An Efficient Causal Multi-group Algorithm Based on Immediate Dependency Constraints

SAUL E. POMARES HERNANDEZ (✉)

LUIS MORALES ROSALES

National Institute of Astrophysics, Optics and Electronics (INAOE), Luis Enrique Erro #1, 72840 Tonantzintla, Puebla, Mexico.

spomares@inaoep.mx
Phone: + (52) 222 2-66-31-00 ext 8227
Fax: + (52) 222 2-66-31-52

JEAN FANCHON(✉)

Laboratory for Analysis and Architecture of Systems of CNRS, 7 Av. Colonel Roche, 31077 Toulouse Cedex, France.

fanchon@laas.fr
Phone : + (33) 5 61 33 64 75
Fax : + (33) 05 61 33 69 36

Abstract

In this paper we present an efficient causal multi-channel algorithm that can be used in a multi-group communication environment, including in the overlapping group case. The algorithm is developed upon an extension we propose to the Immediate Dependency Relation (IDR), which was introduced by Peterson. This IDR extension can be used in multi-group environments in order to identify causal dependencies between messages. In the same way that the IDR characterizes the sufficient control information to ensure causal delivery in a broadcast case, we show that our IDR extension allows us to define the sufficient control information to ensure the causal delivery in a multi-group environment. We show that through the use of the IDR extension we can minimize the amount of control information sent per message without imposing restrictions in interaction or execution (e.g. network topology, rediffusion 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.

Key words: *Immediate Dependency Relation, Causality, Group Communication, Distributed Computing.*

IDR, Immediate Dependency Relation;
FIFO, First In First Out;
CMCA, Causal Multi-Channel Algorithm;
IICD, Immediate Inter-Channel Dependency;
PC, Propagation Constraint;
CI, Causal Information.

1 Introduction

In Group 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 group 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 [20], teleconferencing [25], multimedia systems [2], mobile computing [23], resource allocating algorithms [29] and security domains [27].

Among the various existing causal algorithms, we can distinguish causal broadcasts, where each message is sent to all participants; causal multicasts, where a message is sent to any subset of participants; and multi-groups causal algorithms (which we call indifferently multi-channels algorithms), where participants are organized in overlapping groups. In this last case, a message is broadcasted to all members of a particular group. With regard to algorithms which support multi-group environments, we can distinguish symmetric from 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 present restrictions in the mode of interaction or introduce a large amount of control information.

The present paper proposes a symmetric Causal Multi-Channel Algorithm (CMCA). The main objective of our algorithm is to minimize the amount of

control information (CI) per message. The amount of information necessary to guarantee causal broadcast in a group g with n members is $\theta(n)$ in the worst case [5]. In overlapping groups, this amount is $\theta(g \bullet n)$ in the worst case [5], where g is the number of groups in the system. For large values of n and g, the bounds of $\theta(n)$ and $\theta(g \bullet n)$ are prohibitively high. An original aspect of our work is the definition and the use of an extension of the *Immediate Dependency Relation* (IDR) [18] [22] to the multi-group context. We call this extension the *Immediate Inter-Channel Dependency* (IICD). The IICD relation reduces the amount of CI attached to the messages without imposing restrictions in interaction or execution (e.g. network topology, rediffusion servers, executions models, etc). In the same way that the IDR characterizes the sufficient control information in a broadcast case [22], the IICD allows us to define the sufficient control information to ensure the causal delivery of messages in a multi-channel situation: if any two messages related by IICD are delivered in causal order, then all the messages are delivered in causal order (IICD property). Due to IICD property, our CMCA algorithm can be viewed as a natural extension to the multi-channel case of the minimal causal broadcast algorithm proposed in [21] and [22] for the single-channel case.

The paper proceeds as follows: first, in Section 2 we overview the works which tend to minimize control information in symmetric algorithms. In Section 3 we give the basic definitions, in particular the definition of causal dependency. Next in Section 4, we present the immediate dependency and our proposed extension to the multigroup case. In Section 5, the proposed Causal Multi-channel Algorithm is presented in detail. Section 6 is dedicated to the proofs, which include the proof the IICD property, and the proof of correctness of the CMCA algorithm. Finally, in Section 7 we present a comparison of our algorithm versus other important multi-group communication works.

2 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, a particular channel structure [3], and/or execution models [26]. The following related work involves only the symmetric category, which is the category we are concerned with.

One of the first algorithms is the work done by Peterson [18]. Peterson introduced the *context graph*, which was designed to represent the causality between messages in a conversation algorithm in a broadcast environment. 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 [18] 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 (see Section 4).

The work of Prakash and Raynal [24] extends the immediate dependency property to support multicast environments. 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' on which its delivery is directly dependent upon. This approach is oriented to resolve problems in mobile agent applications; it considers the multicast and broadcast case, but not the case of overlapping channels.

An interesting work is the algorithm proposed by Kshemkalyani and Singhal [13]. Their causal algorithm codes information onto each message regarding all previous causal messages that are not yet guaranteed to be delivered in a causal manner by the algorithm, or that are not known to have already been delivered. They refer to these conditions as *propagation constraints* (Definition 1). The propagation constraints (PC) specify the conditions on the information propagation to enforce causal ordering [13].

