A formal method to prove ordering properties of multicast systems

C. Toinard,[*,1] G. Florin,[*,1] C. Carrez[*,1]

[*]CEDRIC - Centre d'Etudes et de Recherche en Informatique CNAM
292 rue Saint-Martin 75141 Paris Cedex 03, France

Abstract

We consider three kinds of ordering properties provided by a multicast system. The messages can be delivered to the different recipients according to a local order, a causal order or a total order. In this paper, a formal method is proposed to prove the total and causal order of multicast protocols. A generalized formal result for multicast protocols is obtained. Some addressing features for other formal results are considered. This study allows to prove that existing systems have the required property. The results can be used to define new solutions. Moreover, the method makes use of various ordering properties, well fitted to practical systems, to achieve a better understanding of the ordering processing.
Keywords: *distributed systems, multicast protocols, ordering properties, formal results, practical uses in multicast design.*

1 Introduction

A multicast message is sent to a subset of application entities called *a multicast group*. We consider that the groups can overlap and that any application entity is allowed to send a message to a given group. We consider that there is no global time in the system. These hypotheses correspond to the most general case of multicasting because there is no restriction for the group composition and the emitting entity.

With local order, also called FIFO order, the messages are delivered according to their local relations. With causal order, the causal relations between the messages are satisfied. The classical solution, for getting a causal and total order, builds two layers of protocols. A total order is built using the service offered by a causal protocol. Usually, total ordering is not simplified by the causal protocol. The causal layer needs to exchange a set of historical data, associated with the causal relations, in each message. This *causal*

[1]E-mail: {toinard,gerard,carrez}@cnam.fr

history can be big especially for the general class of multicast that we consider. Well known solutions for multicast, [1] [4] [16] [3] [14], use the classical method with two layers. These systems are designed to achieve a general framework of multicasting where different orderings are mixed. Some improvements often are proposed to reduce the overhead introduced by the causal layer within the multicast context. Some protocols succeed to avoid the causal layer in a broadcast context [23] [1]. A formal result [8] proves the causal property of any totally and locally ordered broadcast protocol. Some protocols are proposed in a particular case of multicasting like a multicast group that is reduced to a singleton [13]. Within the general case of multicasting, it is more difficult to avoid completely the causal layer. In this paper, several formal results are defined for that general case. They are not obtained for a specific protocol. They can be used widely to prove that a totally ordered multicast provides a causal order.

Our main result states that an *identified order that respects the local and direct causal relations provides a causal and total order*. Each message is associated with an identifier. The identifiers are not necessarily managed or used by the protocol for performing the ordering. They can be defined in a formal way according to the behavior of the protocol. The identified ordering guarantees that the messages are delivered according to the identified relations that exist between the messages. A local relation occurs for two messages emitted by the same entity. A direct causal relation corresponds to an immediate relation where a given entity observes a message delivery before emitting a new message. A causal relation corresponds to the transitive closure of local and direct causal relations. It is the transitivity of the identified relation that guarantees the transitive closure of local and direct causal relations. That way, the size of the causal story is reduced. This result seems to be the minimal property that a totally ordered multicast protocol must satisfy in order to be causal.

We also distinguish between some ordering issues and addressing methods. For example, a multicast protocol can perform a multicast by sending the message with a broadcast address on the underlying network. With that broadcast addressing, every protocol entity receives the message. According to different addressing methods, new formal results are defined. The whole set of results allows to evaluate in a very general way whether or not a totally ordered multicast system provides a causal order.

2 Definitions

2.1 Basic notations

We denote by $\mathcal{M}$ the whole set of multicast messages. We distinguish the application entities from the protocol entities that provide the ordered multicast service. We denote by $\mathcal{E}_a$ the set of the application entities and by $\mathcal{E}_p$ the set of the protocol entities.

Multicast group: An application entity sends a multicast message M to a subset of the application entities called the *multicast group* and denoted by $\mathcal{G}(M)$.

Addressing group: To reach the desired multicast group $\mathcal{G}(M)$, the message M is sent on the network to a subset of the protocol entities called the *addressing group* and denoted by $\mathcal{A}(\mathcal{G}(M))$.

