"Detecting Immediate Dependencies among Events in Overlapping Multicast Channels"

International Journal of Parallel, Emergent and Distributed Systems
Vol. X, No. X, Month 200X, 000–000

International Journal of Parallel, Emergent and Distributed Systems

ARTICLE TYPE [Research-article]

Article Title [An Efficient Causal Ordering Algorithm for Multicast Communication Channels]

Saul Eduardo Pomares Hernandez[a] , Luis Alberto Morales Rosales[a] and Jean Fanchon[b]

[a] Computer Science Department, National Institute of Astrophysics, Optics and Electronics,
Puebla, Mexico; {spomares, lamorales}@inaoep.mx
[b] Université de Toulouse, LAAS-CNRS,
Toulouse, France; fanchon@laas.fr

Abstract

In this paper we present an efficient causal algorithm that can be used in multicast communication environments, in particular the overlapping multi-channel case, where a participant can belong and communicate through more than one multicast channel. The causal algorithm is built on the paradigm of group communication. The groups are established according to the participant channel subscription. In order to reduce the amount of control information, we propose an extension of the Immediate Dependency Relation (IDR), which was introduced by Peterson in the context of one group. This IDR extension allows us to define necessary and sufficient control information to ensure causal delivery in a multi-group environment. We show that through the use of the IDR extension, we reduce the amount of control information sent per message without imposing restrictions in interaction or execution (e.g. network topology, re-diffusion servers, executions models, etc). These characteristics allow our algorithm to be suitable for use in large distributed decentralized systems. We show the efficiency of our causal algorithm in terms of the overhead timestamped per message.

Keywords*:* Immediate dependency relation; causal ordering; group communication; distributed systems.

I. Introduction

In Multi-party Communication Systems, causal ordering algorithms are an essential tool to exchange information. The use of causal ordering provides built-in message synchronization and reduces the non-determinism in a distributed computation. Causal ordering provides an equivalent of the FIFO property at a global multiparty communication level; it guarantees that actions, like requests or questions, are received before their corresponding reactions, results or responses. The concept of causal ordering is of considerable interest to the design of distributed systems, and can be found in applications of several domains, such as distributed cooperative engineering [18], teleconferencing [15], multimedia systems [3], mobile computing [16], resource allocating algorithms [21] and security domains [17].

We address in this paper the causal ordering of messages over multicast communication channels based on the paradigm of group communication. We consider that the participants are structured in groups according to the multicast channel subscription. In our work we consider that a multicast channel defines a group communication and that a participant can belong to and communicate through more than one channel (overlapping channels, also indistinctly refered as overlapping groups in this paper). The multicast channels considered in this paper are characterized by a reliable and asynchronous communication.

Among the various existing causal algorithms that support multi-group environments, we can distinguish two types: symmetric and asymmetric ones. While in symmetric algorithms all participants share the same role in the system and interact freely without timing or centralized constraints, asymmetric algorithms either introduce restrictions in either the mode of interaction or execution.

The present paper proposes a symmetric Causal Multi-Channel Algorithm (CMCA). The main objective of our algorithm is to reduce the amount of control information (CI) per message. The amount of information necessary to guarantee causal communication in overlapping groups is $\theta(G \cdot N)$ in the worst case [5], where G is the number of groups and N refers to the number of participants in the system. For large values of N and G, the bound of θ($G\cdot N$) is prohibitively high. An original aspect of our work is the definition and the use of an extension of the *Immediate Dependency Relation* (IDR) [12] [13] to the multi-channel context. We call this extension the *Immediate Inter-Channel Dependency* (IICD). The IICD allows us, in a symmetric organization, to identify a sufficient control information to ensure the causal delivery of messages in a multi-channel scenario. Our CMCA algorithm can be viewed as an extension to the broadcast (one channel) causal algorithm presented in [13].

The paper proceeds as follows: first, in Section II we overview the related work. In Section III we give the basic definitions. Next in Section IV, we introduce the immediate dependency and our proposed extension to the multigroup case. In Section V, the proposed Causal Multi-channel Algorithm is shown in detail. Section VI is dedicated to the proofs, which include the proof of the IICD "completeness" property and the proof of correctness of the CMCA algorithm. In Section VII we present a comparison of our algorithm versus other important multi-group communication works. Finally, in Section VIII some conclusions are presented.

II. Related work

As mentioned above, we have surveyed works which aim to reduce the control information needed to ensure causal order delivery in symmetric algorithms. The symmetric algorithms consider that all participants share the same role and the same degree of responsibility in the system; furthermore, they interact freely without timing or centralized constraints. The symmetric category is concerned with identifying the necessary conditions to ensure a causal delivery of messages, and/or with arranging optimal coding to represent and transmit this information. The asymmetric category is composed of algorithms that assume a certain network topology [19], a particular group structure [2], and/or execution models [11]. The following related works involve only the symmetric category, which is the category with which we are concerned.

One of the first algorithms is the work done by Peterson [12]. Peterson introduced the *context graph*, which was designed to represent the causality between messages in a conversation algorithm in a broadcast environment (one group). This graph is directed and acyclic; its vertices correspond to the total set of messages, and the arcs represent the causal relationship between these messages. The reduced graph as presented in [12] shows that if the causal ordering of messages is ensured between every pair of immediate causal predecessor and successor messages, then the causal ordering among all messages will be automatically ensured due to the transitivity of the preceding relation.

The work of Prakash and Raynal [14] extends the immediate dependency property to support multi-party environments (no pre-defined groups exist). Prakash's work does not use nor maintain context graphs to ensure causal ordering. To ensure causal ordering, a message *m* needs to carry information concerning only those messages *m*' which its delivery is directly dependent upon. This approach is oriented to resolve problems in mobile agent applications; it considers multi-party and broadcast communication (one group), but not the multi-group communication case. The overhead for the causal multi-party algorithm is θ(N^2), where N is the number of participants in the system.

Another example of an algorithm is the CBCAST proposed by Birman [5][6], which is based on vector time clocks. To our knowledge, this algorithm is the only symmetric work that supports overlapping groups. The author, in his algorithm, mainly proposes to compress the vector time of a process by attaching only the local vector positions on each message:

1. that have changed since the last local message sent, and
2. that are not known to be *stable* (messages known to be received).

Due to its importance, we present a comparison of our work with this one in Section VIII.

Finally, the work proposed by Maddi and Dahamni [1] uses the immediate dependency property and the second constraint of the previous one. It sends less overhead than Prakash and Raynal [14] in the order of $(N^2\text{-}N\text{-}2)/2$; nevertheless, it uses a point-to-point communication scheme for multi-party environments. Consequently, the algorithm first needs to construct a different causal history for every destination before sending a message, and second, it needs to send n-copies of such message, one for each destination.

III. Preliminaries

Participants and Channels. The application under consideration is composed of a set of participants P that are structured into groups according to the multicast channel organization. We consider that a multicast channel defines a group communication; for this reason, in this paper we refer to them indistinctly. The participants communicate through multicast channels that are characterised by a reliable and asynchronous communication. We denote by C the set of channels, the mapping $Memb$: $C \rightarrow 2^P$ defines for each channel the group of connected participants, and the mapping $Conn$:$P \rightarrow 2^C$ defines for each participant the set of channels to which it is connected. Each participant maintains an integer local clock which is incremented each time it performs an action (*send* or *receive*).

A (finite) **behavior** of such a system is defined by a set of events executed by the participants, partially ordered by a "causal" relation defined below. These events may be emissions or deliveries of messages.

Messages. We consider a finite set of messages M, where each message $m \in M$ is identified by a tuple (participant, integer, channel), $m=(p,x,c)$ where $p \in P$ is the sender of m, denoted by $Src(m)$, x is the value of the local clock of p when m is sent, and $c \in C$ is the channel on which m is multicasted, denoted by $Chan(m)$. The set of destinations $Dest(m)$ of the message m is composed of the participants connected to the channel $Chan(m)$, $Dest(m)=Memb(Chan(m))$. In further sections, additional fields will be introduced to the tuple (p,x,c), but they are not relevant in this section.