Definition 1 The information $d \in m_{i,a}.Dests$ concerning a message $m_{i,a}$ sent to d must propagate along all causal paths, starting at message (i, a) up to, and only up to, the earliest messages (j, b) on any such path, such that either:

PC 1. $delivery_d(m_{i,a}) \overrightarrow{\Rightarrow}$ (j, b), i.e. it is known that $m_{i,a}$ has been delivered or

PC 2. $\exists$ (k, c) | (i, a) $\rightarrow$ (k, c) $\rightarrow$ (j, b) $\wedge$ $d \in m_{k,c}.Dests$ i.e. it is guaranteed that it will be delivered in causal order.

The first PC means that the message $m_{i,a}$ is known to have been delivered in causal order at message (j, b) to destination $d = j$. Therefore, in any future multicast message to destination d, the message does not need to carry information concerning $d \in m_{i,a}.Dests$. The second PC means that there exists a message (k, c) that would ensure the causal delivery of the present message (j, b) with respect to (i, a). If (k, c) is causally delivered with respect to (i, a), the information about $d \in m_{i,a}.Dests$ does not need to be sent.

Another example of an algorithm that falls into the symmetric category is the algorithm proposed by Birman [5], which is based on vector time clocks. The author, in his algorithm, proposes to compress the vector time by sending only the vector positions that have changed between the diffusion of the message in question and the diffusion of the prior local message.

3 Preliminaries

Participants and channels:

The application under consideration is composed of a set of participants P organized into possibly overlapping groups which communicate by asynchronous message passing. Groups correspond to logical broadcast communication channels; the members of a group are connected to the corresponding channel and communicate by sending and receiving messages through it. We denote by C the set of channels, the mapping $Memb : C \rightarrow 2^P$ defines for each channel the set of connected participants, and the mapping $Conn : P \rightarrow 2^C$ defines for each participant the set of channels to which it is connected.

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 broadcasted, and $c \in C$ is the channel on which m is broadcasted, 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 [15]. The causal relation is a strict partial order (transitive and antisymmetric) denoted by $e \rightarrow e'$, i.e. e causally precedes e', 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 set of events E_p involving p is totally ordered:
$e,e' \in E_p \Rightarrow e \rightarrow e' \vee e' \rightarrow e$

2) For each message m and destination p of m, the emission of m precedes its delivery to p: $p \in Dest(m) \Rightarrow send(m) \rightarrow delivery(p,m)$

Causal order delivery in channel 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 [6]:

Definition 3 Causal Order Delivery (one channel):

If $send(m) \rightarrow send(m')$, then $\forall p \in Memb(c)$:

$delivery\ (p,m) \rightarrow delivery\ (p,m')$

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* that belongs to *c*.

The precedence relation on messages denoted by $m \rightarrow m'$ is induced by the precedence relation on events, and defined by:
$m \rightarrow m' \Leftrightarrow send(m) \rightarrow send(m')$

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. Let us note that if $c=Chan(m)$ and $c'=Chan(m')$, we have $Memb(c) \cap Memb(c') = Dest(m) \cap Dest(m')$. 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*, *c*) causally precedes the diffusion of a message (*m*', *c*'), where *c* and *c*' are 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 may receive both messages, i.e. that belong to both channels *c* and *c*' [17].

4 The Immediate Dependency Relations

4.1 Single channel case

The Immediate Dependency Relation (IDR) is the transitive reduction of the causal precedence. The IDR is included in the causal precedence and is the smallest relation which generates it by transitive closure. This relation is important for causal delivery algorithms and protocols since it is necessary, but also sufficient, that the delivery of messages related by IDR respects the order of

their emissions to ensure the causal delivery of all messages. We present below the definition of the IDR relation and its theorem regarding causal delivery.

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

Thus, a message *m* directly precedes a message *m'*, iff no other message *m''* belongs both to the causal future of *m*, and to the causal past of *m'*.

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

This property has been shown in [18] (for a formal proof see [21] and [22]).

4.2 Multi-channel case

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 we refer to it by its acronym IICD. In the same way that the IDR allows us to define the sufficient control information in a broadcast case, the IICD allows us to characterize the sufficient control information to ensure the causal delivery of messages in a multi-channel situation.

For convenience, henceforth, we will denote *send*(m,c) by (m,c).
Definition 6 Immediate Inter-channel Dependency Relation (IICD) $\uparrow$:
$(m,c) \uparrow (m',c') \Leftrightarrow [((m,c) \rightarrow (m', c')) \wedge \forall (m'', c'') \in M, ((m,c) \rightarrow (m'', c'') \rightarrow (m', c')$
$\Rightarrow c'' \neq c \wedge c'' \neq c')]$

By this definition, a message *m* broadcasted on channel *c* has an immediate dependency relation with a message *m'* broadcasted 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'*.

We note that in the case of a single channel, the relation '$\uparrow$' coincides with the IDR relation denoted by '$\downarrow$', which has been previously defined.

The following proposition establishes that if any two messages related by IICD are delivered in causal order, then all the messages are delivered in causal order.

