Pomares Hernández, Saúl E.; López Domínguez, Eduardo; Rodríguez Gómez, Gustavo

An Efficient delta-Causal Distributed Algorithm for Synchronous Cooperative Systems in Unreliable Networks
Computación y Sistemas, vol. 14, núm. 1, julio-septiembre, 2010, pp. 31-44
Instituto Politécnico Nacional
Distrito Federal, México

Available in: http://www.redalyc.org/src/inicio/ArtPdfRed.jsp?iCve=61519184004
An Efficient $\Delta$-Causal Distributed Algorithm for Synchronous Cooperative Systems in Unreliable Networks

Saúl E. Pomares Hernández, Eduardo López Domínguez and Gustavo Rodríguez Gómez
Department of Computer Science, National Institute of Astrophysics, Optics and Electronics (INAOE)
Luis Enrique Erro No. 1, Tonantzintla, Puebla, Mexico, C.P. 72840
spomares@ccc.inaoep.mx, edominguez@ccc.inaoep.mx, grodrig@ccc.inaoep.mx

Abstract. In cooperative systems causal ordering delivery has been used to resolve problems of coherency of type producer-consumer. Causal order delivery is important for distributed systems since it allows an asynchronous execution to participants. When time delivery constraints are considered, ensuring causal delivery becomes more complex, as is the case for synchronous cooperative systems, such as Telemedicine and Teleimmersion. In these systems, the messages (units of data of continuous and discrete media) have an associated lifetime that determines the period of useful time in which the messages must be delivered. On the other hand, generally in these systems there is no time for retransmit them when messages are lost. Causal order with time constraints has previously been addressed, and it is called $\Delta$-causal order. In this paper, we present an efficient $\Delta$-causal distributed algorithm for unreliable networks that is characterized by the use of a forward error correction (FEC) scheme and a distributed method to calculate the message lifetime based on relative time points (i.e. no global time is used). We show the efficiency of our $\Delta$-causal algorithm in terms of the control information attached per message.

Keywords: Cooperative systems, Group communication, Causal order.

1 Introduction

Although synchronous cooperative systems have been a major research focus in computer supported cooperative work (CSCW) for over two decades, there is still a lack of protocols oriented to support their communication requirements, which guarantee, at runtime, ordering dependencies and time constraints. In this sense, causal protocols have been used in cooperative systems mainly to resolve problems of coherency of type producer-consumer (Plesca et al. 2005, Pomares et al. 2002). In general, the causal order delivery (Birman et al. 1993) ensures that for each participant in the system the events (send and delivery of messages) will be seen in the cause-effect order as they occur in the system.

Many works concerning protocols of causal ordering delivery exist (Pomares et al. 2004; Birman et al. 1993; Kshem-kalyani et al. 1998). The previous algorithms assume a reliable transmission without an
associated lifetime per message. These works are not suitable for synchronous cooperative systems since these systems intrinsically have time constraints that are not considered by them.

Synchronous cooperative systems are characterized by two main time constraints. First, the information units (continuous and discrete) have an associated lifetime that establishes the period of time in which the information (messages) must be received; a message that arrives after its lifetime is useless and, consequently, discarded. The second constraint establishes that there is no time for retransmission when messages are lost. In order not to greatly affect the quality of service, a forward recovery scheme is preferable over a backward recovery scheme (Perkins et al. 2003).

Causal order with time constraints has previously been addressed by Baldoni (1998), and it is called \( \Delta \) -causal order. In his work, Baldoni ensures \( \Delta \) -causal order by using a global clock. In our work, we propose an algorithm that achieves \( \Delta \) -causal order in unreliable networks while avoiding the use of global references. To achieve this, we propose an original FEC mechanism and a method to calculate, in a distributed manner, the lifetime per message for continuous and discrete media units. The FEC mechanism ensures that causal order delivery is accomplished even in the presence of lost messages. The lifetime in our work is calculated based on relative time points\(^1\). Our work is intended for the transmission of continuous data, such as audio and video, and discrete data such as text and still images.

We apply our FEC mechanism and our distributed lifetime method to extend the minimal causal broadcast algorithm presented in Pomares et al. 2004. This minimal algorithm only sends control information about messages with an immediate dependency relation (IDR). Messages related by an IDR have a causal distance (see Definition 4) of one (i.e., no intermediate causal message exists between them). In order to support delays and loss of messages, we introduce redundancy on the control information by sending information about messages with a causal distance greater than one. One interesting aspect of our FEC mechanism, as we will show in Section 4, is that the redundancy is dynamically adapted according to the behavior of the system.

The rest of the article is structured in the following way: Section 2 presents the most relevant related works concerning the \( \Delta \) -causal ordering. In Section 3, the system model is described and the background information is presented. Next, we present in Section 4 our \( \Delta \) -causal order algorithm with our proposed FEC mechanism and the distributed life-time method. A sketch of the algorithm correctness proof is presented in Section 5. Finally, some conclusions are presented in Section 6.