Events. Let m be a message, we denote by $send(m)$ the emission event of m by $Src(m)$, and by $delivery(p,m)$ the delivery event of m to the participant p connected to $Chan(m)$. The set of events associated to M is then the set $E = \{send(m),\ m \in M\} \cup \{delivery(p,m),\ m \in M,\ p \in Dest(m)\}$. An emission event $send(m)$, where $m=(p,x,c)$, may also be denoted by $send(p,m)$ or $send(m,c)$ without ambiguity. The subset $E_p \subseteq E$ of events involving p is $E_p = \{send(m),\ p=Src(m)\} \cup \{delivery(p,m),\ p \in Dest(m)\}$.

Causal relation and causal order delivery. Causal order delivery is based on the causal relation $\rightarrow \subseteq E \times E$ between the events E of the system. This relation is also called the "happened before relation," and was first defined by Lamport [8]. The causal relation is a strict partial order (transitive and antisymmetric) defined as follows:

Definition 2. The causal relation "$\rightarrow$" is the least partial order relation on E satisfying the two following properties:
1) For each participant p, the restriction of $\rightarrow$ to the set of events E_p involving p is the total order in

which they occur: in particular, we have $e,e' \in E_p \Rightarrow e \rightarrow e' \vee e' \rightarrow e$
2) For each message *m* and destination *p* of *m*, the emission of *m* precedes its delivery to *p*:
$\forall m \in M,\ \forall p \in Dest(m) : send(m) \rightarrow delivery(p,m)$

The partial order $(E, \rightarrow)$ is usually represented by a directed acyclic graph where for each participant *p*, the events of E_p are drawn on a vertical line.
The complement of the causal relation is called concurrency: a pair of events *e*, *e*' is said to be *concurrent*, denoted $e||e'$, iff $\neg(e \rightarrow e' \vee e' \rightarrow e)$.

Causal order delivery in group communication presents two cases: the broadcast case (one channel) and the multi-channel case, which includes overlapping channels. The causal delivery for the broadcast case is defined as follows [7]:

Definition 3 Broadcast Causal Order Delivery (one channel):
If $send(m) \rightarrow send(m')$, then
$\forall p \in P : delivery(p,m) \rightarrow delivery(p,m')$

Causal order delivery ensures that if the diffusion of a message *m* causally precedes the diffusion of a message *m*', in a channel *c*, then the delivery of *m* precedes the delivery of *m*' to each participant *p*.

The case of causal delivery in a multi-channel environment is more common in channel communication. Two messages sent in different channels may not have the same sets of destinations. The definition of the causal order delivery takes this into account. The definition is as follows:

Definition 4 Multi-channel causal order delivery:
If $send(m,c) \rightarrow send(m',c')$, then
$\forall p \in Memb(c) \cap Memb(c') : delivery(p,m) \rightarrow delivery(p,m')$

Multi-channel causal order delivery guarantees that if the diffusion of a message *m* causally precedes the diffusion of a message *m*', with *c* and *c*' as the diffusion channels of messages *m* and *m*' respectively, then the delivery of *m* causally precedes the delivery of *m*' for all participants *p* that belong to both channels *c* and *c*' [12].

IV. The Immediate Dependency Relation

1. Single channel case (One group)

The Immediate Dependency Relation (IDR) is defined on the set of sending events. It is the transitive reduction of the causal precedence restricted to these events: it is included in the causal precedence and is the smallest relation which generates it by transitive closure.

Definition 5 Immediate dependency relation $\downarrow$ (IDR):
$send(m) \downarrow send(m') \Leftrightarrow [\ send(m) \rightarrow send(m') \wedge \forall\ m'' \in M, \neg\ (send(m) \rightarrow send(m'') \rightarrow send(m'))]$

Thus, *send*(*m*) directly precedes *send*(*m'*), iff *send*(*m*) causally precedes *send*(*m'*) and no other event *send*(*m''*) happens "causally" between *send*(*m*) and *send*(*m'*).

The causal precedence on sending events induces a partial order on the corresponding messages, denoted by the same symbol $\rightarrow$, defined as follows:

$$m \rightarrow m' \Leftrightarrow send(m) \rightarrow send(m').$$

The IDR can thus be viewed as a relation on messages denoted by the same symbol $\downarrow$, and we say that message *m* directly precedes message *m'*:

$$m \downarrow m' \Leftrightarrow send(m) \downarrow send(m').$$

This relation is important for causal delivery algorithms and protocols: the causal delivery of messages related by the IDR is a sufficient condition to ensure the causal delivery of all messages, as shown below:

Theorem 1 Causal broadcast delivery using the IDR relation:
If $\forall\ m, m' \in M,\ send(m) \downarrow send(m') \Rightarrow \forall p \in P: delivery(p,m) \rightarrow delivery(p,m')$
then $\forall\ m, m' \in M, send(m) \rightarrow send(m') \Rightarrow \forall p \in P : delivery(p,m) \rightarrow delivery(p,m')$

The **proof** (see also [12]), relies on the fact that for any pair $send(m) \rightarrow send(m')$ if $send(m) \downarrow send(m')$ does not hold, then we can exhibit a message m'' such that $send(m) \rightarrow send(m'') \rightarrow send(m')$. Using inductive reasoning and the fact that the event $send(m')$ may only have a finite number of "causes" or predecessors for the causal relation, we can find (at least) a sequence ($m_i=(k_i, x_i, c_i)$, $i=0...h$), such that $m=m_0$, $m'=m_h$ and for all $i=0...h-1$, $send(m_i) \downarrow send(m_{i+1})$. For any participant *p* we have $delivery(p,m_i) \downarrow delivery(p,m_{i+1})$, and by transitivity we get the required property.

Clearly, the causal delivery of messages related by IDR is not only a sufficient but also a *necessary* condition for the causal delivery of all causally related messages. From an algorithmic point of view, if the reference of some immediate predecessor *m'* of a message *m* is not piggy-backed with *m*, the causal delivery of *m* with respect to *m'* may fail. Theorem 1 shows that this information is sufficient.

However, in the general case where messages may be sent to different sets of destinations, the causal delivery of a message only with respect to its immediate predecessors is not sufficient to ensure the global property. This is the particular case in our framework of multiple overlapping channels, as shown in the example of subsection 3 below. The next section presents a relation which extends IDR, and which benefits of the same type of properties in this more general framework.

2. Multi-channel case (Multi-group)

In this section, we extend the principle of immediate dependency to the case of multi-channel diffusion. We call this extension the *Immediate Inter-Channel Dependency Relation*, and refer to it by its acronym IICD. In the same way that the IDR allows us to define the minimal sufficient control information in a broadcast case, the IICD allows us to characterize a minimal sufficient control information to ensure the causal delivery of messages in a multi-channel environment.

Henceforth, we will often denote $send(m,c)$ by (m,c).

Definition 6 Immediate Inter-channel Dependency Relation (IICD) $\uparrow$:
$(m,c) \uparrow (m',c') \Leftrightarrow [((m,c) \rightarrow (m', c')) \wedge \forall\ m'' \in M, ((m,c) \rightarrow (m'', c'') \rightarrow (m', c') \Rightarrow c'' \neq c \wedge c'' \neq c')]$

As for IDR, this relation can be viewed as a relation on messages: if c and c' are the channels of m and m', we have $m \uparrow m' \Leftrightarrow (m,c) \uparrow (m',c')$.

By this definition, a message m multicasted on channel c has an immediate dependency relation with a message m' multicasted on c', if m' causally depends on m, i.e. $(m, c) \rightarrow (m', c')$, and if for any intermediate message (m'', c'') such that $(m, c) \rightarrow (m'', c'') \rightarrow (m', c')$, the channel c'' differs from the channels c and c'.
Note that the definition 5 of the IDR relation applies as well in the multiple channel context, viewing for any message m, $send(m)$ as a shortcut for $send(m,c)$ with $c=Chan(m)$. As a direct consequence of the definitions, the IICD relation contains the IDR relation: for any messages m and m' sent on channels c and c', we have that $send(m) \downarrow send(m') \Rightarrow send(m,c) \uparrow send(m',c')$. Furthermore the two relations coincide if C contains a single channel : if $C = \{c\}$, then $send(m) \downarrow send(m') \Leftrightarrow send(m,c) \uparrow send(m',c)$.