Theorem 2 (Proof in Section 6)
If $\forall\ m,m' \in M, (m,c) \uparrow (m',c') \Rightarrow \forall p \in Memb(c) \cap Memb(c')$ *:*
$delivery(p,m) \rightarrow delivery(p,m')$
then $\forall\ m,m' \in M, (m,c) \rightarrow (m',c') \Rightarrow \forall p \in Memb(c) \cap Memb(c')$*:*
$delivery(p,m) \rightarrow delivery(p,m')$

As we show in Section 6, the information concerning the IICD relation between messages is therefore sufficient to ensure their causal delivery.

4.3 Illustration of Immediate Inter-channel Dependency

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

Scenario: The Multi-channel space diagram in Fig. 1 is composed of c_1={ p_1, p_a, p_b, p_2}, c_2={ p_2, p_3} and c_3={ p_1, p_3} where p_a, p_b are local processes 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**.**

Let's 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. 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 ($p_2 \in Memb(c_1) \cap Memb(c_2)$) 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).

Figure 1

5 The Causal Multi-Channel Algorithm (CMCA)

An informal description of the algorithm and data structures is the following: our algorithm uses the IICD relation presented in Definition 6 and Mattern's vector clocks. The information of a process related to a past message is a list of channels, through which the causal information representing this message is not known by the process to have been transmitted. The CMCA algorithm minimizes the overhead by avoiding the transmission of redundant information. In the next sections we present in detail the proposed algorithm. We first present a description of the local identifiers used, then the data structures, followed by the algorithm specification; lastly, a precise roadmap of the code is presented.

5.1 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 Fig. 2, 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 interval 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 $id(p,c) \in I$. The mapping id is one to one, i.e. for any pairs (p,c), $(p',c') \in P \times C$, we have $id(p,c)=id(p',c') \Rightarrow (p,c)=(p',c')$. We often denote by p_i the unique participant p, such that $i=id(p,c)$ for some channel c, and we have $i=id(p,c) \Rightarrow p=p_i$. For any $i \in I$, we denote by $ch(i)$ an 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 Fig. 2, participant p_1 has two identifiers, one for each channel it belongs to (c_1 and c_3).

Figure 2

5.2 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 6.2 : $(k,x,c) \in CI(p) \Rightarrow x = VT(p)[k]$). 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* broadcasts 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 (local) clock value 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.

5.3 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. $ci.ch\_dests \leftarrow ci.ch\_dests \setminus c$
9. **endif**
10. **if** $ci.dests = \varnothing$ **then**
11. $CI(p) \leftarrow CI(p) \setminus ci$
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. $CI(p) \leftarrow CI(p) \bigcup \{(i, t, c, Conn(p) \setminus c)\}$

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

To impose a causal delivery

18. **If not** $(t = VT(p)[i] + 1 \wedge$
19. $\forall (l, x, d) \in H(m) : d \in Conn(p) \Rightarrow x \le VT(p)[l]\ )$ **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. $CI(p) \leftarrow CI(p) \bigcup \{(i, t, c, Conn(p)\}$
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 /*** skip */
32. **endif**
33. **if** $x = y$ **then**
34. $UPD(\ ci_{l,y,d}\ ,\ c\ )$
35. **endif**
36. **endif**
37. **else** /* $d \notin Conn(p)$ */
38. **if** $(\exists y\ (l, y, d) \in CI(p))$ **then**
39. **if** $x < y$ **then /***skip */
40. **endif**
41. **if** $x = y$ **then**
42. $UPD(\ ci_{l,y,d}\ ,\ c\ )$
43. **endif**
44. **if** $x > y$ **then**
45. $VT(p)[l] = x$
46. $CI(p) \leftarrow CI(p)\ \setminus\ \{(l, y, d)\}$
47. $CI(p) \leftarrow CI(p) \bigcup \{(l, x, d, Conn(p))\}$
48. **endif**
49. **else /*** $\neg \exists y : (l, y, d) \in CI(p)$ ***/**
50. **if** ($VT(p)[l] < x$) **then**
51. $VT(p)[l] = x$
52. $CI(p) \leftarrow CI(p) \bigcup \{(l, x, d, Conn(p))\}$
53. **endif**
54. **endif**
55. **endif**
56. **endfor**
57. **endif**

IV. Updating

58. $UPD(ci_{k,x,d}, c)$
59. **if** $(c \neq d)$ **then**
60. $ci_{k,x,d}.ch\_dests \leftarrow ci_{k,x,d}.ch\_dests \setminus c$
61. **if** $\left(ci_{k,x,d}.ch\_dests = \varnothing\right)$ **then**
62. $CI(p) \leftarrow CI(p) \setminus \{(k,x,d)\}$
63. **endif**
64. **else** /* $c = d$ */
65. $CI(p) \leftarrow CI(p) \setminus \{(k,x,d)\}$
66. **endif**

5.4 Algorithm Description

We provide in this section a description of the CMCA algorithm presented in Section 5.3. We do the presentation in a sequential order with respect to the numeration of the lines. Our explanation is focused on the description of the updating process of structures H_m and CI.