2 Related Work

The most important work that tackles the problem of causality and time constraints was presented by (Baldoni et al. 1998). Baldoni addresses the problem of causality and time constraints by introducing the \( \Delta \) -causal order. Informally, the \( \Delta \) -causal order says that a message \( m \) is \( \Delta \) -causally-related to another message \( m' \) if \( m \) causally precedes \( m' \) and arrives before its deadline. Baldoni ensures message \( \Delta \) -causal order by using a reference global clock. More specifically, by using a global clock, Baldoni determines if a message accomplishes the causal order and the time delivery constraints. The \( \Delta \) -causal order defined by Baldoni is correct; however, the use of a global clock is not suitable for distributed communication systems where the message transmission delay is not negligible (Lamport et al. 1978).

Several \( \Delta \) -causal algorithms exist (Baldoni et al. 1996; Prakash et al. 1997; Tachikawa et al. 1997) that ensure causal ordering in the presence of lost messages and time delivery constraints; however, to achieve this, all of them use some type of global reference (shared memory, wall clock, master-slave scheme, etc).

3 Preliminaries

3.1 The System Model

Processes: The application under consideration is composed of a set of processes \( P = \{j, k, \ldots\} \) organized into a group that communicates by passing non reliable broadcast asynchronous messages. Messages: We consider a finite set of messages \( M \), where each message \( m \in M \) is identified by a tuple \( m = (p, t) \), where \( p \in P \) is the sender of \( m \), denoted by

\(^1\) A relative time point establishes a reference point from which it is possible to calculate a period of time.
3.2 Background and Definitions

The Happened-Before Relation. The happened-before relation was defined by Lamport (1978). The happened-before relation establishes possible precedence dependencies in a set of events without using physical clocks. It is a strict partial order (i.e. irreflexive, asymmetric and transitive) defined as follows:

Definition 1. The causal relation “→” is the least partial order relation on E satisfying the following properties:
1. If a and b are events belonging to the same process, and a was originated before b, then a→b.
2. If a is the send message of a process, and b is the reception of the same message in another process, then a→b.
3. If a→b and b→c, then a→c.

By using Definition 1, we define that a pair of events are concurrent related “a || b” only if

¬(a→b ∨ b→a)

The precedence relation on messages denoted by m→m’ is induced by the precedence relation on events, and is defined by:

m→m’ ⇔ send(m)→send(m’)

The Immediate Dependency Relation. The Immediate Dependency Relation (IDR) introduced in Pomares et al. 2004 is the propagation threshold of the control information regarding the messages sent in the causal past that must be transmitted to ensure a causal delivery. We denote it by ↓, and its formal definition is the following:

Definition 2. Immediate Dependency Relation “↓” (IDR):

a↓b⇔[(a → b) ∧ ∀ c ∈ E, ¬(a → c→ b)]

Thus, an event a directly precedes an event b, iff no other event c belonging to E exists, such that a precedes c and c precedes b. We note that the IDR relation is the transitive reduction of the Lamport’s relation. This is important because if the delivery of messages respects the order of the diffusion for all pairs of IDR related messages, then the delivery respects the causal order for all messages (Pomares et al. 2004). This property is formally defined for the broadcast case as follows:

Property 1:

∀ m,m’ ∈ M, if send(m) ↓ send(m’) ⇒ ∀ p ∈ P : delivery(p,m)→delivery(p,m’) then send(m)→send(m’) ⇒ ∀ p ∈ P : delivery(p,,m)→delivery(p,,m’)

The Δ-Causal Ordering. The Δ-causal ordering has been introduced in Baldoni et al. 1998; it is formally defined for the broadcast case as follows:

Definition 3. A set of events E satisfies theΔ -causal ordering if:
1. All events that arrive inΔ are delivered within Δ.
All other events are considered to be lost or discarded, and therefore, are never delivered.
2. All delivery events respect causal ordering, i.e.

∀ m,m’∈ M, if send(m)→send(m’), then ∀ p ∈ P : delivery(p,m)→delivery(p,m’)

The Causal Distance. The causal distance identifies the number of causal messages that exist in a linearization between a pair of messages in the system (Lopez et al. 2005). Formally, the causal distance is defined as follows:

Definition 4. Let m and m’ be messages, the distance d(m,m’) is defined for any pair m and m’ (send(m)→send(m’)), such that d(m,m’) is the integer n for some sequence of messages (m_i, i= 0...n), with m= m_0 and
\( m' = m_n \), such that \( \text{send}(m_i)_i\text{send}(m_{i+1}) \) for all \( i = 0 \ldots n - 1 \).