Events: Let us consider the transmission of a message M between an emitting application entity i and a recipient application entity j. Two protocol entities x and y perform the multicast service for, respectively, the two application entities i and j. We distinguish the communication events ($E^x(M)$, $S^x(M)$, $R^y(M)$ and $D^y(M)$) occurring on the protocol entities from the events occurring on the application entities ($E^i(M)$ and $D^j(M)$). E, D are emission and delivery events. S and R are sending and receiving events through the underlying network. An interface exists between the protocol and application entities. This interface can be modeled (without loss of generality) using communication queues.

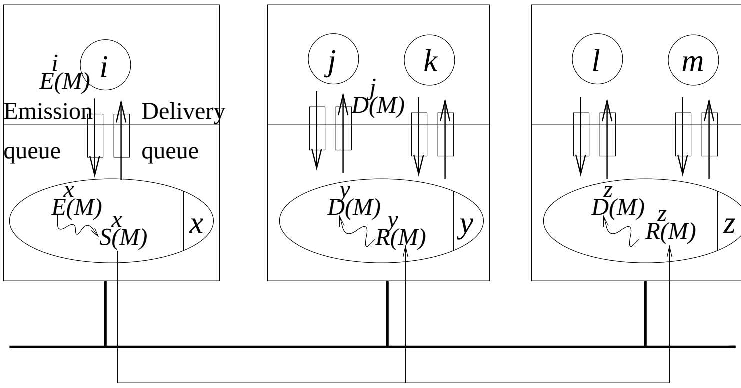

Figure 1: Communication events

Figure 1 defines the transmission of M to $\mathcal{G}(M) = j$ with $\mathcal{A}(\mathcal{G}(M)) = \{y, z\}$.

In this paper, we will only consider the events occurring on the protocol entities because it is not difficult to build a correct interface between the application and the protocol.

The reason is briefly explained.

We will define some relationships $\rightarrow_{\mathcal{R}_p}$ between the messages according to precedence properties between the protocol events. Corresponding relationships $\rightarrow_{\mathcal{R}_a}$ between the messages can be defined for the application according to the application events. The application requires the messages to be delivered according to a given application relationship (formally $M \rightarrow_{\mathcal{R}_a} M' \implies M \rightarrow_{do_a} M'$ where $\rightarrow_{do_a}$ means that M is delivered before M' to every application entity belonging to $\mathcal{G}(M) \cap \mathcal{G}(M')$). The protocol guarantees an ordered delivery by satisfying a protocol relationship (formally $M \rightarrow_{\mathcal{R}_p} M' \implies M \rightarrow_{do_p} M'$ where $\rightarrow_{do_p}$ means that M is delivered before M' by every protocol entity belonging to $\mathcal{A}(\mathcal{G}(M)) \cap \mathcal{A}(\mathcal{G}(M')))$. As the protocol does not observe the application events, the interface must transpose an application relation $M \rightarrow_{\mathcal{R}_a} M'$ to the corresponding protocol relation $M \rightarrow_{\mathcal{R}_p} M'$. Moreover, the reordering built at the protocol level (i.e. $M \rightarrow_{do_p} M'$)

must be conserved by the interface (i.e. $M \rightarrow_{do_a} M'$). That way, the application observes the desired ordering.

In [18], we formally defined the properties that must satisfy the interface. When these properties are satisfied we show that $M \rightarrow_{\mathcal{R}_a} M' \Longrightarrow M \rightarrow_{\mathcal{R}_p} M'$ and that $M \rightarrow_{do_p} M' \Longrightarrow M \rightarrow_{do_a} M'$. The ordering built by the protocol may therefore be considered alone while providing the desired ordering to the application. As this is a minor point, we do not further develop the proof for the interface. In practice, there is no real difficulty in building a correct interface (for example, FIFO queues between the application and the protocol solve the interface).

Further on, we define and denote simply by $\rightarrow_{\mathcal{R}}$ and $\rightarrow_{do}$ the protocol relationships $\rightarrow_{\mathcal{R}_p}$ and $\rightarrow_{do_p}$ for conciseness in the notations. So, the proofs are established for the protocol entities but they satisfy the corresponding application relationships and provide the desired ordering to the application.