Clearly, the causal delivery of messages related by the IICD relation is a *necessary* condition for the causal delivery of all messages. As shown in the 3-channel example of subsection 3 below, if the reference of some IICD predecessor m' of a message m is not piggy-backed with m, the causal delivery of m with respect to m' may fail.
This information needed to ensure the causal delivery of IICD related messages is also *sufficient*: The following proposition proved in section VII establishes that if any two messages related by IICD are delivered in causal order, then all messages are delivered in causal order.

Theorem 2 (Proof in section VII)
If $\forall\ send(m,c), send(m',c') \in E,\ send(m,c) \uparrow send(m',c') \Rightarrow \forall p \in Memb(c) \cap Memb(c')$:
$delivery(p,m) \rightarrow delivery(p,m')$
then $\forall \mathrm{send}(m,c), \mathrm{send}(m',c') \in E, \mathrm{send}(m,c) \rightarrow \mathrm{send}(m',c') \Rightarrow \forall p \in Memb(c) \cap Memb(c')$:
$delivery(p,m) \rightarrow delivery(p,m')$

Our algorithm is based on the tracking of the IICD relation between messages during an execution, and on a delivery of messages which respects that ordering.

3. Illustration of immediate inter-channel dependency

In order to better illustrate Definition 6 and Theorem 2, we present the following scenario example.

Scenario: The Multi-channel diagram in Figure 1 is composed of $c_1=\{ p_1, p_4, p_5, p_2\}$, $c_2=\{ p_2, p_3\}$ and $c_3=\{ p_1, p_3\}$ where p_4, p_5 are local participants to channel c_1. Consider the emission of message m_4, such that $((m_2, c_1)||(m_3, c_1)) \uparrow (m_4, c_3)$. According to Definition 6, the messages which have an IICD with m_4 are messages m_2 and m_3. Therefore, the information that corresponds to these messages is sent as control information to m_4. This information is not taken into account for the delivery of m_4 by participant p_3 since $p_3 \notin Memb(c_1) \cap Memb(c_3)$.

Participant p_3 only uses this information to update his system history file and for future diffusion of messages, as in the case of the diffusion of message m_5.

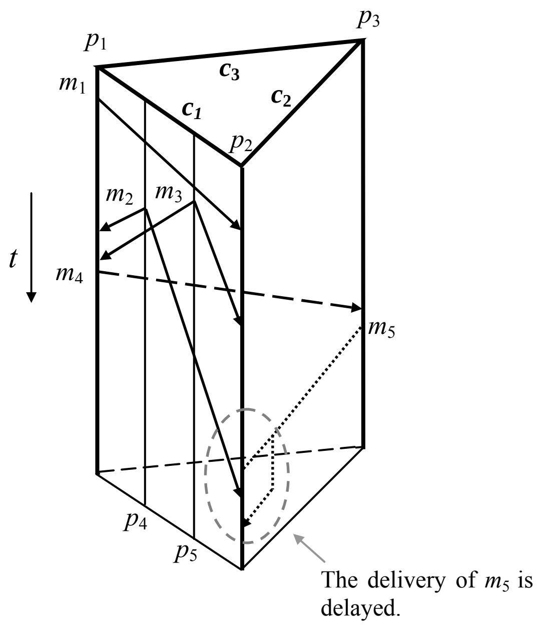

Figure. 1 Multi-channel scenario

Let us now consider message m_5 (Figure 1). At the moment of diffusion of m_5, we apply Definition 6 to each message in the causal history of p_3. We find that the messages that have an IICD with m_5 are m_2, m_3 and m_4 (see Figure 2). Thus, as in the previous cases, the control information timestamped to message m_5 corresponds to the messages which have an IICD to m_5. Message m_5 is delivered to p_2 ($p_2 \in$ *Memb*(c_1) ∩ *Memb*(c_2)) only after messages m_2 and m_3 have been delivered. These messages, m_2 and m_3, are delivered to p_2 only after message m_1 has been delivered. This is ensured by the Immediate Dependency Relation (IDR).

$$\begin{tikzcd}
(m_2, c_1) \arrow[dr] & & \\
 & (m_4, c_3) \arrow[r] & (m_5, c_2) \\
(m_3, c_1) \arrow[ur] & &
\end{tikzcd}$$

Figure. 2 IICD graph for m_5

V. The causal multi-channel algorithm

1. Local identifiers

In this algorithm, a participant locally manages an identifier for each channel to which it belongs. The distribution of the participants in the channels is transparent to the participants. This means that a given participant is aware only of a list of identifiers and not of the channels′ constitution. For example, in Figure 3, participants p_1 and p_2 interact through channels c_2 and c_3 with participant p_3. Nevertheless, neither one of them knows that both are interacting with the same participant.

We recall that P is the set of participant identifiers, C the set of channels, the mappings $Memb : C \rightarrow 2^P$ and $Conn : P \rightarrow 2^C$ define for each channel the set of connected participants, and for each participant, the set of channels to which it is connected.

Let I be an initial set of integers disjoint from P. The mapping $id : P\times C\rightarrow I$ associates to each participant p and each channel $c\in Conn(p)$ a unique identifier $id(p,c) \in I$. For any $i\in I$, we denote by $ch(i)$ the unique channel c such that $i=id(p,c)$ for some p, and we have $i=id(p,c) \Rightarrow ch(i)=c$. For example, in Figure 3, participant p_1 has two identifiers denoted by $i=id(p_1,c_1)$ and $i'=id(p_1,c_3)$; one for each channel to which it belongs, where $ch(i)= c_1$ and $ch(i')= c_3$, respectively.

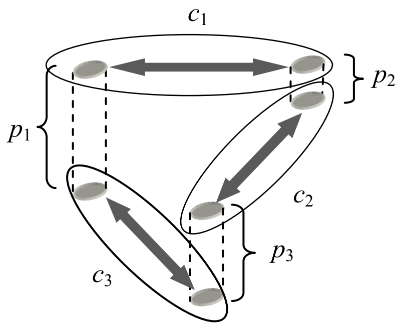

Figure. 3 Connection scheme

2. Data structures

Local states: The state of a participant p is defined by two data structures: $VT(p)$ and $CI(p)$.
- $VT(p)$ is the vector time. For each participant q and each channel $c\in Conn(q)$, there is an element $VT(p)[j]$ where $j=id(q,c)$ The size of VT is thus equal to the sum of all channel sizes $\Sigma_{c\in C} |Memb(c)|$.
- $CI(p)$ is the control information structure. It is a set of entries $ci_{k,t,c}= (k,t,c,ch\_dests)$ where (k,t,c) is a message identifier (the message diffused by the participant p_k at its local clock value t, in the channel $c\in Conn(p_k)$) and ch_dest is a set of channel identifiers explained below.

The information in the vector time $VT(p)$ contains the local view which the participant p has of the causal history of the system. In particular, the element $VT(p)[j]$ represents the greatest message number from the identifier j and 'seen' by p. It has a *total* view if, at a given instant t, it contains information of the last known message from each participant. It has a *partial* view if it contains this information only for a subset of the participants. It is through the $VT(p)$ structure that we are able to guarantee the causal delivery.

The structure $CI(p)$ also contains information about the causal history of p. The information in $CI(p)$, at any moment, is a partial copy of $VT(p)$, i.e. for each entry (k, t, c, ch_dests) of $CI(p)$, we have $t=VT(p)[k]$ (See Lemma 2 : $(k, t, c) \in CI(p)\Rightarrow x=VT(p)[k]$). The set ch_dest contains the channels where the identifier (k,t,c) (identifier of a message in the causal history of p) may not have been already broadcasted: when p multicasts a new message m in one of the channels of ch_dest, then m must carry the information (k, t, c) in its field $H(m)$ to ensure its causal delivery.