4 The \( \Delta \)-Causal Order Algorithm

In order to avoid the use of a global clock, we propose an original FEC mechanism and a distributed lifetime method that verifies if a message satisfies or not its deadline. In this section we give a general description of each mechanism separately, and then integrate them to the minimal broadcast causal algorithm.

4.1 The FEC Mechanism

All FEC mechanisms introduce some kind of redundancy to support the loss of information. The redundancy in causal algorithms represents the number of times that information about a causal message is sent in the system. The causal algorithm presented in Pomares et al. 2004, which uses the IDR relation, is minimal because the IDR relation identifies the necessary and sufficient causal information that needs to be sent attached per message (denoted in this paper by \( H(m) \)). Even when this is a minimal algorithm, redundant control information is still transmitted in some communication scenarios. Our FEC mechanism identifies and uses this inherent redundancy in order to be efficient and will only add extra redundancy when it is needed. The purpose of adding extra redundancy is to increase the probability that causal order delivery will be obtained, even in the presence of lost messages and significant network delays.

Redundancy and the IDR relation

To ensure causal ordering, the minimal algorithm only sends control information attached to each message about messages with an immediate dependency relation. For two messages that are IDR-related \( (m \downarrow m') \), the causal distance is equal to one \( (d(m,m') = 1) \). Note that for the serial case, a message \( m \) has only one immediate predecessor (best case), and that a message \( m \) can have at most \( n \) immediate predecessors, one for each process.

For the serial case, for messages that are IDR-related, there is no redundancy in the control information sent. For example, in the serial scenario depicted in Figure 1a, message \( m_3 \) only sends causal information about message \( m_2 \) \( (H(m_3) = \{ m_2 = (p_3, t_2) \}) \) and message \( m_2 \) only sends information about message \( m_1 \) \( (H(m_2) = \{ m_1 = (p_1, t_1) \}) \). In this case, if a message is lost, the causal order delivery can be violated. As shown in Figure 1a, the causal order delivery is violated because at the reception of message \( m_3 \), process \( p_3 \) cannot determine if a message preceding \( m_2 \) exists or not. With the IDR information on \( m_3 \), process \( p_3 \) can only detect that it missed message \( m_2 \). In order not to stop the system execution, process \( p_5 \) considers message \( m_2 \) as lost and then delivers \( m_3 \). In this scenario, \( m_1 \) can be delivered after \( m_3 \), which violates the causal ordering.

For the concurrent relation, inherent redundancy exists on the causal information sent. For example, in the scenario depicted in Figure 1b, messages \( m_2 \) and \( m_4 \) have the same immediate predecessor \( m_1 \), and therefore \( m_2 \) and \( m_3 \) send information about message \( m_1 \) \( (H(m_2) = H(m_3) = \{ m_1 = (p_1, t_1) \}) \). If either message \( m_2 \) or \( m_3 \) is lost, message \( m_1 \) can still be detected as shown in Figure 1b. In this scenario, \( m_2 \) is lost and \( m_3 \) successfully arrives at \( p_5 \). With the IDR information on \( m_3 \), process \( p_5 \) determines that \( m_1 \) exists, which precedes message \( m_3 \). To deliver \( m_3 \), process \( p_5 \) establishes message \( m_1 \) as lost. In this scenario, \( m_1 \) arrives at \( p_5 \) after the delivery of message \( m_4 \), but since message \( m_1 \) has been established as lost, it is immediately discarded. Therefore, causal order is ensured.

Our Proposal

In order to support the loss of messages, we propose to increase the redundancy in the control information sent per message by sending information about causally-related messages with a causal distance greater than one. For example, in Figure 1a, if we establish a causal distance of two \( (\text{causal\_distance} = 2) \), this means that message \( m_3 \) must send information about \( m_2 \) and \( m_1 \).
To be efficient, the redundancy is increased considering the inherent redundancy introduced by the IDR relation. We formally define that redundancy about a message \( m \), denoted by \( \text{redundancy}_p(m) \), determines the number of times that the information about a causal message \( m \) has been seen (received) by a participant \( p \). As previously described, the redundancy increases as the number of concurrent messages increases. Taking into account \( \text{redundancy}_p(m) \) with a causal distance greater than one (\( \text{causal\ distance} > 1 \)), we establish that a message \( m' \) must include information about a causal message \( m \) (\( m \rightarrow m' \)) only if the following propagation constraints are satisfied:

\[
\begin{align*}
\text{PC1:} & \quad d(m, m') \leq \text{causal\ distance} \quad \text{and} \\
\text{PC2:} & \quad \text{causal\ distance} > \text{redundancy}_p(m)
\end{align*}
\]