Table 1

6 Proofs

We first show in Section 6.1 that the causal delivery of messages in immediate inter-channel dependency is sufficient to ensure the causal delivery of all messages. Next, we present the proof of correctness of the CMCA algorithm. This proof has two aspects: we first show in Section 6.2 that no delivery order is imposed to messages which are not causally dependent on each other. Secondly, we demonstrate in Section 6.3 that the algorithm indeed delivers in proper order any pair of messages causally dependent on each other.

6.1 Proof of the IICD property

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 $send(m_i)\downarrow send(m_{i+1})$ for all $i =0...n-1$.

Theorem 2

If $\forall\ m,m' \in M, (m,c) \uparrow (m',c') \Rightarrow \forall p \in Dest(m) \cap Dest(m')$: $delivery(p,m) \rightarrow delivery(p,m')$

then $\forall\ m,m' \in M, (m,c) \rightarrow (m',c') \Rightarrow \forall p \in Dest(m) \cap Dest(m')$: $delivery(p,m) \rightarrow delivery(p,m')$

Proof:

Let m and m' be such that $send(m) \rightarrow send(m')$ we show that $\forall p \in Dest(m) \cap Dest(m') : delivery(p,m) \rightarrow delivery(p,m')$. We know this property is satisfied if $send(m) \uparrow send(m')$. Otherwise, let $d(m,m') = n$, $c=Chan(m)$ and $c'=Chan(m')$. 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/or $send(m'')\uparrow send(m')$, 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 there is an integer $0 \leq k \leq n-1$ such that $Chan(m_i)=c$ for all $i \leq k$ and $Chan(m_i)=c'$ for all $i>k$. Let a common recipient $p \in Dest(m) \cap Dest(m')$, this means that $p \in Memb(c) \cap Memb(c')$, and thus, that p receives all the messages of the sequence ($m_i=(k_i, x_i, c_i)$, $i=0...h$). Due to 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 $delivery(p,m) \rightarrow delivery(p,m')$. □

6.2 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. This is shown in Theorem 3: if a message m appears in the structure $H(m')$ of a message m', then m

causally precedes *m'*. This means that the constraints applied to the delivery of a message only depend on its causal past.

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 process *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 47 or 52 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 $send(m_i) \rightarrow send(m_{i-1}) \ldots$. $send(m_1) \rightarrow send(m')$ 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 $send(m) \rightarrow send(m_i) \rightarrow send(m_{i-1}) \ldots send(m_1) \rightarrow send(m')$ concludes the proof. □

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

Lemma 1 If a state *s* of a process *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 $(l,x,c) \in H(m')$ (instruction lines 45 and 51). 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 $\downarrow s$). □

6.3 Proof of correctness (2)

The second proof of correctness shows that whenever two messages m and m' are causally dependent, i.e. $send(m) \rightarrow send(m')$, 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 $p \in Dest(m) \cap Dest(m')$. We show that if $send(m,c) \uparrow send(m',c')$, when the reception of m' by some destination $p \in Dest(m) \cap Dest(m')$ occurs, either the message m has already been delivered to p, or $H(m')$ contains the entry (l,x,c) corresponding to m or an entry (l,y,c) with $x < y$ (for a message emitted after m by the same source on the same channel). This ensures that m has to be delivered to p before m'.

To make the 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 $\downarrow s = \{{}^{\circ}s\} \cup \downarrow({}^{\circ}s)$.

The following four Lemmas are used in the proof of Theorem 4. Lemma 2 ensures that the structure $CI(p)$ is a 'partial copy' of $VT(p)$, i.e. 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 instruction line 17 on sending the message (l,x,c), and instruction line 3 ensures that $x = VT(p)[l]$, either on delivery of the message (l,x,c) at line 27, and when this instruction is executed, test line 18 and instruction 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 instruction line 47 (resp. at line 52), and the instruction previously executed at line 45, (resp. at line 51) 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 line 37 is executed. If condition line 38 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 45 is executed yielding $VT(p)[l] = x$. Otherwise, condition line 38 is false, execution starts line 49 and the instruction lines 50 and 51 ensure that $VT(p)[l] \geq x$. □

Lemma 4 If at state s of a process 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 $c' \in (l,x,c).Dest$.

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 $(l,x,c).Dest=Conn(p)$ at instruction line 27. The channel $c' \in Conn(p)$ is suppressed from $(l,x,c).Dest$ only in two cases:

- when a message m' with $(l,x,c) \in H(m')$ is emitted by p to the channel c' (instruction 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 34); 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 instruction line 45 (resp. line 51); and in that case, the entry (l,x,c) is added to $CI(p)$ with $(l,x,c).Dest=Conn(p)$ by instruction line 47 (resp. line 52). The channel $c' \in Conn(p)$ is suppressed from $(l,x,c).Dest$ only in two cases :

- either when a message m' with $(l,x,c) \in H(m')$ is emitted by p to the channel c' (instruction 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 42); 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. It shows that when a process p such that $VT(p)[l] = x \neq 0$ with $c = ch(l)$, broadcasts a message m to a channel c' without the information (l,x,c), it implies that this information (the tuple (l,x,c)) had been previously broadcasted to the same channel c'.