2.2 Order properties

Different ordered deliveries are defined using five relations between the messages, namely, a local relation, a direct causal relation, a causal relation, an identified relation and an ordered delivery relation. For most of these relations, we use the *happened before* relation (denoted by $\rightarrow$).

Ordered delivery relation: When two different messages M and M' are delivered in the same order, by every protocol entity that receives these two messages, we say they present an ordered delivery relation,
$M \rightarrow_{do} M' \Longleftrightarrow \forall\ z \in \mathcal{A}(\mathcal{G}(M)) \cap \mathcal{A}(\mathcal{G}(M'))$, $D^z(M) \rightarrow D^z(M')$. This relation is used to define the different ordering properties.

Local order: We define a *local relation* when a given entity x emits a message M before emitting another message M', $M \rightarrow_l M' \Longleftrightarrow \exists\ x \in \mathcal{E}_p$, $E^x(M) \rightarrow E^x(M')$.

With local order, the messages are delivered in order according to their local relations, $M \rightarrow_l M' \Longrightarrow M \rightarrow_{do} M'$.

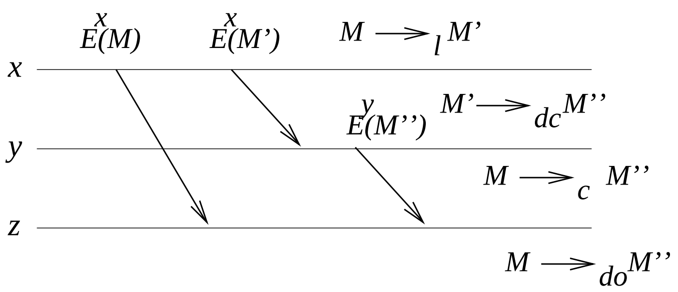

Figure 2: Causal order

Causal order: A *direct causal relation* is defined when a protocol entity x delivers a message M before emitting a message M', $M \rightarrow_{dc} M' \Longleftrightarrow \exists\ x \in \mathcal{E}_p$, $D^x(M) \rightarrow E^x(M')$.

A *causal relation* (equivalent to [12]) is the transitive closure of $\rightarrow_l$ and $\rightarrow_{dc}$,
$M \rightarrow_c M' \iff [\, \exists\, M_0 = M, \ldots, M_{\beta+1} = M' \in \mathcal{M},\ \beta \geq 0] :\ M_{q-1} \rightarrow_l M_q \ \bigvee\ M_{q-1} \rightarrow_{dc} M_q$.

With a causal order, the messages are delivered according to their causal relations,
$M \rightarrow_c M' \Longrightarrow M \rightarrow_{do} M'$

In figure 2, M locally precedes M' and M' precedes M'' according to the direct causal relation (because y delivers M' before emitting M''). By transitivity, M causally precedes M''. As the multicast protocol performs a causal order, M is delivered before M''.

Total order: A protocol, delivering each pair of messages in the same order, is commonly called total ordering, $\forall M, M' \in \mathcal{M},\ M \neq M' : M \rightarrow_{do} M' \ \bigvee\ M' \rightarrow_{do} M$.

It is commonly known that a total order does not satisfy the causal relations. In figure 3, the protocol performs a total order and M'' is delivered before M. But, the ordering is arbitrary and does not respect the causal relation $M \rightarrow_c M''$. That is why the basic solution, to provide a causal and total order, consists in building two layers of protocols. At the lower level, a protocol carries out a causal ordering. Then, a second protocol provides a total order using the causal services. That way, the total order preserves the causal relations.

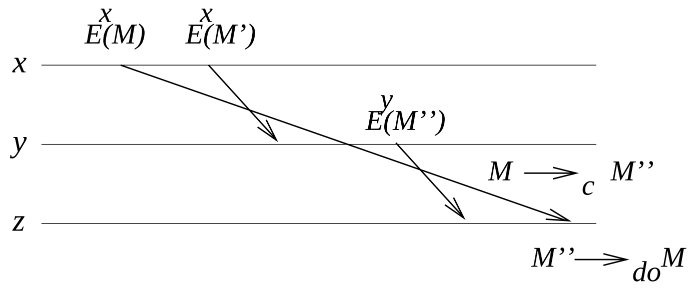

Figure 3: Total order

Identified order: In order to get formal results, we define a new property. The authors of multicast systems do generally not use this definition. But, we will see that many totally ordered protocols satisfy this definition. This order property is defined in two points.