With both of these PCs, the control information sent per message is dynamically adapted to the behavior of the system by only introducing redundancy when it is needed. For example, with \( \text{causal\ distance} = 2 \), message \( m_3 \), shown in Figure 1a, must send causal information about \( m_2 \) and \( m_1 \) (\( H(m_3)=\{m_2, m_1\} \)) because \( p_4 \) has \( \text{redundancy}(m_1) \) equal to one and a causal distance of \( d(m_1, m_3) = 2 \) and \( d(m_2, m_3) = 1 \), respectively. Nevertheless, for the scenario presented in Figure 1b, message \( m_4 \) must send information only about messages \( m_2 \) and \( m_3 \) (\( H(m_4)=\{m_2, m_3\} \)) and not about \( m_1 \), even when \( d(m_1, m_4) = 2 \). This is done because the \( \text{redundancy}(m_1) \) seen by \( p_4 \) is equal to 2, and therefore, it does not satisfy the second PC.

We note that the value of \( \text{redundancy}_p(m) \) can differ between participants since it is calculated from the messages received by each one. In a general case, according to the analysis presented in Appendix I, it is sufficient to take a \( \text{causal\ distance} \) equal to 5 since the probability that three or more consecutive and/or concurrent messages can be lost is very low.

### 4.2 The Distributed Lifetime Method

The distributed lifetime method identifies two cases. one case for continuous media data, and another case for discrete media data. In the transmission of continuous media data, such as audio and video it is possible to establish relative time points (ReTPs) since the messages are periodically sent, and this ReTPs can be used to determine if the units of data satisfies or not its lifetime. Nevertheless, the ReTPs must be dynamically established in order to support random transmission delays. For the case of discrete media units is totally different since they are no periodical, which means that the period of time between emissions is variable. For this reason, we propose for a discrete media unit to calculate its lifetime based on the lifetime of the causally-related messages of continuous media that are included in its causal control information. Next, we will present the lifetime method for continuous media followed by the method for discrete media.

### Establishing Relative Time Points and Deadline Points for Continuous Media

In order to establish the relative time points and deadline points, we assume that the transmission of data, such as audio and video, is executed by transferring messages at a relatively constant rate, and that these messages are sequentially timestamped. By taking into account these hypotheses, we establish a ReTP at the reception of the most recent message. For example, in Figure 2, the reception of message \( m_1 \) establishes the first relative time point \( rtp_1 \); the reception of \( m_2 \) establishes the \( rtp_2 \), and so on.

**Fig. 2.** Streaming scenario

Based on the ReTPs, the deadline point of a message \( m \), denoted by \( \text{deadline\_cont}(m) \), is determined from the ReTP previously established. If no message is lost and the messages arrive in order, we have:

\[
\text{deadline\_cont}(m) = rtp_{i-1} + \Delta : i \geq 1
\] (1)
where $\Delta$ is the message lifetime that establishes the maximum transmission delay supported\(^2\).

If we consider lost or discarded messages, the equation above is redefined as follows:

$$\text{deadline\_cont}(m_i)=\text{rtp}_x + (i-x)\Delta : x < i \text{ and } i \geq 1$$

(2)

where $\text{rtp}_x$ is the last relative time point established. This general equation is used in the rest of the paper to calculate the deadline for continuous media.

For broadcast asynchronous communication, each process $p \in P$ locally establishes its own ReTPs, and therefore, its own deadline points. For this case, we denote a deadline as $\text{deadline\_cont}(m, p)$, which determines the deadline point for a message $m$ at a process $p$. It is the same case for discrete media as we will present in the next paragraph.

**Deadline Points for Discrete Media**

As we previously said, the period of time between the emission of messages of discrete media is variable, and therefore we cannot take it as reference. For this reason, we take the deadline points of the messages of continuous media included in the causal information $H(m)$. For each message we take its causal information $H(m)$, and we find its maximum deadline point. We take this the maximum deadline point as reference to determine the deadline point for the discrete message. Formally, a deadline point for a discrete message is calculated as follows:

$$\text{deadline\_disc}(m) = \max\{\text{deadline\_cont}(m') : m' \in H(m) \text{ and } \text{type}_{m'}=\text{continuous}\} + \delta$$

(3)

where $\delta$ is the lifetime established for discrete media units. We recall that every message in its causal information $H(m)$ satisfies the PC1 and PC2 presented in Section 4.1.

More specifically, we take the maximum deadline point since in order to causally deliver a message $m$, it must wait until and delivery after that every message that belongs to its $H(m)$ has been causally delivered or has been discarded as a consequence of its lifetime expiration, which is determined by its deadline point.

In the case that $m' \in H(m)$, such that $\text{type}_{m'}=\text{continuous}$ we have:

$$\text{deadline\_disc}(m) = \text{receive\_time}_p(m) + \delta$$

where $\text{receive\_time}_p(m)$ gives the time when message $m$ has been received at participant $p$.

