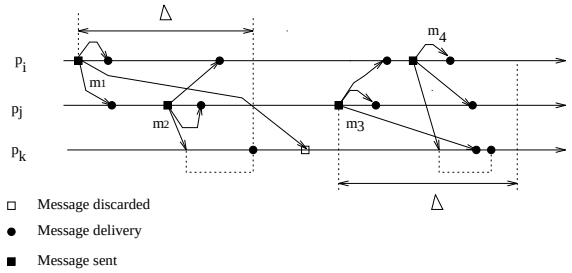

Figure 1: Deliveries of broadcast communications respecting Δ-causal ordering

real-time data. That is, messages can be discarded if they arrive at their destinations when their data are obsolete (after the expiration of their deadlines).

Definition [Δ-Causal Ordering]: *A distributed computation $\widehat{E}$ respects Δ-causal ordering if:*

i. All messages in $M(\widehat{E})$ that arrive within Δ are delivered within Δ, all the others are never delivered (they are lost or discarded);

ii. All delivery events respect causal ordering, i.e.,
For any two messages $m1$ and $m2 \in M_{\widehat{E}}$ that arrive within Δ we have: **if** $send(m1) \rightarrow send(m2)$ **and** *$m1$, $m2$ have the same destination process,* **then** $deliver(m1) \rightarrow deliver(m2)$.

Note that Δ-causal ordering meets the original causal ordering definition [BJ87] when channels are reliable (all the messages will be eventually delivered) and Δ is greater than any message transmission delay over the distributed system (in this case the content of messages has actually no real-time characteristics). In the context of broadcast communications, an example of deliveries respecting Δ-causal ordering is shown in Figure 1.

2.6 A Class of Multimedia Real-Time Applications

Multimedia real-time collaborative applications or *groupware real-time applications* such as teleconferencing, shared window systems, group word-processors, videophones systems, require participants to exchange real-time *audio* and *video* information over a communication network. For example in a teleconferencing the audio and video signal of each participant should be received by all the others. To preserve the real-time interaction of such applications, the audio and video information have a maximum transmission delay. Information arriving at the destination site with greater delays must be discarded since their use may seriously degrade the quality of the real-time interaction [Wak93, Fer90, HB94].

Consider as an example an analog video signal. The video is sampled at the source site and such samples produce a *continuous* flow of information over the communication network. In order to reconstruct a high quality analog signal at the destination site, as many samples as possible must be delivered within the maximum transmission delay, and samples arriving within the maximum transmission delay should be delivered in the order they were sent. Deliveries of samples out of order, or with higher delay than the maximum could greatly deteriorate the quality of the analog signal at the destination site (for example, the audio signal is very sensitive to out of order delivery of samples) [HB94, Yav92].

Such signals may however tolerate a loss of information as long as these losses do not cause a major degradation of analog signals (making them unintelligible) at the destination site [Wak93, HB94, Fer90]. Another important property is that video and audio signals do not impose uniformity upon the reception of samples, i.e., a sample can be received by only a subset of all the participants. Such a property results in a different quality of the audio and video signal for each participant. Such a

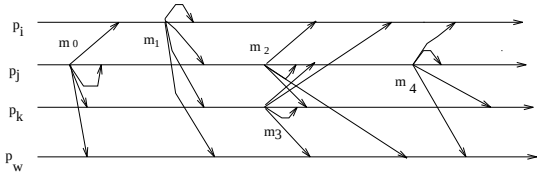

Figure 2: Example of a computation

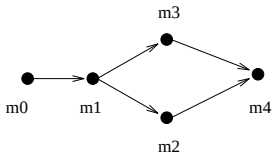

Figure 3: Message causality graph of the computation of Figure 2

situation does not deteriorate the conference interaction as long as participants receive an intelligible signal.

The notion of Δ–causal ordering is well suited for such a class of applications.

3 An Efficient Δ-Causal Broadcasting Protocol

An implementation of Δ-causal ordering consists of implementing a protocol over the original underlying system, such that events of distributed computations can be Δ-causally ordered at the application level. At the underlying system level, three types of events are associated with each message: *send*, *arrival*, and *delivery* (only *send* and *delivery* events are visible to processes). Due to delivery constraints imposed by Δ-causal ordering, when a message arrives either it is discarded (if it has missed its deadline) or it can be delayed until its delivery condition becomes true (if the message arrives within its deadline but some, not yet received, messages causally precede it and their deadlines have not yet expired).