Messages: A message m is composed of an identifier (k,t,c) and an attached causal information $H(m)$. Formally, a message m is a tuple $m = (k, t, c, H(m)))$, where:
- k is the identifier of the sender $p=Src(m)$ for the channel c, i.e. $k=id(p,c)$
- $t=VT(p)[k]$ is the logical clock of p for k when m is sent
- c is the channel in which m is broadcasted, $c=Chan(m)$

- $H(m)$ is a set of tuples (i, t, c) representing messages
The structure $H(m)$ contains identifiers of messages causally preceding the message m and needed for the causal delivery of m. The structure $H(m)$ is built before the message is broadcasted and attached to it.

Note. The following nomenclature is used in the algorithm: i , j , k and l represent channel member identifiers; t, x and y are logical clocks; c and d are diffusion channels; and lastly, C is the set of channels in the system.

3. Algorithm specification

I. Initially,

```
1.  $VT(p)[k] = 0 \ \forall\ k{:}1\ldots\Sigma_{g\in G} |g|$
2.  $CI(p) \leftarrow \varnothing$
```

II. For each *message diffused* by p into channel *c* with $i=id(p,c)$

```
3.  $VT(p)[i] = VT(p)[i] + 1$
4.  $H(m) \leftarrow \varnothing$
5.  for all $ci \in CI(p)$: $ci=(k, x, d, ch\_dests)$
6.      if $c \in ci.ch\_dests$ then
7.          $H(m) \leftarrow H(m) \cup \{(k,x,d)\}$
8.          $ci.ch\_dests \leftarrow ci.ch\_dests \setminus c$  /* erase $c$ from $ci$ */
9.      endif
10.     if $ci.ch\_dests = \varnothing$ then
11.         $CI(p) \leftarrow CI(p) \setminus ci$      /* erase $ci$ from $CI(p)$ */
12.     endif
13. endfor
14. $t=VT(p)[i]$
15. $m=(i, t, c, content, H(m))$
16. send($m$) into the channel $c$
17. $CI(p) \leftarrow CI(p) \cup \{(i,t,c,Conn(p) \setminus c)\}$  /* adds a new $ci$ to $CI(p)$ */
```

III. For each $m = (i, t, c, content, H(m))$ received by p with $j = id(p,c)$

```
18. If not $t=VT(p)[i]+1 \wedge$   /*To verify causal delivery*/
19. $\forall (l,x,d) \in H(m) : d \in Conn(p) \Rightarrow x \leq VT(p)[l]$ then
20.     wait
21. else
22.     Delivery($content$)
23.     $VT(p)[i] = VT(p)[i]+1$
24.     if ($\exists x{:}(i,x,c) \in CI(p)$) then
25.         $CI(p) \leftarrow CI(p) \setminus \{(i,x,c)\}$
26.     endif
27.     $CI(p) \leftarrow CI(p) \cup \{(i,t,c,Conn(p))\}$
28.     for all $(l,x,d) \in H(m)$
29.         if ($d \in Conn(p)$) then
```

```
30.           if ($\exists y$:($l,y,d$) $\in CI(p)$) then/*$x \le y$*/
31.             if $x = y$ then
32.               $UPD(ci_{l,y,d}, c)$
33.             endif
34.           endif
35.         else /* $d \notin Conn(p)$ */
36.           if ($\exists y$:($l,y,d$) $\in CI(p)$) then
37.             if $x = y$ then
38.               $UPD(ci_{l,y,d}, c)$
39.             endif
40.             if $x > y$ then
41.               $VT(p)[l] = x$
42.               $CI(p) \leftarrow CI(p) \setminus \{(l,y,d)\}$
43.               $CI(p) \leftarrow CI(p) \cup \{(l,y,d, Conn(p))\}$
44.             endif
45.           else /* $\neg \exists y$:($l,y,d$) $\in CI(p)$) */
46.             if ($VT(p)[l] < x$) then
47.               $VT(p)[l] = x$
48.               $CI(p) \leftarrow CI(p) \cup \{(l,\mathrm{x},d, Conn(p))\}$
49.             endif
50.           endif
51.         endif
52.     endfor
53. endif
```

IV. Updating

```
54. $UPD(ci_{k,x,,d}, c)$
55.   if ($c \neq d$) then
56.     $ci_{k,x,d}.ch\_dests \leftarrow ci_{k,x,d}.ch\_dests \setminus c$
57.       if $ci_{k,x,d}.ch\_dests = \emptyset$ then
58.         $CI(p) \leftarrow CI(p) \setminus \{(k,x,d)\}$
59.       endif
60.   else /* $c = d$ */
61.       $CI(p) \leftarrow CI(p) \setminus \{(k,x,d)\}$
62.   endif
```

4. Example scenario

The multi-channel scenario presented in Figure 1 is composed of $c_1 = \{p_1, p_4, p_5, p_2\}$, $c_2 = \{p_2, p_3\}$, $c_3 = \{p_1, p_3\}$. The *VT* positions are assigned as follows:

Participants	(p_1, c_1)	(p_1, c_3)	(p_2, c_1)	(p_2, c_2)	(p_3, c_2)	(p_3, c_3)	(p_4, c_1)	(p_5, c_1)
VT Positions	*VT*[1]	*VT*[2]	*VT*[3]	*VT*[4]	*VT*[5]	*VT*[6]	*VT*[7]	*VT*[8]

Henceforth, we use the *VT* positions as participant identifiers to send messages in the respective channels. Before sending m_5, we have at p_3 $CI(p_3)=\{(7,1, c_1, \{c_2, c_3\}), (8,1, c_1, \{c_2, c_3\}),(2,1, c_3, \{c_2, c_3\})\}$ $VT(p_3)=(0,1,0,0,0,0,1,1)$. At p_2, we have $CI(p_2)=\{(8,1, c_1, \{c_2, c_3\})$ and $VT(p_2)=(1,0,0,0,0,0,0,1)$.

Sending message m_5 **in channel** c_2. First, at participant p_3, the $VT(p_3)[5]$ position is increased by one, and the $H(m_5)$ is constructed as follows (lines 5-13). We verify for each entry $ci=(k,x,d,ch\_dests)\in CI(p_3)$ if c_2 belongs to *ci.ch_dests*. If this is the case, we attach the entry (k,x,d) to $H(m_5)$. Each entry in $H(m_5)$ identifies messages IICD related to m_5. At the same time that $H(m_5)$ is constructed, the $CI(p_3)$ structure is updated. First, channel c_2 is erased from *ci.ch_dests*, and then, if *ci.ch_dests* is empty, the entire *ci* entry is erased from $CI(p_3)$. In this way, a participant never diffuses redundant information. Finally, we construct message $m_5=(5,1,c_2,content,\{(7,1,c_1),(8,1,c_1), (2,1,c_3)\})$ and proceed to send it.

Receiving message m_5 **at participant** p_2. Participant p_2 verifies the condition delivery of line 18 as follows. First, we verify that m_5 satisfies the FIFO delivery, and then we continue to verify the information on $H(m_5) = \{(7,1,c_1), (8,1,c_1), (2,1,c_3)\}$. In this case, the FIFO condition is satisfied ($VT(p_2)[5]+1$ is equal to 1). We then take the first entry $(7,1,c_1)\in H(m_5)$ and find that it does not satisfy the delivery condition because $VT(p_2)[7]\neq 1$. This means that p_2 is missing some message from p_5 that causally precedes m_5. Therefore, message m_5 cannot be delivered yet; instead, it is placed on hold. Message m_5 will only be delivered after the reception of all message(s) that causally precede it.

VI. Proofs

1. Proof of the IICD completeness property

First, we introduce a " causal" distance between messages which is used in the present section.

Let m and m' be messages, the distance $d(m,m')$ is defined for any pair m and m' such that $send(m)\rightarrow send(m')$: $d(m,m')$ is the greatest integer n, such that for some sequence of messages (m_i, $i= 0...n$) with $m= m_0$ and $m'=m_n$, we have $send(m_i) \rightarrow send(m_{i+1})$ for all $i =0...n-1$. Due to the acyclic nature of the causal order and a sensible "finite causes" property of a behavior, such a greatest integer always exists for any pair m,m' such that $send(m)\rightarrow send(m')$. Notice that we consider a *greatest* sequence's length because it is useful in the proofs, while a shortest sequence length is usually considered to define distances in graphs.