### 4.3 The Algorithm Code

**Data Structures**

The main data structures used in the algorithm are:

- $VT(p)$ is the vector time. For each process $p$ there is an element $VT(p)[j]$ where $j$ is a process identifier. When we need to refer to a specific process with its respective identifier, we write $p_j$. The size of $VT$ is equal to the number of processes in the group. $VT(p)$ contains the local view that process $p$ has of the elements of the system. In particular, element $VT(p)[k]$ represents the greatest element number of the identifier $k$ and ‘seen’ in causal order by $p$. It is through the $VT(p)$ structure that we are able to guarantee the causal delivery of elements.

- $CI(p)$ is the control information structure. It is a set of entries $(k, t, d, \text{type})$. Each entry in $CI(p)$ denotes a message that satisfies the PC1 and PC2 propagation constraints and that can potentially be attached in the next message $m$ sent by $p$. The entry $(k, t, d, \text{type})$ represents a message diffused by participant $k$ at a logical local timeclock $t = VT(p)[k]$, $d$ contains the redundancy of $m=(k, t)$ seen by $p$ and $\text{type}$ determines the type of media transmitted, which can be continuous or discrete.

- The structure of a message $m$ is a quadruplet $m=(j, t, \text{type}, \text{content}, H(m))$, where:
  - $j$ is the participant identifier.
  - $t=VT(p)[j]$ is the logical local clock at node $j$.
  - $\text{type}$ determines the type of media transmitted.
  - $\text{content}$ is the structure that carries the media data.

---

\(^2\) For simplicity, in our work, we consider only $\Delta$ for all messages in the system.
- \( H(m) \) contains the set of all entries \((k, t, \text{type})\) about messages that satisfy the propagation constraints (PC1 and PC2) with \( m \).

- The causal\( _\text{distance} \) variable is the predetermined causal distance considered.

- \( VTIME(p) \) is a vector that contains the most recent relative time points. The size of \( VTIME(p) \) is equal to \( VT(p) \) (one element for each process in the system).

- \( current\text{\_time}(p) \) represents the physical local time of a process \( p \).

- \( deadline(k,t,\text{type}) \) is a function that calculates the deadline point for message \( m \) identified by \((k,t)\) where \( k \) is the sender process and \( t \) is the logical clock. This function considers the two types of discrete and continuous messages.

**Algorithm Description**

The \( \Delta \)-causal algorithm is presented in Table 1. When a message \( m \) is broadcasted (continuous or discrete) by a process \( p \), the \( H(m) \) is constructed by adding entries from the \( CI(p) \) (lines 12-14) to it. Each entry in \( H(m) \) satisfies, with respect to \( m \), the PC1 and PC2 propagation constraints. In order to comply with PC1 and PC2, we use a logical counter \( d \) by each entry in \( CI(p) \) \(((k, t, d, \text{type}) \in CI(p))\). The variable \( d \) is increased by one each time that its associated entry is added to a \( H(m) \) by a process \( p \) (line 13) or when that entry is received in a \( H(m) \) (lines 39-40). The variable \( d \) contains the redundancy of the message \( m=(k, t) \) seen by \( p \). We note that only when no concurrent messages exist the value of \( d \) specifies the causal distance between events.

**The \( \Delta \)-Causal Delivery Condition.** At the reception of a message, \( m=(k, t, \text{type, } H(m)) \) will be immediately discarded if it has already been marked as lost \((t < VTIME(k))\) or if it misses its deadline (line 20). If \( m \) is not discarded, it is delivered as soon as the \( \Delta \)-causal delivery condition becomes true (lines 26-30). This delivery condition ensures that a message \( m \) is delivered in its lifetime (lines 27 and 30) and that it will be delivered if and only if all messages causally related to it have either been delivered or have been established as missing, i.e. that its lifetime has been expired. *A posteriori*, these messages are marked as lost in lines 34-35, and therefore, they will never be delivered. We note that in order to ensure \( \Delta \)-causal delivery if at a participant \( p \), a message \( m' \) in the causal future of a message \( m \) \((m \rightarrow m') \) has a smaller lifetime than \( m \) \((\text{deadline}(m,p) > \text{deadline}(m',p)) \) and both messages have arrived but are not yet delivered, we make \( \text{deadline}(m) = \text{deadline}(m') \). This is done in order to ensure Rule one of definition 3, which says that all events that arrive in \( \Delta \) are delivered within \( \Delta \). This behavior is better illustrated in the proof of Section 5.

**Overhead analysis.** In order to be efficient, each entry in \( CI(p) \), and eventually in \( H(m) \) corresponds to the most recent message sent by a process \( p_j \in P \) and causally received by \( p \). This is possible since each message \( m \) is sequentially timestamped with its respective local logical clock of \( p_j = \text{Src}(m) \). By knowing the sequential order, a participant \( p_j \) can determine at any message reception if a message or set of messages diffused by \( p_j \) has been lost, independently of the causal distance.