The core of the protocol then consists in defining the delivery condition DC(m) associated with each broadcast message m in such a way that m *is delivered as soon as possible*, i.e., when all broadcast messages that causally precede m have either been delivered (for example, the delivery of message $m4$ in process P_i depicted in Figure 1) or have been lost or they are still in transit when their deadlines expire (for example, the delivery of message $m2$ in process P_k depicted in Figure 1). Next, some notions and variables are introduced.

3.1 Message Causality Graph

The message causality graph of a distributed computation $\widehat{E} = (E, \rightarrow)$ is a directed acyclic graph whose vertices are all messages belonging to $M_{\widehat{E}}$, such that there is a direct edge from m_1 to m_2 (denoted $m_1 \rightsquigarrow m_2$) if (i) $send(m_1) \rightarrow send(m_2)$ and (ii) $\not\exists m \ : \ send(m_1) \rightarrow send(m) \ and \ send(m) \rightarrow send(m_2)$. In the following we say that m_1 is an *immediate predecessor* of m_2. It is to be noted that a message m can have at most $n-1$ immediate predecessors, one from each process. The causality graph associated with the computation of Figure 2 is shown in Figure 3.

Let $\stackrel{+}{\rightsquigarrow}$ denote the transitive closure of $\rightsquigarrow$. Two messages m_1 and m_2 are said to be *concurrent* if neither of them is a predecessor of the other in the message causality graph, *i.e.*, $\neg(m_1 \stackrel{+}{\rightsquigarrow} m_2)$ and $\neg(m_2 \stackrel{+}{\rightsquigarrow} m_1)$. The sendings of concurrent messages constitute independent events.

3.2 Data Structures

Processes have access to a variable *current_time* of type *time* (whose domain is the set of non-negative integers) that is continuously updated by an underlying physical clock synchronization protocol. This variable represents the global physical time as perceived by processes; *current_time* is used to timestamp messages.

Moreover, each process P_i manages (i) a local variable $sent\_time_i$ in which it stores the time of its last broadcasting and (ii) an array $DEL_i : array[1..n]\ of\ time$ whose meaning is as follows: $DEL_i[j] = d \Leftrightarrow$ the last message broadcast from P_j and delivered to P_i was sent at the global time d.

Each entry of this array is initialized to a value lower than the initial value of the variable $current\_time$.

3.3 Delivery Condition

In order to express the delivery condition $DC(m)$ each message m has to carry some control information that will delay its delivery until all messages that precede it in the message causality graph have either been delivered or have missed their deadlines. In what follows, first we address the problem to ensure correct causal ordering for message delivery by exploiting the message causality graph, and then we solve the problems due to the lifetime of messages.

Tracking direct dependencies to ensure causal order

Figure 3 shows that the delivery of m_4 cannot occur before the delivery of m_0, m_1, m_2, and m_3. But, due to transitive dependencies, the delivery of m_4 can depend only on its direct predecessors m_2 and m_3 as these two messages can only be delivered after m_1 which, in turn, can be delivered only after the deliver of m_o. This simple observation allows us to reduce the control information piggybacked on messages as shown in the next paragraph.

Each message m, uniquely identified from a pair $(sender\_identity, sending\_time)$, called *timestamp*, carries only the timestamps of its immediate predecessors in the message causality graph. This set of timestamps constitutes the *causal barrier* (CB_m) of message m [PRS95]. As indicated in Section 3.1, this set has at most $n-1$ elements. The fewer the number of immediate concurrent predecessors m, the smaller the number of *timestamps* in CB_m.

So, for a destination process P_i, if we consider only causal deliveries, the delivery constraint $DC(m)$ associated with m is:

$$\forall (k,d) \in CB_m \ : \ (d \leq DEL_i[k])$$

which expresses all immediate predecessors of m in the message causality graph have been delivered to process P_i.

Taking lifetime into account

A message m is discarded if it misses its deadline, i.e., if:

$(send\_time_m + \Delta < current\_time)$

By definition, all messages arriving at their destination(s) within their deadlines must be delivered. So a message m is delivered as soon as every message contained in its causal barrier CB_m either has been delivered (part C1 of $DC(m)$), or missed its deadline (part C2 of $DC(m)$).