Theorem 2
If $\forall\ m,m'\in M$, $\forall\ c,c'\in Chan$: $send(m,c) \uparrow send(m',c') \Rightarrow \forall p \in Memb(c) \cap Memb(c')$:
$delivery(p,m) \rightarrow delivery(p,m')$
then $\forall\ m,m'\in M$, $\forall\ c,c'\in Chan$: $send(m,c)\rightarrow send(m',c') \Rightarrow \forall p \in Memb(c) \cap Memb(c')$:
$delivery(p,m) \rightarrow delivery(p,m')$

Proof: Let m and m' be such that $send(m)\rightarrow send(m')$. We show that $\forall p \in Dest(m) \cap Dest(m')$: $delivery(p,m) \rightarrow delivery(p,m')$. By hypothesis, this property is satisfied if $send(m) \uparrow send(m')$. Otherwise, let $d(m,m') = n$, $c=Chan(m)$ and $c'=Chan(m')$. In that case, by definition of the IICD $\uparrow$, we can find a message $m''=(k'', x'', c'')$ such that $send(m) \rightarrow send(m'') \rightarrow send(m')$ and such that $Chan(m'')=c$ or $Chan(m'')=c'$. If we have not $send(m)\uparrow send(m'')$ and $send(m'')\uparrow send(m')$, we can repeatedly find new intermediate messages and constitute a sequence ($m_i=(k_i, x_i, c_i)$, $i=0...h$), such that $m=m_0$, $m'=m_h$ and for all $i=0...h-1$, $send(m_i)\rightarrow send(m_{i+1})$. Due to the finite distance $d(m,m') = n$, the size of such sequence is bounded by n, and thus, in a finite number of steps, we build a sequence such that for all $i=0...h-1$, $send(m_i)\uparrow send(m_{i+1})$. Furthermore, by construction we have $Chan(m_i)\in\{c,c'\}$ for all $i=0...h$, and there is an integer $0\leq k\leq h-1$ such that $Chan(m_i)=c$ for all $0\leq i\leq k$ and $Chan(m_i)=c'$ for all $k<i\leq h$. Let a common recipient $p \in Dest(m) \cap Dest(m')$; this means that $p \in Memb(c)\cap Memb(c')$, and thus, p receives all the messages of the sequence ($m_i=(k_i, x_i, c_i)$, $i=0...h$). Due to the hypothesis, $send(m_i)\uparrow send(m_{i+1})$ implies $delivery(p, m_i)\rightarrow delivery(p, m_{i+1})$ for all $i=0...h-1$, and this induces that $delivery(p,m) \rightarrow delivery(p,m')$. □

2. Proof of correctness (1)

In this section we show that the delivery restrictions imposed by the algorithm do not exceed the causal delivery constraints. No delivery order is imposed to messages which are not causally dependent on each other. Precisely, Theorem 3 shows that if a message piggybacks some causal information about some other message, then the latter belongs to the causes of the former.

Theorem 3 For any two messages $m=(k,x,c)$ and m' then:
$(k,x,c) \in H(m') \Rightarrow send(m) \rightarrow send(m')$

Proof: Let $p= Src(m')$, if $(k,x,c) \in H(m')$, due to instruction lines 5 and 7, $(k,x,c) \in CI(p)$ when m' is sent by participant p. There are two cases:
A) The element (k,x,c) is added to $CI(p)$ by instruction line 27 when m is delivered to p. This must have occurred prior to the emission of m' and thus we have $delivery\ (m) \rightarrow send(m')$, and thus $send(m) \rightarrow send(m')$.
B) The element (k,x,c) is added to $CI(p)$ by instruction lines 43 or 48 when some message $m_1=(k_1,x_1,c_1) \neq m'$, such that $(k,x,c) \in H(m_1)$ was delivered to p. In that case, we have $delivery(p,m_1) \rightarrow send(m')$.

Cases A and B show that if $(k,x,c) \in H(m')$, then either $send(m) \rightarrow send(m')$ or for some $m_1=(k_1,x_1,c_1) \neq m'$ we have $(k,x,c) \in H(m_1)$ and $send\ (m_1) \rightarrow send(m')$. If $send(m) \rightarrow send(m')$ does not hold, then case B holds and we can apply the same deduction to the messages m and m_1 because $(k,x,c) \in H(m_1)$. We have $m_1 \neq m$ because otherwise $send(m) \rightarrow send(m')$. If $send(m) \rightarrow send(m_1)$ does not hold, we iterate the step and exhibit a message $m_2 \neq m$ such that $send(m_2) \rightarrow send(m_1)$ and $(k,x,c) \in H(m_2)$. At step $i-1$, we have $(k,x,c) \in H(m_{i-1})$, for some message $m_{i-1} \neq m$ such that $send(m_{i-1}) \rightarrow send(m_{i-2})$, and if $send(m) \rightarrow send(m_{i-1})$ does not hold, we can find a message $m_i \neq m$ such that $send(m_i) \rightarrow send(m_{i-1})$.... $send(m_1) \rightarrow send(m')$ and $(k,x,c) \in H(m_i)$. Because such a sequence may not be infinite, for some step i and message m_i we must have $send(m) \rightarrow send(m_i)$, and the property $send(m) \rightarrow send(m_i) \rightarrow send(m_{i-1})$.... $send(m_1) \rightarrow send(m')$ concludes the proof. □

The following Lemma is a consequence of the previous theorem and used in the next section.

To make the next proofs clearer, we give some definitions.

Due to the total order of the actions of a single participant p, its behaviors can be modeled by alternated sequences of states and events $s_0, e_1, s_1, e_2, s_2 ,.. ,e_n, s_n$, where the states s_i are values of $VT(p)$ and $CI(p)$, and where each event e_i is an action *send*(*p,m*) or *delivery*(*p,m*) for some message *m*, and represents the associated sequence of instructions. The *causal past* of an event $e \in E$, denoted $\downarrow e$, is the set of the events which precede *e* for the relation $\rightarrow$, i.e. $\downarrow e=\{e' \in E,\ e' \rightarrow e\}$. The *immediate cause* of a state *s* of a participant *p*, denoted $^{\circ}s$, is the unique last event which occurred on *p* before reaching the state *s*, for instance $e_i={}^{\circ}s_i$ in the sequence $s_0, e_1, s_1, e_2, s_2 ,.. ,e_n, s_n$. The *causal past* of a state s, denoted $\downarrow s$ is the set of events composed of its immediate cause $^{\circ}s$ and the causal past of $^{\circ}s$, formally $\downarrow s = \{^{\circ}s\} \cup \downarrow(^{\circ}s)$.

Lemma 1 If a state *s* of a participant *p* satisfies $VT(p)[l] = x \neq 0$, then the event *send*(*m*) where $m=(l,x,c)$ with $c = ch(l)$, is in the past $\downarrow s$ of *s*.
Proof: There are two cases, depending on whether *p* is connected to *c* or not.
1. $p \in Memb(c)$. In that case, $VT(p)[l]$ may be modified and set to *x* only on the delivery to *p* of the message $m=(l,x,c)$ received from *l* on the channel *c* (instruction line 23). The event *delivery*(*p,m*) then belongs to $\downarrow s$ as well as the corresponding emission event *send*(*m*).
2. $p \notin Memb(c)$. In that case, $VT(p)[l]$ may be modified and set to *x* only on the reception by *p* of a message *m'* such that $(l,x,c) \in H(m')$ (lines 41 and 47). The delivery event *delivery*(*p,m'*) then belongs to $\downarrow s$ and also the corresponding emission event *send*(*m'*). Using Theorem 3, we can conclude that $send(m) \rightarrow send(m')$, and that *send*(*m*) is also a cause of s (*send*(*m*) belongs to $\downarrow s$). □

3. Proof of correctness (2)

The second proof of correctness shows that whenever two messages *m* and *m'* are such that $send(m) \rightarrow send(m')$, we have $delivery(p,m) \rightarrow delivery(p,m')$ for any common destination $p \in Dest(m) \cap Dest(m')$. Due to Theorem 2, it is sufficient to prove this property for messages *m* and *m'* in immediate inter-channel dependency, and indeed Theorem 4 shows that if $send(m) \uparrow send(m')$, we have $delivery(p,m) \rightarrow delivery(p,m')$ for any common destination $p \in Dest(m) \cap Dest(m')$.