Firstly, according to the behavior of the protocol, we can define a function id where each message gets one identifier that is taken in a space $\mathcal{I}d$ ordered according to an order relation $\leq_{id}$ (*reflexive, antisymetric, transitive*). Thus, we can define an *identified relation* $\rightarrow_{id}$, $M \rightarrow_{id} M' \iff [M \neq M' \ \bigwedge\ id(M) \leq_{id} id(M')]$.

Secondly, the messages are delivered according to their identifiers,
$M \rightarrow_{id} M' \Longrightarrow M \rightarrow_{do} M'$.

Although some protocols use global identifiers (like a global numbering of messages), it does not mean that the protocol directly manages the identifiers to carry out the ordering. For example, the protocol can manage some sequence numbers that are distinct from the identifiers. For the identified ordering, we simply have to define an identified relation that respects the sequencing method, then we have to prove that M is delivered before M' when $M \rightarrow_{id} M'$. Further on, the distinction between the sequencing, performed by the protocol, and the identifiers is augmented. The important point is the transitivity of $\rightarrow_{id}$ when considering different messages (a consequence of the transitivity of $\leq_{id}$).

Identified order satisfying an extra relation: We can define an identified order that respects a relation $\rightarrow_{\mathcal{R}}$, $M \rightarrow_{\mathcal{R}} M' \implies M \rightarrow_{id} M'$.

This definition is used later when considering an identified order that satisfies $\rightarrow_l$ or $\rightarrow_{dc}$. For example, when M locally precedes M' the identifier of M is lower than the identifier of M'.

3 Formal results

The results are defined for multicast. Results for any particular case of multicast protocol, such as a broadcast, are clearly stated. Our point of view is to define the results, give the proofs and show briefly how to use these results.

3.1 Identified order deals with a total order

Theorem 1 *Identified order $\implies$ total order*

Proof: By definition of an identified ordering, each message is associated with an identifier such that, $\forall\ M,\ M' \in \mathcal{M},\ M \neq M' :\ M \rightarrow_{id} M' \bigvee M' \rightarrow_{id} M$. Again by definition, an identified ordering delivers the messages according to their identifiers, $M \rightarrow_{id} M' \implies M \rightarrow_{do} M'$. So, we get a total ordering, $\forall\ M,\ M' \in \mathcal{M},\ M \neq M' :$ $M \rightarrow_{do} M' \bigvee M' \rightarrow_{do} M$. $\square$

The converse is not true. It is possible to get a total ordering that is not an identified ordering.

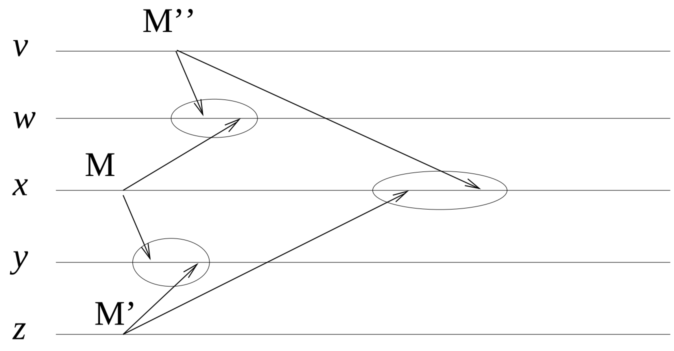

Figure 4: Total order that is not an identified order

Figure 4 gives an example of a protocol that provides a total order. We get three group intersections reduced to a singleton. For the intersection $\mathcal{A}(\mathcal{G}(M)) \cap \mathcal{A}(\mathcal{G}(M')) = y$, we get $M \rightarrow_{do} M'$. For the intersection $\mathcal{A}(\mathcal{G}(M')) \cap \mathcal{A}(\mathcal{G}(M'')) = x$, we get $M' \rightarrow_{do} M''$. For the intersection $\mathcal{A}(\mathcal{G}(M)) \cap \mathcal{A}(\mathcal{G}(M'')) = w$, we get $M'' \rightarrow_{do} M$. With that protocol, if we had an identified ordering we should have: $M \rightarrow_{id} M'$, $M' \rightarrow_{id} M''$, $M'' \rightarrow_{id} M$. But, this is in contradiction with the transitive property of the identified relation $\rightarrow_{id}$. So, the protocol does not provide an identified ordering.