Lemma 5 If a process 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) $(l,x,c) \in H(m'')$

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 instruction lines 6 and 7, no entry $(l,x,c) \in CI(p)$ exists such that $c' \in (l,x,c).Dest$. Then, by the previous Lemma 4, a message m' exists such that $(l,x,c) \in H(m')$ and $send(m',c') \rightarrow send(m'',c')$. Due to Theorem 3, we have $send(m,c) \rightarrow send(m',c')$. □

Theorem 4

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

Proof:

We prove by induction on the distance $d(m,m')$ between m and m'. The distance $d(m,m')$ is defined for any pair of messages m and m' such that $send(m) \rightarrow send(m')$, and 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 $send(m_i) \downarrow send(m_{i+1})$ for all $i =0...n-1$.

If $d(m,m')=1$, then $send(p,m) \downarrow send(p',m')$. If $p \neq p'$, we have $delivery(p',m) \rightarrow send(p',m')$. Let $m=(k,x,c)$, we have $(k, x, c) \in CI(p')$ after $delivery(p',m)$ (instruction line 27), and thus $(k, x, c) \in H(m')$ (instruction line 7). For any $q \in Dest(m) \bigcap Dest(m')$ the delivery of m to q precedes the delivery of m', and we have $delivery(q,m) \rightarrow delivery(q,m')$.

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 $d(m,m') = n$, there is a message sequence $(m_i=(k_i, x_i, c_i)$, $i=0...h)$, such that $m=m_0$, $m'=m_h$ and for all $i=0...h-1$, $send(m_i) \downarrow send(m_{i+1})$. By definition of $d(m,m')$ we have $h \leq d(m,m')$. Furthermore, due to hypothesis $send(m) \uparrow send(m')$, we also have $c_0 \neq c_i \neq c_n$ for all $i=1...h-1$.

For all i=0...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...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 the broadcast of m_i to c_i occurs, by Lemma 5 only two cases may occur:

- $(k_0, y, c_0) \in H(m_i)$. 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) broadcasted on c_i, such that $(k_0, y, c_0) \in H(m)$ and $send(k_0, y, c_0) \rightarrow send(m) \rightarrow send(m_i)$. Due to $x_0 \leq y$, we have $send(k_0, x_0, c_0) \rightarrow send(m)$, 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 $delivery(p_{i+1},m) \rightarrow delivery(p_{i+1},m_i)$. 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 broadcast 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 $send(m,c) \rightarrow send(m'',c') \rightarrow send(m',c')$, 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 $delivery(q,m) \rightarrow delivery(q,m')$. □

The correction of the algorithm results from Theorems 3 and 4:

Corollary 1

For any messages *m* and *m'*, we have

$send(m) \rightarrow send(m') \Rightarrow \forall p \in Dest(m) \bigcap Dest(m') : delivery(p,m) \rightarrow delivery(p,m')$

7 The Causal Multi-Channel Algorithm vs other algorithms

In this section we compare the characteristics of our CMCA algorithm with other main existing algorithms (Table 3). 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's compatible with a *dynamic configuration*. Because of its dynamic configuration, one does not need to know in advance the exact composition of the channel participants. Furthermore, the dynamic configuration permits the channel composition to change during the execution time.

As one can observe in Table 3, 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 [3]. The authors have sacrificed a symmetric organization by structuring participants into sub-channels and by using re-diffusion servers, thus impeding a direct interaction among participants. Next, let's take the example of the algorithm presented by the work of Causal Separators [26]. The main disadvantage of this algorithm is its static configuration, meaning that it must be carried out off-line. Finally, let's consider the causal "Low cost approach" algorithm [17]. To our knowledge, this algorithm does use a minimal amount of control information; however, its disadvantage is that it works by synchronous phase execution, and thus, considerably limits the interaction among participants and introduces delays in the transmission.

Table 2

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 only timestamps onto each message a vector of size $|G|$, where $|G|$ is the number of channels in the system. As far as the Causal Separators algorithm is concerned, it timestamps a vector of size $|L_s|$ per message, where $|L_s|$ represents the number of participants in a causal zone. Since the message in question must cross through one or more causal zones in order to reach its destination, the final amount of causal information becomes $\Sigma|L_s|$, which is the sum of the different causal zones crossed. The case of the Daisy Architecture algorithm is similar to the previous one mentioned, in the sense that the sub-channels are of a fixed size denoted by l, and therefore, the amount of control information sent is $l \cdot (s)$, where s refers to the number of sub-channels that the message in question must cross in order to reach its final destination.

With respect to the overhead generated by the CBCAST of Isis algorithm, we find that in the worst case $\Sigma|g| : g \in G$, which is the sum of the elements of each group of the system. This appears to be the same overhead generated by our CMCA algorithm, but by reconsidering the study completed in [22], we briefly mention that the overhead is determined by the probability that different conditions will arise. In the CBCAST algorithm, the overhead is determined by the number of messages received either in parallel or in sequential order between local emissions. In our algorithm, the amount of overhead is only determined by the number of parallel messages received. As presented in [22], if the behavior is a serial reception, the amount of overhead generated in the single-channel case is equal to $|CI|=1$, while in the multigroup case it is equal to $|CI|=|G|$.