Since \( H(m) \) only has the most recent messages that precede a message \( m \), the overhead per message in this algorithm to ensure \( \Delta \)-causal ordering is given by the cardinality of \( H(m) \), which can fluctuate in our case between 0 and \( n-1 \) \((0 \leq |H(m)| \leq n-1) \). For the serial case, \(|H(m)| \) is at most the causal\_distance established \((|H(m)| \leq \text{causal\_distance}) \), and for the case of concurrent messages, the worst case is at most \( n-1 \) \((|H(m)| \leq n-1) \), which is the same boundary for messages that are IDR related \((\text{causal\_distance}=1) \).

We note that in our algorithm, as for the minimal causal algorithm in (Pomares et al. 2004), the likelihood that the worst case will occur approaches zero as the number of participants in the group grows. Compared with algorithms that are exclusively based on vector clocks (Matern et al. 1989), our worst case denotes for them the constant overhead that must always be attached per message.
### Table 1. Multimedia Broadcast Δ-Causal algorithm

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Initially</td>
</tr>
<tr>
<td>2.</td>
<td>$VT[p][j] = 0 \forall j : 1\ldots n$ /* Vector clock */</td>
</tr>
<tr>
<td>3.</td>
<td>$VTIME[p][j] = 0 \forall j : 1\ldots n$ /* Vector with the ReTPs */</td>
</tr>
<tr>
<td>4.</td>
<td>$CI(p), H(m), \text{deadline} _\text{arr}(m) \leftarrow \emptyset$</td>
</tr>
<tr>
<td>5.</td>
<td>$\text{causal} _\text{distance} = z$</td>
</tr>
<tr>
<td>6.</td>
<td>let $\text{deadline} _\text{cont}(k,t) = VTIME[p][k] + (t - VT[p][k]) * \Delta$</td>
</tr>
</tbody>
</table>
| 7.   | let $\text{deadline} \_\text{disc}(k,t) = \text{if } \exists m' = (k',t') \in H(m) \text{ such that } type_{m'} = \text{continuous} \text{ then}$
|     | $\max(\text{deadline} \_\text{cont}(m' = (k',t')) : (k',t') \in H(m = (k,t)))$ |
|     | $\text{else}$
|     | $\text{receive} \_\text{time}_p(m) + \delta$ |
| 8.   | let $\text{deadline}(k,t,type) = \text{if } type = \text{continuous} \text{ then}$
|     | $\text{deadline} \_\text{cont}(k,t)$ |
|     | $\text{else}$
|     | $\text{deadline} \_\text{disc}(k,t)$ |
| 9.   | let $\text{deadline} \_\text{arr}(k,t,type) = \{ \text{deadline}(m = (k,t,type)) \cup \text{deadline}(m' = (k',t'.type')) : m' \text{ arrived in its lifetime and not yet delivered and } m \rightarrow m' \}$ |
| 10.  | For each diffusion of message send$(m)$, at $p_j$ |
| 11.  | $VT[p][j] = VT[p][j] + 1$ |
| 12.  | for all $(k,t,d,type) \in CI(p)$ |
| 13.  | $(k,t,d,type) \leftarrow (k,t,d+1,type)$ /* Accounts for redundancy */ |
| 14.  | $H(m) \leftarrow H(m) \cup \{ k,t,type \}$ endfor |
| 15.  | $m = (j, t=VT[p][j], type, content, H(m))$ |
| 16.  | Diffusion: send$(m)$ |
| 17.  | for all $(k,t,d,type) \in CI(p)$ if $d = \text{causal} \_\text{distance}$ then |
| 18.  | $CI(p) \leftarrow CI(p) / (k,t,d,type)$ endfor |
| 19.  | For each reception receive$(m)$ at $p, m = (k, t, type, content, H(m))$ |
| 20.  | if $(t < VT[p][k]) \text{ or } \text{deadline}(k, t, type) < \text{current} \_\text{time}(p)$ then |
| 21.  | if not $(t < VT[p][k])$ then |
| 22.  | $VTIME[p][k] = \text{current} \_\text{time}(p)$ endif |
| 23.  | $VT[p][k] = \max(VT[p][k], t)$ |
| 24.  | Discard$(m)$ |
| 25.  | else |
5 Correctness Proof

To show that our algorithm ensures the causal delivery (correctness), we give a sketch of proof. In order to do the proof as simple as possible, we focus on showing that the time constraints are satisfied and that the causal order is guaranteed independently of the data type. For this reason, we avoid using the type of data when referring to the messages. We refer to them only by the participant identifier and the logical clock, such as \( m = (k, t) \).