Hence, the delivery constraint $DC(m)$ associated with m is:

$$\forall (k,d) \in CB_m : \begin{array}{ll} d \leq DEL_i[k] & (C1) \\ or & \\ d + \Delta < current\_time & (C2) \end{array}$$

```
 procedure SEND(m, i):  % m is the message, P_i is the sender %
begin    % this procedure is executed atomically %
     send_time_i := current_time;                                          (S1)
     for each x ∈ P do send (m, send_time_i, CB_i) to P_x od                (S2)
     CB_i := (i, send_time_i);                                             (S3)
end.

when (m, send_time_m, CB_m) arrives at P_i from P_j:
begin
     if send_time_m + Δ < current_time                                     (R1)
     then discard(m);                                                      (R2)
     else                                                                  (R3)
          wait (∀(k, d) ∈ CB_m :                                           (R4)
                  (d ≤ DEL_i[k])                                           (R5)
                ∨ (d < current_time − Δ));                                 (R6)
          DEL_i[j] := send_time_m;                                         (R7)
          CB_i := CB_i − CB_m ∪ {(j, send_time_m)};                        (R8)
          deliver(m)                                                       (R9)
     fi                                                                   (R10)
end.
```

Figure 4: the Δ-causal broadcasting protocol

The notion of *causal barrier* causes the addition of a new data structure managed by each process P_i: a set CB_i of message timestamps. This set is initially empty and its size varies between zero and $n-1$.

3.4 The Algorithm

The Δ-causal ordering protocol is described in Figure 4. When broadcasting a message m, process P_i sends m with the current value of the causal barrier CB_i and the time of the sending of m (line S2). Moreover, if P_i sends another message m', the delivery of m' will only be constrained by the delivery of m, to the current knowledge of P_i. So P_i resets CB_i to the timestamp of m (line S3)[2].

Upon its arrival, a message m will be immediately discarded if it misses its deadline (lines R1 and R2). If m is not discarded, its delivery is determined by the *delivery condition*. Message m is delivered as soon as the predicate DC(m) is true (R4–R6). When a broadcast message m, sent by P_j, is delivered to P_i, the state variables DEL_i (lines R7) and CB_i (lines R8) are updated according to their definitions. In particular, the first part of line R8 (namely, $CB_i := CB_i - CB_m$) removes from the causal barrier CB_i, all immediate predecessors of message m (they are contained in CB_m), since as m is delivered, all immediate predecessors of m either were delivered or missed their deadlines. Then, the second part of line R8 (namely, $CB_i := CB_i \cup \{(j, send\_time_m)\}$) adds m's timestamp to CB_i in order to constraint the delivery of messages that will be immediate successors of m, in the message causality graph, to occur after the delivery of m.

Note that if we assume reliable channels and Δ greater than any message transmission delay over the network, we no longer have real-time delivery constraints and then by suppressing lines (R1–R3,

[2] If, before sending such a message m', P_i is delivered messages, CB_i will be updated accordingly; see line R8 and its explanation in the next paragraph.

R6 and R10) and by replacing physical time with a logical one (line S1: $send\_time_i := send\_time_i + 1$), we get a protocol for asynchronous causal broadcasting that compares favorably with previous pure causal ordering protocols [BJ87, RST91].

3.5 Correctness Proof

3.6 Liveness Property

Before proving the *liveness property*, let us note that because of the definition of the causal barrier sets CB, and the continuous progress of the variable $current\_time$, the following relation holds for each message m:

$$\forall\ (k, d) \in CB_m :: send\_time_m > d \qquad \text{(P)}$$

Theorem 1 (Liveness) *(i) All messages arriving within their deadlines will be delivered within their deadlines and (ii) all messages arriving after the expiration of their deadlines will be discarded.*

Proof Point (ii) follows from the test (line R1) of the protocol of Figure 4. Point (i) is proved by contradiction. Suppose that there exists a message m that arrived within its deadline and has not been delivered within its deadline. Hence, on its deadline, from the delivery condition DC(m) (lines R4–R6), the following condition NDC follows:

$$\exists (k, d) \in CB_m :$$
$$(d > DEL_i[k]) \wedge (d \geq current\_time - \Delta)$$

On the deadline of message m we have: $current\_time = send\_time_m + \Delta$. So the second term of NDC becomes: $\exists (k, d) \in CB_m : (d \geq send\_time_m)$. This contradicts property P.

It follows that at the deadline of an arrived message m, NDC is false contradicting our initial assumption. Therefore, for any message m, arrived before its deadline, the delivery condition DC(m) will be verified before its deadline expires. •

3.7 Safety Property

Lemma 1
Let m_0, timestamped (k_0, d_0), and m_x be two messages such that m_0 is an immediate predecessor of m_x in the message causality graph $m_0 \rightsquigarrow m_x$ (i.e., $send(m_0) \rightarrow send(m_x)$ and $\nexists m : send(m_0) \rightarrow send(m)$ and $send(m) \rightarrow send(m_x)$). Then $(k_0, d_0) \in CB_{m_x}$.