The following four Lemmas are used in the proof of Theorem 4. Lemma 2 ensures that for each entry (l,x,c) in $CI(p)$ we have $x = VT(p)[l]$.

Lemma 2 $(l,x,c) \in CI(p) \Rightarrow x = VT(p)[l]$

Proof: There are two cases:
1) $c \in Conn(p)$: the element (l,x,c) is added to $CI(p)$ either at line 17 on sending the message (l,x,c), and instruction line 3 ensures that $x = VT(p)[l]$, or on delivery of the message (l,x,c) at line 27, and when this instruction is executed, test line 18 and line 23 ensure that $x = VT(p)[l]$.
2) $c \notin Conn(p)$: the element (l,x,c) is added to $CI(p)$ due to line 43 (resp. at line 48), and the instruction previously executed at line 41, (resp. at line 47) is precisely the instantiation $VT(p)[l] = x$. □
The following Lemma shows that the values of the array $VT(p)$ are modified accordingly with the values of $H(m)$ by the algorithm when a message *m* is delivered.

Lemma 3 After the delivery of a message *m* to the participant *p*, for each entry $(l,x,c) \in H(m)$ we have $VT(p)[l] \geq x$.

Proof: There are two cases:
1) $c \in Conn(p)$: The condition for the delivery of m at line 19 is precisely $VT(p)[l] \geq x$.
2) $c \notin Conn(p)$: The code beginning at line 35 is executed. If condition line 36 is true, either $(l,y,c) \in CI(p)$ for some $y \geq x$, in which case, due to Lemma 2, we have $VT(p)[l]=y \geq x$, or the instruction line 41 is executed yielding $VT(p)[l] = x$. Otherwise, condition line 36 is false, execution starts at line 45 and the lines 46 and 47 ensure that $VT(p)[l] \geq x$. □

Lemma 4 If at state s of a participant p we have $VT(p)[l] = x \neq 0$ with $c=ch(l)$, then for any $c' \in Conn(p)$ one of the two following cases holds:
1) There exists an entry $(l,x,c) \in CI(p)$ such that $c' \in (l,x,c).ch\_dest$.
2) There exists a message m' with $c'=Chan(m')$ and an event $e=send(p,m',c')$ or $e=delivery(p,m',c')$ in the past of s, such that $(l,x,c) \in H(m')$.

Proof: There are two cases:
1) $c \in Conn(p)$: the value of $VT(p)[l]$ is set to x by instruction line 23 and the entry (l,x,c) is added to $CI(p)$ with $(l,x,c).ch\_dest=Conn(p)$ at line 27. The channel $c' \in Conn(p)$ is suppressed from $(l,x,c).cd\_dest$ only in two cases:
- when a message m' with $(l,x,c) \in H(m')$ is emitted by p to the channel c' (line 8); in that case, $send(p, m', c')$ is in the past of s,
- or when a message m' with $(l,x,c) \in H(m')$ is delivered to p through the channel c' (UPD line 32); and in that case, $delivery(p,m',c')$ belongs to $\downarrow s$.
2) $c \notin Conn(p)$: the value of $VT(p)[l]$ is set to x by line 41 (or line 47); and in that case, the entry (l,x,c) is added to $CI(p)$ with $(l,x,c).ch.\_dest=Conn(p)$ by line 43 (resp. line 48). The channel $c' \in Conn(p)$ is suppressed from $(l,x,c).cd\_dest$ only in two cases :
- either when a message m' with $(l,x,c) \in H(m')$ is emitted by p to the channel c' (line 8); in that case, $send(p, m', c')$ is in the past of s,
- or when a message m' with $(l,x,c) \in H(m')$ is delivered to p through the channel c' (UPD line 38); and in that case, $delivery(p,m',c')$ belongs to $\downarrow s$. □
The following lemma is a direct consequence of the previous one and of Theorem 3.

Lemma 5 If a participant p such that $VT(p)[l] = x \neq 0$ where $c = ch(l)$, executes the action $send(m',c')$, then one of the two following cases holds:
1) $(l,x,c) \in H(m')$ or
2) There is a message m'' broadcasted to the same channel $c'= Chan(m') = Chan(m'')$ such that $(l,x,c) \in H(m'')$ and $send(m'',c') \rightarrow send(m',c')$. Furthermore, the message $m=(l,x,c)$ is such that $send(m,c) \rightarrow send(m'',c') \rightarrow send(m',c')$.

Proof: If $(l,x,c) \notin H(m')$ when $send(m',c')$ occurs**,** then due to lines 6 and 7, no entry $(l,x,c) \in CI(p)$ exists such that $c' \in (l,x,c).ch\_dest$. Then, by the previous Lemma 4 and Theorem 3, we have $send(m,c) \rightarrow send(m',c')$. □

Theorem 4 $\forall\ m,m' \in M$, $send(m) \uparrow send(m') \Rightarrow \forall p \in Dest(m) \cap Dest(m')$:
$delivery(p,m) \rightarrow delivery(p,m')$.

Proof: We prove by induction on the distance $d(m,m')$ between m and m'. Recall that $d(m,m')$ is defined for any pair of messages m and m' such that $send(m) \to send(m')$, and is the greatest integer n such that for some sequence of messages (m_i, $i = 0 \ldots n$) with $m = m_0$ and $m' = m_n$, we have $send(m_i) \to send(m_{i+1})$ for all $i = 0 \ldots n-1$.

If $d(m,m')=1$, then by definition $send(p,m) \downarrow send(p',m')$ (note that this is the IDR). We have to check that $(l,x,c) \in H(m')$ The proof depends on whether both messages are sent by the same sender or not and relies on Lemma 5:

a) If $p=p'$, and $m=(l,x,c)$, then $VT(p)[l] = x$ with $l=id(p,c)$ after $send(m,c)$ (line 14).
b) If $p \neq p'$, due to the definition of $\to$ we must have have $delivery(p',m) \to send(p',m')$. Let $m=(l,x,c)$, we have $VT(p)[l] = x$ with $l=id(p',c)$ after $delivery(p',m)$ (lines 18 and 23).

By Lemma 5 we must have $(l, x, c) \in H(m')$ (case 1 of lemma 5) because case 2 contradicts the hypothesis $send(p,m) \downarrow send(p',m')$. This implies that for any $q \in Dest(m) \cap Dest(m')$, the delivery of m to q must occur before the delivery of m', and we have $delivery(q,m) \to delivery(q,m')$.

Induction step *n*. We suppose the property true for any messages m and m' such that $d(m,m') < n$, and show that it holds if $d(m,m') = n$. Let m and m' be such that $send(m) \uparrow send(m')$ and $d(m,m') = n$. Because $send(m) \to send(m')$ there is (at least) a message sequence ($m_i=(k_i, x_i, c_i)$, $i=0 \ldots h$), such that $m=m_0$, $m'=m_h$ and for all $i=0 \ldots h-1$, $send(m_i) \downarrow send(m_{i+1})$.

By definition of $d(m,m')$ we have $h \leq d(m,m')$. Furthermore, due to the hypothesis $send(m) \uparrow send(m')$, we also have $c_0 \neq c_i \neq c_h$ for all $i=1 \ldots h-1$.

For all $i=0 \ldots h$, we denote by p_i the participant such that $k_i = id(p_i, c_i)$ and by s_i the state of p_i at which the event $send(p_i, m_i)$ occurs. We show that for all $i=0 \ldots h$, the state s_i satisfies the property $VT(p_i)[k_0] \geq x_0$. The proof is by induction on the index i.