Theorem 1. (Liveness) i) All messages arriving within their deadlines and whose deliveries do not violate causal ordering will be delivered within their deadlines, and ii) All messages arriving after the expiration of their deadlines or whose delivery would cause a causal violation will be discarded.

Proof Point ii) is ensured from the test of line 20. Point i) is proved by contradiction. Suppose that a message \( m = (k, t) \) exists that arrived within its deadline, but is not delivered within its deadline. To proof this, we first introduce Lemma 1.

Lemma 1 Each variable in \( VTIME(p)[j] \), for all \( j:1…n \) does not decrease.

The proof follows directly from the algorithm (lines 20 and 31)

To proof Point i) we have two cases:

a) Messages from the same source. For this case, by using Lemma 1, we have the following property:

\[
P1) \text{For all } m = (k, t) \in M : \text{Src}(m) = p \Rightarrow \forall \ p \in P, \text{ deadline}_p(k, t) < \text{deadline}_p(k, t') \forall \ t':1,2,…,t-1
\]
Denying the first part of the delivery condition (line 27) that corresponds to messages from the same source, we have that
\[ \exists (k, t'), t' < t : (\text{deadline}(k, t' = \text{VT}(p)[k]) \geq \text{current}_t(p)) \]
On the deadline of message \( m = (k, t) \), we have that \( \text{current}_t(p) = \text{deadline}(k, t) \). So by direct replacement, we have:
\[ \exists (k, t'), t' < t : (\text{deadline}(k, t') \geq \text{deadline}(k, t)) \]
This sentence contradicts property P1. It follows that at the deadline of an arrived message \( m \), the first part of the denied delivery condition is false, thus contradicting our initial assumption.

b) Messages from a different source. To proof this case, we first introduce two functions: \( \text{delivery}_p(m) \), which is the time when a message \( m \) is delivered at a process \( p \) and \( \text{discarded}_p(m) \), which is the time when a message \( m \) is marked as missing at a process \( p \). By using lines 27, 30 and Lemma 1, we have the second and third properties:

P2) For all \( m', m \in M \), \( m' \rightarrow m \) received at \( p \in P \Rightarrow \text{delivery}_p(m) \leq \text{deadline}_p(m) \)

P3) For all \( m', m \in M \), \( m' \rightarrow m \); \( m' \) has not been received at \( p \in P \Rightarrow \text{discarded}_p(m') \leq \text{deadline}_p(m) \)

Next, we only present the proof that involves P2. The proof involving P3 is similar and not presented here.

For \( m' \rightarrow m \) received at \( p \in P \). By denying the second part of the delivery condition (line 30), we have:
\[ \exists m' = (l, x) \in H(m): \]
\[ (\text{min}(\text{deadline}(l, x), \text{deadline}_p(m))) > \text{current}_t(p)) \]
If we do \( \text{delivery}_p(m, l, x) = \text{min}(\text{deadline}(l, x), \text{deadline}_p(m))) \), we have
\[ \exists m' = (l, x) \in H(m): \]
\[ (\text{delivery}_p(m, l, x)) > \text{current}_t(p) \]
On the deadline of message \( m = (k, t) \), we have that \( \text{current}_t(p) = \text{deadline}(k, t) \). So by direct replacement, we have:
\[ \exists m' = (l, x) \in H(m): \]
\[ (\text{delivery}_p(m, l, x)) > \text{deadline}(m) \]
This sentence contradicts property P2. It follows that at the deadline of an arrived message \( m \), the second part of the denied delivery condition is false, thus contradicting our initial assumption.

Lemma 2. For all \( m', m \in M \), \( m' \rightarrow m \) such that \( \text{Src}(m') \neq \text{Src}(m) \) and \( \text{redundancy}_p(m') \leq \text{causal}_p(m) \) implies that \( m' = (l, x) \in H(m) \)

This is accomplished by the procedures at the diffusion message by lines 12 and 17, and at the reception message by lines 38, 39 and 41.

Theorem 2. (Correctness) For all \( m', m \in M \), \( m' \rightarrow m \) such that \( \text{delivery}_p(m') \leq \text{causal}_p(m) \) implies that \( m' \rightarrow m \).

Proof. Let us consider two messages \( m_0 \) and \( m_n \) such that \( \text{send}(m_0) \rightarrow \text{send}(m_n) \) and both are received by \( p \). We show that they are delivered to \( p \) according to causal ordering.

For this proof, we have two general cases. The proof is by induction on the distance \( d(m_0, m_n) \).

Base case: \( d(m_0, m_n) = 1 \) and \( d(m_0, m_n) \leq \text{causal}_p(m) \)

In this case, \( m_0 \) is IDR related to \( m_n \), and from lemma 2 and since always \( d(m', m) \leq \text{redundancy}(m') \), we have \( m_0 \in H(m_n) \). It follows that line 29 will delay the delivery of \( m_n \) until after the delivery of \( m_0 \).