8 Conclusions

In this article we mentioned briefly that an optimal size of control information (overhead) and a symmetric organization are some of the characteristics of causal algorithms suitable to support large decentralized distributed systems. We have shown in this paper that our IDR extension for the multi-channel case minimizes the overhead timestamped per message without introducing restrictions in interaction or execution. Finally, we have presented an efficient causal algorithm for the generic multi-channel case, including the overlapping case, which is based on our IDR extension. We showed the efficiency of our causal algorithm in terms of the overhead timestamped per message.

References

1. Awerbuch B: Complexity of Network Synchronization. Journal of the ACM, Vol. 32, No 4, 1985, pp 801-823

2. 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

3. Baldoni R, Friedman R, Van Renesse R: The Hierarchical Daisy Architecture for Causal Delivery. 17th IEEE Int'l Conference on Distributed Computing Systems, May 1997

4. Birman K, Joseph T: Reliable Communication in Presence of Failures. ACM Trans. Compt. Syst., 5(1), Feb. 1987, pp17-76

5. Birman K, Schiper A, Stephenson P:Lightweight Causal and Atomic Group Multicast. ACM Trans. Compt. Syst. Vol. 9, No. 3, Aug. 1991, pp 272-314

6. Birman K: The Process Group Approach to Reliable Distributed Computing. Communications of the ACM, Vol. 36, No. 12, 1993, pp 36-53

7. Charron-Bost B: Concerning the Size of Logical Clocks in Distributed Systems. Inf Process Lett 39, 1991, pp 11-16

8. Fayolle G, Flajolet P, Jacquet P: Analysis of a Stack Algorithm for Random Multiple Access Communication: Special Issue on Random-Access Communication, IEEE Transactions on Information Theory IT-31, 1985, pp 244-254

9. Fidge C A: Timestamps in Message-Passing Systems that Preserve Partial Ordering. Australian Computer Science Communications, Vol 10, No. 1, 1988, pp 56-66

10. Dictionnaire HACHETTE Encyclopédique Illustré, ISBN 2-01-28-0476-4, 1996

11. Jacquet P : Contribution de l'Analyse d'Algorithmes a l'Évaluation d' Algorithmes de Communication. PhD thesis of l'Université de Paris-Sud Centre d'Orsay, Nov. 1989

12. Helary J, Melideo G, Raynal M: Tracking Causality in Distributed Systems: a Suite of Efficient Algorithms. In Proc. 7th Int'l Colloquium on Structural Information and Communication on Complexity, Carleton Scientific, June 2000

13. Kshemkalyani A D, Singhal M: An Optimal Algorithm for Generalized Causal Message Ordering. Proc. 15th Annual ACM Symposium on Principles of Distributed Computing, ACM, May 1996, pp 87-87
14. Kshemkalyani A D, Singhal M: Necessary and Sufficient Conditions on Information for Causal Message Ordering and their Optimal Implementation. Distributed Computing 11:91-111 (1998)
15. Lamport L: Time, Clocks and the Ordering of Messages in Distributed Systems. Communications ACM 21(7), 1978, pp 558-565
16. Mattern F: Virtual Time and Global States of Distributed Systems. Parallel and Distributed Algorithms, North-Holland, 1989, pp 215-226
17. Mostefaoui A, Raynal M: Causal Multicast in Overlapping Groups: Towards a Low Cost Approach. IEEE Workshop on Future Trends of Distributed Computer Systems, Sept.1993, pp 136-142
18. Peterson L, Buchholz N, Schlichting R: Preserving and Using Context Information in Interprocess Communication. ACM Transaction on Computer Systems, 7:217-246 (1989)
19. Pomares Hernandez S, Fanchon J, Drira K, Diaz M: Causal broadcast protocol for very large group communication systems. Journal Studia Informatica Universalis, Editions Suger, Special Issue vol. 2: 175-188 (2001)
20. Pomares Hernandez S, Fanchon J, Drira K, Diaz M: An Efficient Multi-Group Distributed Coordination Algorithm for Collaboration Engineering Activities. IEEE Int. Conf. on Systems, Man and Cybernetics, Hammamet, Tunisia, October, 2002.
21. Pomares Hernandez S, "Services De Coordination Et Algorithmes De Diffusion Causale Pour Les Applications Coopératives Distribuées", PhD thesis of National Institute Poltechnique of Toulouse, France, November, 2002
22. Pomares Hernandez S, Fanchon J, Drira K: The immediate dependency relation: an optimal way to ensure causal group communication. Annual Review of Scalable Computing, Editions World Scientific, Séries on Scalable Computing, Vol.6: pp 61-79 (2004)
23. 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
24. Prakash R, Raynal M, Singhal M: An Adaptive Causal Ordering Algorithm Suited to Mobile Computing Environments. Journal of Parallel and Distributed Computing, pp 190-204 (1997)
25. Ravindran K, Prasad B: Communication Structures and paradigms for distributed conferencing applications. 12th IEEE Int. Conf. On Distributed Computing Systems, May 1992
26. Rodrigues L, Verissimo P: Causal Separators and Topological Timestamping: an Approach to Support Causal Multicast in Large-Scale Systems. Proc. 15th Int'l Conference on Distributed Computing Systems, Vancouver, British Columbia, Canada, May 1995
27. 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
28. Schwarz R, Mattern F: Detecting Causal Relationships in Distributed Computations: in Search of the Holy Grail. Distributed Computing 7:149-174 (1994)
29. Torres-Rojas F J, Ahamad M: Plausible Clocks: Constant Size Logical Clocks for Distributed Systems. Distributed Computing, 12:179-196 (1999)