At state s_0, when $send(k_0, x_0, c_0)$ occurs, due to instruction line 4 we must have $VT(p_0)[k_0] = x_0$.
Suppose that $VT(p_i)[k_0] \geq x_0$ at state s_i, we show that $VT(p_{i+1})[k_0] \geq x_0$ at state $s_{i+1.}$ If $VT(p_i)[k_0] = y \geq x_0$ when m_i on c_i occurs, by Lemma 5 only two cases may take place:
- $(k_0, y, c_0) \in H(m_i)$. In that case, due to Lemma 3, we have $VT(p_{i+1})[k_0] = y \geq x_0$ after $delivery(p_{i+1},m_i)$, and this remains true at state $s_{i+1.}$ when the broadcast of m_{i+1} to c_{i+1} occurs.
- There is a message $m= (k, x, c_i)$ multicasted on c_i, such that $(k_0, y, c_0) \in H(m)$ and $send(k_0, y, c_0) \to send(m) \to send(m_i)$. Due to $x_0 \leq y$, we have $send(k_0, x_0, c_0) \to send(m)$, and because $c_i \neq c_0$, we have $m \neq m_0$. The distance $d(m,m_i)$ is strictly lower than $d(m_0,m_i)$, thus strictly lower than $n=d(m_0,m_h)$. We can apply to m and m_i the induction hypothesis and conclude that $delivery(p_{i+1},m) \to delivery(p_{i+1},m_i)$. Due to the fact that $(k_0, y, c_0) \in H(m)$ with $y \geq x_0$, we have $VT(p_{i+1})[k_0] = y \geq x_0$ after $delivery(p_{i+1},m)$ and thus also after $delivery(p_{i+1},m_i)$, in particular at state s_{i+1} when the multicast of m_{i+1} to c_{i+1} occurs.

Using the induction hypothesis, we have shown that for any messages $m=(k,x,c)$ and $m'=(k',x',c')$, such that $send(m) \uparrow send(m')$ and $d(m,m') = n$, we have $VT(p')[k] \geq x$ when $send(p',m')$ occurs. Let $y = VT(p')[k] \geq x$, we can conclude that $(l,y,c) \in H(m')$: otherwise, due to Lemma 5, there would be a message m'' broadcasted to the channel $c'=Chan(m')=Chan(m'')$ such that $send(m,c) \to send(m'',c') \to send(m',c')$, and this contradicts the hypothesis $send(m) \uparrow send(m')$. For any participant $q \in Dest(m) \cap Dest(m')$, the property $(l,y,c) \in H(m')$ ensures that the delivery of m to q precedes the delivery of m', and we have $delivery(q,m) \to delivery(q,m')$. □

The correctness of the algorithm results from Theorems 3 and 4.

Corollary:
For any messages m and m' :
If $send(m) \rightarrow send(m')$, then $\forall p \in Dest(c) \cap Dest(c')$: $delivery(p,m) \rightarrow delivery(p,m')$

VII. The CMCA vs other algorithms

Table 1. Comparison of causally ordered multi-group algorithms

Causal Multi-group Algorithms	**Configuration**	**Organization**	**Diffusion**	**Overhead**	**Principle**
CMCA (our algorithm)	Dynamic	Symmetric	Asynchronous	(worst case) $\Sigma\lvert g\rvert : g \in G$	Immediate Inter-Channel Dependency Relation
CBCAST (Isis)	Dynamic	Symmetric	Asynchronous	(worst case) $\Sigma\lvert g\rvert : g \in G$	Time clock compression
Daisy Architecture	Dynamic	Asymmetric	Asynchronous	(worst case) $(l \cdot (\lvert S\rvert+1)) \cdot (\lvert HS\rvert+1)$	Logical group structure / Rediffusion servers
Two-layered group	Dynamic	Asymmetric	Asynchronous	(worst case) $(l_i+k)l_i+k^2$	Logical group structure / Rediffusion servers
Causal Separators	Static	Asymmetric	Asynchronous	(worst case) $\Sigma\lvert L\rvert : L \in S$	Network topology / Rediffusion servers
Low cost approach	Dynamic	Asymmetric	Synchronous	(always) $\lvert G\rvert$	Synchronous execution by phases

In this section we compare the characteristics of our CMCA algorithm with other main existing multi-group algorithms that consider overlapping groups (Table 1). The CMCA algorithm presents a *symmetric organization*, meaning that, among other things, our algorithm considers all participants to be equal, which permits them to interact without the need of a mediator. The CMCA algorithm allows an *asynchronous diffusion* of messages, providing participants the freedom to interact at the moment they desire. Finally, it is compatible with a dynamic configuration since the group structure relies on a logical structure which can change with time. In order to support a dynamic configuration, our CMCA algorithm needs to add an underlying membership algorithm, such as [9].

As one can observe in Table 1, the only other algorithm which shares the same characteristics as our CMCA algorithm is the Isis CBCAST algorithm. The remaining ones have sacrificed some of these properties in order to reduce the amount of control information needed to be sent.

Let us first consider the Daisy Architecture algorithm [2]. The authors have sacrificed a symmetric organization by structuring participants into sub-groups and by using re-diffusion servers (one per subgroup), thus impeding a direct interaction among participants. One more recent work that follows the same philosophy as the previous one, and therefore presents the same disadvantages is the work by Taguchi et al. [20]. The main difference between them is that Taguchi treats differently the communication at a local level (intra-group) and at a server level. Next, let's take the algorithm called Causal Separators [19]. In this work, the causal zones S establish the groups of participants. The causal zones are established in an off-line mode according to the physical network topology. The main disadvantage of this work is that the configuration must be previously carried before launching the protocol, and it is not able to support a reconfiguration at runtime without stopping the system.

At last, we present the causal "Low cost approach" algorithm [11]. To our knowledge, this algorithm does use a minimal amount of control information; however, its disadvantage is that it works by synchronous phase execution, where every participant can send only one message per phase, and thus, considerably limits the interaction among participants and introduces several delays in the communication.

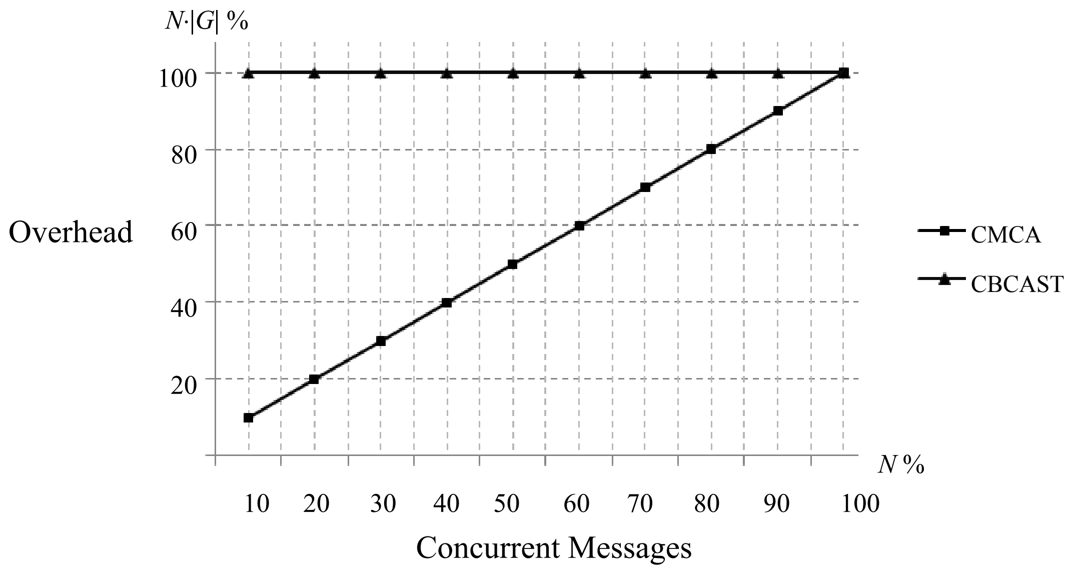

Figure. 4 Overhead sent in the worst case varying the number of concurrent messages

Now, let us discuss the amount of control information needed to be sent by each algorithm during the diffusion of messages in order to ensure a causal delivery. As previously noted, the "Low cost approach" algorithm is the algorithm which sends the least amount of control information. This algorithm timestamps only into each message a vector of size $|G|$, where $|G|$ is the number of groups in the system. As far as the Causal Separators algorithm is concerned, it timestamps per message in a causal zone a vector of size $|L|$, which is the number of participants in such causal zone. Since the message in question can cross through one or more causal zones in order to reach its destination, the final amount in the worst case of causal information becomes $\Sigma|L|$, $L \in S$, which is the sum of the different causal zones in the system. The case of the Daisy Architecture algorithm is similar to the previous one mentioned. The Daisy Architecture logically structures the sub-groups into channels of a size l. The maximum causal information sent for a hierarchy of level two is $(l\cdot(|S|+1))\cdot(|HS|+1)$, where $|S|$ is the number of servers (one per sub-group) and $|HS|$ is the number of hyper-servers (one per Daisy Architecture). Next, the two-layered algorithm proposed by Taguchi is a variant of the previous one. The overhead that it sends is $(l_i+k)l_i$ for local emissions and $(l_i+k)l_i + k^2$ for global messages (inter-group communication), where k is the number of subgroups in the system and l_i is the number of participant in a sub-group i.