Induction case: \( d(m_0, m_n) \geq 2 \) and \( d(m_0, m_n) \leq \text{causal}_p(m) \)

By induction, we have that all messages of the set \( \{m_i \in M : m_{i+1} \downarrow m_i \text{ for all } i = 1 \ldots n-1\} \) that are delivered to \( p \) are delivered in causal order. For the
induction phase, we have two cases depending on whether \( m_{n+1} \) has been delivered or discarded at \( p \).

\textbf{a) For } \( m_{n+1} \) \textbf{delivered at } \( p \). We have \( m_{n+1} \) that immediately precedes \( m_n \) so the base case applies to these messages: \( m_{n+1} \) is delivered before \( m_n \) and by transitivity \( m_0 \) is delivered before \( m_n \).

\textbf{b) For } \( m_{n+1} \) \textbf{discarded at } \( p \). In this case \( m_{n+1} \in H(m_n) \) and by Lemma 1 and Lemma 2 and P3, it follows that \( m_n \) is delivered after that \( disc_{time_p} (m_{n+1}) \). By lines 27, 30 and Lemma 1, we have that for a message \( m_i \) that belongs to the path \( m_0 \) to \( m_{n+1} \) implies that the delivery or discarded time of \( m_i \) is less than or equal to the discarded time of \( m_{n+1} \). Consequently, \( m_n \) is delivered at \( p \) after \( m_0 \).

We notice that when a message \( m_{n+3} \) such that \( n \cdot \gamma > \text{causal distance} \), we have \( m_{n+3} \notin H(m_n) \) and therefore, we cannot ensure the causal delivery of \( m_{n+3} \) with respect to \( m_n \).

\section{Conclusions}

An efficient \( \Delta \) -causal algorithm has been presented. The algorithm is efficient since the control information attached per message is dynamically adapted to the behavior of the system. We have shown that this control information allows us to perform a causal forward error recovery when messages are lost. Our algorithm ensures \( \Delta \) -causal order delivery without using a global clock. To avoid the use of a global clock, we have proposed an original FEC mechanism and a distributed lifetime method. Our \( \Delta \) -causal algorithm is suitable for synchronous cooperative systems since it performs a forward error recovery, and it neither uses global references nor requires previous knowledge of the behavior of the system.

\section{References}


Appendix I

Analysis of Probabilities

Let us consider that we have \( E_i \) independent events (send events) with \( i = 1, \ldots, m \), and let us suppose that the rate of their delivery or loss is \( \lambda > 0 \), obeying a Poisson distribution.

\[
p(j) = p\{X = j\} = e^{-\lambda} \frac{\lambda^j}{j!}, \quad j = 0,1,\ldots,m,
\]

where \( X \) is a random variable that takes one of the values 0,1,\ldots.

We consider the events to be successful if they arrive to their destination, and unsuccessful to those who do not arrive. Suppose that there are \( n < m \) events of the \( m \) possible ones, then

1. The probability that an event is unsuccessful is
   \[
p(0) = e^{-\lambda}
   \]
2. The probability that at least one event is unsuccessful is given by
   \[
p\{X \geq 1\} = 1 - p(0) = 1 - e^{-\lambda}
   \]
3. The probability that there is no more than \( n \) successful events is given by
   \[
p\{X \leq n\} = e^{-\lambda} \sum_{j=0}^{n} \frac{\lambda^j}{j!}
   \]
4. The probability that there are at least \( n \) or more unsuccessful events that do not arrive, is given by
   \[
p\{X \geq n\} = 1 - e^{-\lambda} \sum_{j=0}^{n} \frac{\lambda^j}{j!}
   \]

If we consider a loss rate of \( \lambda = 0.1 \) (Olsen et al. 2003), the diagram corresponding to case (4), which is the case that we are interested in, is presented below.

As we can see, the diagram approaches zero extremely fast. From values \( n \geq 3 \), the likelihood becomes negligible.

Saul Eduardo Pomares Hernandez is a Researcher in the Computer Science Department at the National Institute of Astrophysics, Optics and Electronics (INAOE), 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.

Eduardo Lopez Dominguez is currently a PhD student in the Department of Computer Science at the INAOE. He holds the M.S. degree in Computer Science from the same institute in 2006. His research interests include mobile distributed systems and multimedia communications. His postgraduate studies are supported by the National Council of Science and Technology of Mexico (CONACYT).
**Gustavo Rodriguez Gomez:** Received the Bachelor degree and Master degree in Mathematics from the National Autonomous University of Mexico (UNAM). He has a PhD degree in Computational Sciences from the INAOE. His current research interests include scientific computing and the numerical solution of partial differential equations and ordinary differential equations with radial basis functions, multirate methods also called subcycling methods.