Figure 1 Multi-channel scenario

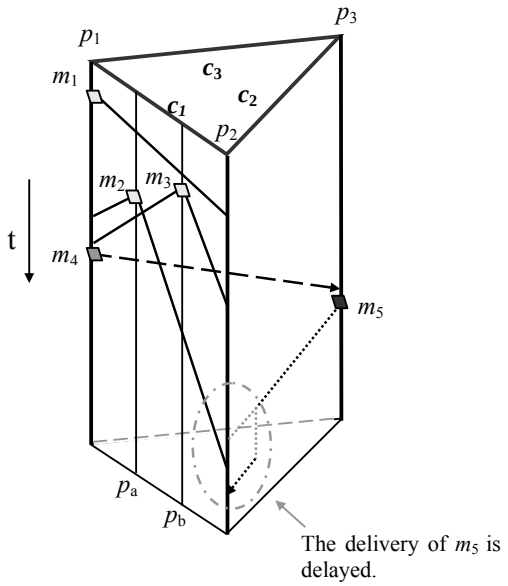

Figure 2 Connection scheme

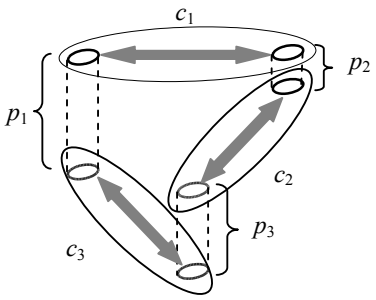

Table 1 Algorithm Description

CODE	Description
I. Initially	At the beginning, each process p locally initializes its structures $VT(p)$ and $CI(p)$ in the following manner:
1. $VT(p)[k] = 0 \; \forall \; k{:}1 \ldots \Sigma_{g \in G} \lvert g \rvert$	Each position in the vector $VT(p)$ is filled by zero.
2. $CI(p) \leftarrow \varnothing$	Structure $CI(p)$ is emptied of all elements.
II. For each *message* diffused by p into channel c with $i=id(p,c)$	Sending procedure
3. $VT(p)[i] = VT(p)[\, i] + 1$	Position i of vector $VT(p)$ is increased by one.
4. $H(m) \leftarrow \varnothing$	$H(m)$ is emptied of all elements when each message is sent.

5. **for all** $ci \in CI(p)$ **:** ci=(k, x, d , ch_dests)	In lines 5-13 we construct the structure $H(m)$ from the information in $CI(p)$ and update the $CI(p)$.
6. **if** $c \in ci.ch\_dests$ **then** 7. $H(m) \leftarrow H(m) \cup \{(k,x,d)\}$ 8. $ci.ch\_dests \leftarrow ci.ch\_dests \setminus c$ 9. **endif**	If channel c belongs to the field $ci.ch\_dests$ a new entry (k,x,d) is added to $H(m)$ and then the channel c is deleted from $ci.ch\_dests$.
10. **if** $ci.ch\_dests = \varnothing$ **then** 11. $CI(p) \leftarrow CI(p) \setminus ci$ 12. **endif**	If $ci.ch\_dests$ is empty, then the entry ci is erased from $CI(p)$.
13. **endfor**	
14. t=$VT(p)$[i]	The content of $VT(p)$**[** i] is assigned to t.
15. m=(i , t, d, $message$, $H(m)$)	Construction of message m
16. **send**(m) into **the channel** c	Broadcasting of message m on channel c
17. $CI(p) \leftarrow CI(p) \cup \{(i,t,c,Conn(p) \setminus c)\}$	Insertion of the element ci=(i, t, c,$Conn(p)$ \ c) to $CI(p)$
III. For each m **=** $(i,t,c,message,H(m))$ **received by** p **with** j=$id(p,c)$	Reception procedure
To impose a causal delivery Condition of Multi-channel delivery 18. **If not** $(t = VT(p)[i] + 1 \wedge$	The delivery condition is divided into two parts. The first part verifies that the receptions for a given process satisfy the FIFO order delivery.
19. $\forall (l,x,d) \in H(m) : d \in Conn(p) \Rightarrow x \leq VT(p)[l]$)	The second part verifies that any element (l,x,d) of $H(m)$ such that p is connected to d has been previously delivered to p.
then 20. **wait**	If the delivery condition is not satisfied, then the message is placed on hold until the missing message arrives.
21. **else** 22. ***Delivery*****(***message***)**	If the delivery condition is satisfied, then the message is delivered to the application and we proceed to update the vector $VT(p)$ and $CI(p)$ as follows:
23. $VT(p)[i] = VT(p)[i] + 1$	The position i of vector $VT(p)$ is increased by one.
24. **if** $(\exists x\,(i,x,c) \in CI(p)$ **then** 25. $CI(p) \leftarrow CI(p) \setminus \{(i,x,c)\}$ 26. **endif** 27. $CI(p) \leftarrow CI(p) \cup \{(i,t,c,Conn(p))\}$	If there exists an element identified by (I,x,c) in $CI(p)$ with the same sender p_i of the present message m, then this element is replaced by an element with the identifier (i,t,c) of m.
28. **for all** $(l,x,d) \in H(m)$	Updating process of $CI(p)$ with regard to the information to contained in structure $H(m)$