Finally, the overhead generated for the CBCAST algorithm is in the worst case $\Sigma|g| : g \in G$, which is the sum of the participants of each channel of the system. This appears to be the same overhead generated by our CMCA algorithm, but as we will briefly present, the overhead is determined by the probability that different conditions will arise. In the CBCAST algorithm, the overhead attached per message for a participant is mainly determined by the positions that have been changed in its local *VT* (vector time) between local send events [5][6]. This means that the smaller the number of send events by a participant is, the greater the amount of causal overhead that can be sent per message.

In the CMCA algorithm, the amount of overhead timestamped per message relies only on the IICD related messages at a given message emission. To achieve this, all participants collaborate, updating the view of IICD related messages at each local *delivery* or *send* event. Each local *delivery* or *send* event has the property of eliminating old immediate dependencies and establish new ones. This property is important since a participant can remain as *listener* without sending messages, and its local causal history (potential overhead) will remain updated reflecting only IICD related messages ($CI(p)$ structure). To better illustrate this behavior, we present the following analysis. Consider a multi-group environment where all participants belong to all channels. If only a *send* event exists at a time (serial case), the overhead for the worst case for a *listener* in our CMCA algorithm is only $|G|$, while in the CBCAST the worst case is $N\cdot|G|$. If we now consider that at most two *send* events exist at a time in the system (i.e. at most two *concurrent send* events), the worst case for a *listener* in the CMCA is equal to $2\cdot|G|$; for three *send* events at most, the overhead can be $3\cdot|G|$ and so on, until one arrives at $N\cdot|G|$ (one *send* event per participant). We note that the CMCA algorithm dynamically adapts the worst case according to the number of *concurrent send* events that can exist in the system. However, for the CBCAST algorithm, the worst case remains $N\cdot|G|$ for all these cases (see Fig. 4).

Finally, we note that all the algorithms presented assume as hypothesis a reliable communication and a finite transmission delay. Nevertheless, there are presently some works that propose some mechanisms in order to eliminate such hypothesis in broadcast environments. For details, see [4] and [10].

VIII. Conclusions

We have introduced in this paper an immediate inter-channel dependency relation, which is an extension to the Immediate Dependency Relation. We have shown that the IICD characterizes the necessary information to ensure the causal delivery of messages in overlapping multi-channel environments. We have presented an efficient causal multi-channel algorithm, based on the IICD relation and benefiting of its “minimality” property, which is characterized by an asynchronous execution and a symmetric organization, allowing a free and direct interaction among the participants. We compare our CMCA algorithm with the CBCAST of Isis which is the only one (as far as we know) that shares the same properties. We show that our algorithm sends less causal overhead timestamped per message than the CBCAST.

IX. References

[1] A. Maddi, and F. Dahamni, *An efficient algorithm for causal messages ordering*, Proceedings of the 2001 ACM symposium on Applied computing, (2001) ISBN:1-58113-287-5, pp. 499 – 503.
[2] R. Baldoni, R. Friedman, and R. Van Renesse, *The hierarchical daisy architecture for causal delivery*, Distributed. System Engineering 6, 1999, pp. 71–81.
[3] Baldoni R, Prakash R, Raynal M, Singhal M: *Efficient causally ordered communications for multimedia real-time applications*. In Proc of the 4th International Symposium on High Performance

Distributed computing, Whasington, D.C., Aug. 1995, pp 140-147.

[4] Baldoni R., Prakash R., Raynal M., and Singhal M., 1998, *Efficient Δ-causal Broadcasting*, International Journal of Computer Systems Science and Engineering (IJCSSE), Volume 13, Number 5, pp. 263-269, 1998.

[5] K. Birman, A. Schiper, and P. Stephenson, *Lightweight Causal and Atomic Group Multicast*, ACM Trans. Compt. Syst. Vol. 9, Number 3, Aug. 1991, pp. 272-314.

[6] Birman, Kenneth, *Reliable Distributed Systems Technologies, Web Services, and Applications,* Editorial Sringer, 2005.

[7] C. A. Fidge, *Timestamps in message-passing systems that preserve partial ordering*, Australian Computer Science Communications, Vol. 10, Number 1, 1988, pp. 56-66.

[8] L. Lamport, *Time, clocks and the ordering of messages in distributed systems*, Communications ACM Vol. 21, Issue 7, 1978, pp. 558-565.

[9] Kal Lin and Vassos Hadzilacos, *Asynchronous group membership with oracles*, Proceedings of the 13th International Symposium on Distributed Computing (DISC), ISBN:3-540-66531-5, 1999, pp.79-83.

[10] Lopez, E., J. Estudillo, J. Fanchon, S. Pomares, *A Fault-tolerant Causal Broadcast Algorithm to be Applied to Unreliable Networks*, Proc. 17th International Conference on Parallel and Distributed Computing and Systems, 2005, pp. 465-470.

[11] Mostefaoui, and M. Raynal, *Causal multicast in overlapping groups: towards a low cost approach*, IEEE Workshop on Future Trends of Distributed Computer Systems, Sept.1993, pp 136-142.

[12] L. Peterson, N. Buchholz, and R. Schlichting, *Preserving and using context information in interprocess Communication*, ACM Transaction on Computer Systems, Vol. 7, 1989, pp. 217-246.

[13] S. Pomares Hernandez, J. Fanchon, and K. Drira, *The immediate dependency relation: an optimal way to ensure causal group communication*, Annual Review of Scalable Computing, Series on Scalable Computing, Vol. 6, 2004, pp 61-79.

[14] R. Prakash, M. Raynal, and M. Singhal, *An adaptive causal ordering algorithm suited to mobile computing environments*, Journal of Parallel and Distributed Computing, 1997, pp. 190-204.

[15] Ravindran K, Prasad B: *Communication Structures and paradigms for distributed conferencing applications*. 12th IEEE Int. Conf. On Distributed Computing Systems, May 1992

[16] Prakash R, Raynal M, Singhal M: *An effient causal ordering algorithm for mobile computing environments*. Technical report, Parallel Architectures Dept., Inst. Nat. de Rech. En Inf. et en Aut., France, 1995.

[17] Schneider F: *Implementing fault-tolerant services unsing the state machine spproach: A tutorial*, ACM Compt. Surveys, vol 22, No. 4, Dec. 1990, pp 299-319

[18] Chengzheng Sun , Xiaohua Jia , Yanchun Zhang , Yun Yang , David Chen, Achieving convergence, causality preservation, and intention preservation in real-time cooperative editing systems, ACM Transactions on Computer-Human Interaction (TOCHI), v.5 n.1, March 1998, pp.63-108,.

[19] L. Rodrigues, P. Verissimo, *Causal separators and topological timestamping: an approach to support causal multicast in large-scale systems*, Proc. 15th Int. Conference on Distributed Computing Systems, Vancouver, British Columbia, Canada, May 1995.

[20] K. Taguchi, M. Takizawa, *Two-layered protocol for a large-scale group of processes*, Proceedings of Ninth International Conference on Parallel and Distributed Systems, 17-20 Dec. 2002, pp. 171- 176.

[21] Torres-Rojas F J, Ahamad M: *Plausible Clocks: Constant Size Logical Clocks for Distributed Systems*. Distributed Computing, 12:179-196 (1999).