Though a total order is not necessarily an identified order, most of the totally ordered protocols perform an identified order.

How to use the result:

a) The protocol can manage the identifiers like [11] [6] [8] [16] [2]. In these cases, a server of order associates a global sequence number to each message. Such a sequence number, managed by the protocol, corresponds directly to the required identifiers.

b) It can be proved that the messages are associated with identifiers according to the protocol behavior. For example, in [18] [20] we prove that a protocol using a tree of servers (for message ordering), in a similar way to [9], performs an identified order. Each server manages a local sequence number. Each message M is multicast by one server which is the multicast root $R(M)$. $R(M)$ and may be any underlying node propagate M. $R(M)$ can be different from the root of the tree. Any message carries only one sequence number that is allocated locally by a server when forwarding the message to its descendants. So, the sequence number that is carried in the message changes when forwarded by a server. We denote by $L(R(M))$ the sequence number allocated by the root $R(M)$ to initiate the multicast of a message M. We may associate to M an identifier which is the concatenation $C_M[D]$ of D local sequence numbers, where D is the depth of the tree. For each level l from the root of the tree down to $R(M)$, the element $C_M[l]$ is the sequence number $L(R(M_l))$ of the last message M_l initiated by the node at level l and received by $R(M)$ before initiating the multicast of M. For each level l from $R(M)$ to the leaves, the element $C_M[l]$ is null. We may then define $C_M <_{id} C_{M'}$ if and only if $C_M[l] < C_{M'}[l]$, where l is the smallest j such that $id(M)[j] < id(M')[j]$. The proof that $\leq_{id}$ is an order relation and that the protocol performs an identified order can be found in [18] and [20]. The identifiers are not known by the protocol entities and are not used for the reordering. A receiving entity uses the sequence number from its father to perform the reordering.

3.2 Main result

Theorem 2 *An identified order that satisfies both the local and direct causal relations provides a causal and total order*

Proof: We consider a causal relation $M \rightarrow_c M'$.
$M \rightarrow_c M' \iff [\, \exists\, M_0 = M, \ldots, M_{\beta+1} = M' \in \mathcal{M},\ \beta \geq 0] \, [\, M_{q-1} \rightarrow_l M_q \,] \bigvee [\, M_{q-1} \rightarrow_{dc} M_q \,]$

The identified ordering satisfies the local relations: $M \rightarrow_l M' \Longrightarrow M \rightarrow_{id} M'$. It also satisfies the direct causal relations: $M \rightarrow_{dc} M' \Longrightarrow M \rightarrow_{id} M'$. So,
$M \rightarrow_c M' \Longrightarrow [\, \exists\, M_0 = M, \ldots, M_{\beta+1} = M' \in \mathcal{M},\ \beta \geq 0] \, [\, M_{q-1} \rightarrow_{id} M_q \,] \bigvee [\, M_{q-1} \rightarrow_{id} M_q \,]$
$\Longrightarrow M_0 \rightarrow_{id} M_1 \rightarrow_{id} \ldots \rightarrow_{id} M_{\beta+1}$.

As $\rightarrow_{id}$ is transitive, $M \rightarrow_{id} M'$. As we suppose an identified ordering, $M \rightarrow_{do} M'$. □

How to use the result:

a) This result provides a very general way for multicast protocols to get a causal and total order without designing a causal layer. The protocol does not directly manage the transitive closure of $\rightarrow_l$ and $\rightarrow_{dc}$ and does not carry the causal story.