	(lines 28-56).
29. **if** ($d \in Conn(p)$) **then**	The updating process of $CI(p)$ is divided into two parts: the first part (lines 29-36) concerns each entry (l,x,d) in $H(m)$ such that p is connected to d. The second part (lines 37-57) concerns the remaining entries (l,x,d) in $H(m)$, such that $d \notin Conn(p)$.
30. **if** $(\exists y\,(l,y,d) \in CI(p))$ **then**/*$x \le y$*/ 31. **if** $x < y$ **then** /* skip */ 32. **endif** 33. **if** $x = y$ **then** 34. $UPD(ci_{l,y,d}, c)$ 35. **endif** 36. **endif**	When $d \in Conn(p)$, if there exists an entry (l,y,d) in $CI(p)$ of the same sender p_l then, we have two possibilities: $x < y$ or $x = y$. When $x = y$ we proceed to update the $CI(p)$ with the function $UPD()$ in lines 58-66.
37. **else** /* $d \notin Conn(p)$ */ 38. **if** $(\exists y\,(l,y,d) \in CI(p))$ **then** 39. **if** $x < y$ **then** /* skip*/ 40. **endif** 41. **if** $x = y$ **then** 42. $UPD(ci_{l,y,d}, c)$ 43. **endif** 44. **if** $x > y$ **then** 45. $VT(p)[l] = x$ 46. $CI(p) \leftarrow CI(p) \setminus \{(l,y,d)\}$ 47. $CI(p) \leftarrow CI(p) \cup \{(l,x,d,Conn(p))\}$ 48. **endif**	When $d \notin Conn(p)$, if there exists an entry (l,y,d) in $CI(p)$ of the same sender p_l then, we have three possibilities: $x < y$, $x = y$, or $x > y$. When $x = y$ we proceed to update the $CI(p)$ with the function $UPD()$ in lines 58-66. When $x > y$ the position l of the vector $VT(p)$ is updated with the value of x and the previous entry (l,y,d) is replaced with a most recent element (l,x,d).
49. **else** /* $\neg\exists y\,(l,y,d) \in CI(p)$ */ 50. **if** $(VT(p)[l] < x)$ **then** 51. $VT(p)[l] = x$ 52. $CI(p) \leftarrow CI(p) \cup \{(l,x,d,Conn(p))\}$ 53. **endif** 54. **endif** 55. **endif** 56. **endfor** 57. **endif**	When $d \notin Conn(p)$, and there doesn't exists an entry (l,y,d) in $CI(p)$, the position l of the vector $VT(p)$ is updated with the value of x, and an entry identified by (l,x,d) is added to $CI(p)$.
IV. Update function 58. $UPD(ci_{k,x,d}, c)$ 59. **if** $(c \neq d)$ **then** 60. $ci_{k,x,d}.ch\_dests \leftarrow ci_{k,x,d}.ch\_dests \setminus c$ 61. **if** $(ci_{k,x,d}.ch\_dests = \varnothing)$ **then** 62. $CI(p) \leftarrow CI(p) \setminus \{(k,x,d)\}$ 63. **endif** 64. **else** /* $c = d$ */ 65. $CI(p) \leftarrow CI(p) \setminus \{(k,x,d)\}$ 66. **endif**	In the update function when $c \neq d$ we first erase the channel c from $ci_{k,x,d}.ch\_dests$ and then, only if $ci_{k,x,d}.ch\_dests$ is empty, the element (k,x,d) is erased from $CI(p)$. Finally, when $c = d$, the element (k,x,d) is directly erased from $CI(p)$.

Table 2 Comparison of causally ordered multi-channel algorithms

Causal Multigroup Algorithms	**Configuration**	**Organization**	**Diffusion**	**Overhead**	**Principle**
CMCA (Our algorithm)	Dynamic	Symmetric	Asynchronous	(worst case) $\Sigma\|g\| : g \in G$	Immediate Inter-Channel dependency relation
CBCAST (Isis)	Dynamic	Symmetric	Asynchronous	(worst case) $\Sigma\|g\| : g \in G$	Time clock compression
Daisy Architecture	Dynamic	Asymmetric	Asynchronous	(always) $l_{\cdot}(s)$	Logical group structure / Rediffusion servers
Causal Separators	Static	Asymmetric	Asynchronous	$\Sigma\|L_s\| : L_s \in P$	Web topology / Rediffusion servers
Low cost approach	Dynamic	Asymmetric	Synchronous by phases	(always) $\|G\|$	Synchronous execution