Proof Let P_i be the sender of m_x. As m_0 is an immediate predecessor of m_x ($m_0 \rightsquigarrow m_x$), two cases have to be considered (Figure 5): before it sent m_x, either P_i has sent m_0 (case S), or P_i has been delivered m_0 (case D). In both cases P_i updates CB_i (at line S3 in case S, and at line R8 in case D), which now includes (k_0, d_0). We show that, as m_0 is an immediate predecessor of m_x in the message causality graph, P_i does not suppress (k_0, d_0) from CB_i until m_x is sent (and consequently (k_0, d_0) will belong to CB_{m_x}).

First, between the sending (case S) or the delivery (case D), of m_0 and the sending of m_x, P_i does not send any message (otherwise, m_0 would not be an immediate predecessor of m_x, contradicting our hypothesis).

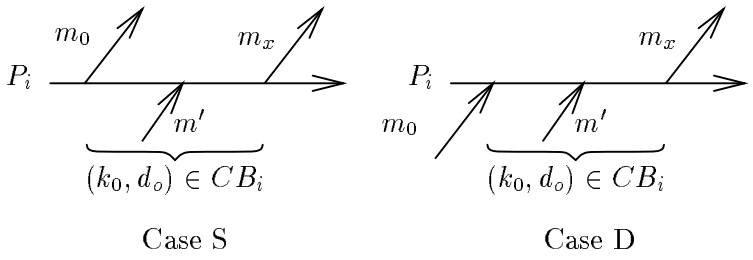

Figure 5: Update of CB_i

Suppose between the sending (case *S*) or the delivery (case *D*), of m_0 and the sending of m_x, P_i is delivered a message m'. Three cases have to be considered.
case i). If m_0 and m' are concurrent in the message causality graph, then $(k_0, d_0) \notin CB_{m'}$. It follows that when m' is delivered at P_i, (k_0, d_0) cannot be suppressed from CB_i at line R8.
case ii). m_0 and m' are not concurrent and $m_0 \overset{+}{\rightsquigarrow} m'$ (*i.e.,* m_0 is a predecessor of m' in the message causality graph). As m' is delivered to P_i before m_x is sent, it follows that $m_0 \overset{+}{\rightsquigarrow} m' \rightsquigarrow m_x$ and so, m_0 is not an immediate predecessor of m_x in the message causality graph, contradicting our hypothesis.
case iii). m_0 and m' are not concurrent and $m' \overset{+}{\rightsquigarrow} m_0$ (*i.e.,* m' is a predecessor of m_0 in the message causality graph). Then, (k_0, d_0) cannot belong to $CB_{m'}$ and consequently cannot be suppressed from CB_i at line R8 when m' is delivered to P_i. •

Theorem 2 (Safety) *Delivery events respect causal order.*
Proof Let us consider two messages m_0 and m_x such that (1) $send(m_0) \rightarrow send(m_x)$ (*i.e.,* m_0 is a predecessor of m_x in the message causality graph: $m_0 \overset{+}{\rightsquigarrow} m_x$) and (2) both are delivered to a process P_i. We show that they are delivered to P_i according to causal ordering, *i.e.,* m_0 is delivered to P_i before m_x. The proof is by induction on the lenght l of the path from m_0 to m_x in the message causality graph.
Base case: l=1.
In this case, m_0 is an immediate predecessor of m_x in the message causality graph. From Lemma 1, we have $(k_0, d_0) \in CB_{m_x}$. It follows that lines R4 and R5 will delay the delivery of m_x until after the delivery of m_0.
Induction case: $l \geq 2$.
Let $m_0 \rightsquigarrow m_1 \rightsquigarrow m_2 \cdots \rightsquigarrow m_{x-1} \rightsquigarrow m_x$ be the path from m_0 to m_x in the message causality graph. By induction hypothesis, all messages of the set $\{m_0, m_1, \ldots, m_{x-1}\}$ that are delivered to P_i are delivered according to causal ordering. Two cases have to be considered depending on whether m_{x-1} has been delivered or discarded by P_i (m_{x-1} is timestamped (k_{x-1}, d_{x-1})).
case i: m_{x-1} is delivered to P_i. Then, as m_{x-1} immediately precedes m_x in the message causality graph, the *Base case* applies to these messages: m_{x-1} is delivered before m_x and by transitivity (induction hypothesis) m_0 is delivered before m_x at P_i.
case ii: m_{x-1} is discarded by P_i. In that case, due to $(k_{x-1}, d_{x-1}) \in CB_{m_x}$ (Lemma 1) and lines R4 and R6, it follows that m_x was delivered after the deadline associated with m_{x-1} had expired. Let us assume that this delivery deadline of m_{x-1} corresponds to time t. Each message m_k in the path $m_0 \overset{+}{\rightsquigarrow} m_x$, such that $1 \leq k \leq (x-1)$, has its deadline at or before time t. Hence, if m_k has not been discarded by, or delivered to P_i by time t, it will necessarily be discarded by P_i on its arrival at the

process. This is because its deadline expired before t. From our hypothesis, m_0 is not discarded; it is delivered before t. Consequently, m_x is delivered at P_i after m_0. •

4 Conclusion

In the context of broadcast communication, the Δ-causal ordering abstraction matches the requirements of an important class of multimedia applications, namely, distributed multimedia real-time applications, where real-time communications of each participant should be received by all the others. This flow of information must preserve the causal dependency even though part of the information can be lost or can be discarded if it violates the real-time constraints.

Protocols implementing pure causal ordering [BJ87, RST91] and Δ-causal ordering [BMR95, AS95] suffer from the typical pitfall of the timestamping (logical or physical) technique: to ensure Δ- or pure causal ordering, in the context of broadcasting, messages have to piggyback a vector of n integers where n is the number of processes [Mat89].

In this paper, we introduced a new and efficient Δ-causal ordering protocol reducing the quantity of information piggybacked on messages. Only in the worst case, this information has the same size as previous algorithms. This reduction is obtained by exploiting the transitive dependencies of the message causality graph of a distributed computation. We used the *causal barrier* for a message m that corresponds to the set of the immediate predecessors of m in the message causality graph. Each message m carries its causal barrier CB_m, and the message is delivered to the destination process as soon as each message belonging to its causal barrier has either been delivered or has missed its deadline. The cardinality of a CB set is between zero an $n-1$.

References

[AS95] F. Adelstein, M. Singhal. Real-time causal message ordering in multimedia systems. In *Proceedings of the* 15^{th} *IEEE Int. Conf. on Distributed Computing Systems*, pages 36–43, 1995.

[BMR95] R. Baldoni, A. Mostefaoui, and M. Raynal. Causal delivery of messages with real-time data in unreliable networks. *Journal of Real-Time Systems*, 10(3):1–18, 1996.

[BJ87] K. Birman and T. Joseph. Reliable communication in the presence of failures. *ACM Transactions on Computer Systems*, 5(1):47–76, 1987.

[Fer90] D. Ferrari. Clients requirements for real-time communication services. *IEEE Communication Magazine*, 65–72, 1990.

[GZ89] R. Gusella and S. Zatti. The accuracy of the clock synchronization achieved by TEMPO in Berkeley UNIX 4.3BSD. *IEEE Transactions Software Engineering*, 15(7):847–853, 1989.

[HB94] T. Houdoin and D. Bonjour. ATM and AAL layer issues concerning multimedia applications. *Annals of Telecommunications*, 49(5):230–240, 1994.

[JSS94] K. Jeffay, D.L. Stone, and F.D. Smith. Transport and display mechanisms for multimedia conferencing across packet switched networks. *Computer Networks*, 26:1281–1304, 1994.

[KO87] H. Kopetz and W. Ochsenreiter. Clock synchronization in distributed real-time systems. *IEEE Transactions on Computers*, (8):933–940, 1987.

[Lam78] L. Lamport. Time, clocks and the ordering of events in a distributed systems. *Communications of the ACM*, 21(7):558–565, 1978.

[Mat89] F. Mattern. Virtual time and global states of distributed systems. In *Cosnard, Quinton, Raynal and Robert Editors, Proceedings of the International Workshop on Parallel and Distributed Algorithms*, pages 215–226, North Holland, 1989.

[PRS95] R. Prakash, M. Raynal, and M. Singhal. An efficient causal ordering algorithm for mobile computing environments. In *Proceedings of the* 16^{th} *IEEE Int. Conf. on Distributed Computing Systems*,Hong-Kong, june 1996.

[RB93] K. Ravindran and V. Bansal. Delay compensation protocols for synchronization of multimedia data streams. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):574–589, 1993.

[RST91] M. Raynal, A. Schiper, and S. Toueg. The causal ordering abstraction and a simple way to implement it. *Inform. Process. Lett.*, 39:343–350, 1991.

[Ver94] P. Verissimo. Ordering and timeliness requirements of dependable real-time programs. *Real-time Systems*, 7:105–128, 1994.

[Wak93] I. Wakeman. Packetized video: options for interaction between the user, the network and the codec. *The Computer Journal*, 36(1):55–66, 1993.

[Yav92] R. Yavatkar. MCP: a protocol for coordination and temporal synchronization in multimedia collaborative applications. In *Proceedings of the* 12^{th} *IEEE Int. Conf. on Distributed Computing Systems*, pages 606–613, 1992.

Unit´e de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399