b) Conditions on theorem (2), i.e. $(M \rightarrow_l M' \vee M \rightarrow_{dc} M') \Longrightarrow M \rightarrow_{id} M'$, seem to be the minimal properties that a multicast protocol must satisfy. Counter examples can be given. For brevity, we consider the single following counter example. We assume a protocol that performs an identified order, a local order and a direct causal order, i.e. $(M \rightarrow_{id} M' \vee M \rightarrow_l M' \vee M \rightarrow_{dc} M') \Longrightarrow M \rightarrow_{do} M'$. The direct causal order can be defined like other orderings by satisfying $\rightarrow_{dc}$. Figure 5 illustrates such a protocol. We can assume an identified order where $M \rightarrow_{id} M'$, $M'' \rightarrow_{id} M'''$, $M''' \rightarrow_{id} M$ and $M''' \rightarrow_{id} M'$. Conforming to the identified order, $M''' \rightarrow_{id} M \Longrightarrow M''' \rightarrow_{do} M$. The local order is provided because the local relation $M \rightarrow_l M'$ is respected. The direct causal order is provided because the direct causal relations $M' \rightarrow_{dc} M''$ and $M'' \rightarrow_{dc} M'''$ are satisfied. But, the causal relation $M \rightarrow_c M'''$ is not satisfied. So, an identified order and a local order and a direct causal order do not guarantee a causal order.

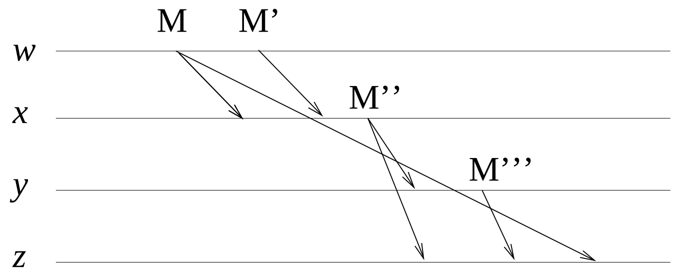

Figure 5: Identified order and local order and direct causal order

c) In [8] a protocol is proposed. We will simply recall the principles of this protocol and give the framework for the proof. This multicast protocol uses a server of order. It requires two messages. First, the emitting protocol entity sends a point to point request (including an application message M) to the server in order to get the multicasting of M. Then, the server processes the request and forwards M by multicasting a second message to the recipient group $\mathcal{A}(G(M))$. The point to point message carries a local sequence number LSN. The emitting protocol entity guarantees that $M \rightarrow_l M' \Longrightarrow LSN < LSN'$ by increasing the local sequence numbers from each application message received through the interface. When processing a new request, the server assigns a global sequence number GSN to the message. The new request gets a global sequence number that is greater than the previous requests. Moreover, the server processes the requests of a given emitter according to the local sequence numbers ($LSN < LSN' \Longrightarrow GSN < GSN'$). The server multicasts a message with the local sequence number LSN and the global sequence number GSN plus one vector $LastGSN[n]$ of sequence numbers (n is the number of entities and $LastGSN[x]$ is the last sequence number that has been sent, before GSN, to x by the server). A receiving entity x delivers the messages according to the global sequence numbers, $GSN < GSN' \Longrightarrow M \rightarrow_{do} M'$.

Each request M is associated with a global sequence number GSN, $id(M) = GSN$. The server guarantees that two identical messages get the same global sequence number. Therefore, we can define an identified relation, $M \rightarrow_{id} M' \Longleftrightarrow GSN < GSN' \Longleftrightarrow M \neq M' \wedge GSN \leq GSN'$. An entity delivers the messages according to their global sequence numbers, $M \rightarrow_{id} M' \Longrightarrow M \rightarrow_{do} M'$. Thus, $M \rightarrow_{l} M' \Longrightarrow M \rightarrow_{id} M'$. $R^{server}(M) \rightarrow D^{x}(M) \rightarrow E^{x}(M') \rightarrow R^{server}(M')$ because the server receives M before x can deliver the message M and x emits the message M' before the server receives it. Thus, $M \rightarrow_{dc} M' \Longrightarrow M \rightarrow_{id} M'$. That way, we get an identified order that respects the local relation and the direct causal relation. The protocol satisfies the theorem (2) and a causal and total order is provided.

The vector $LastGSN[n]$ is not really useful for ordering. The information $LastGSN[x]$, enables a receiver x to know whether a message is missing because x is not recipient or because of a transmission error (lost or de-sequencing). Thus, x can deliver the message as soon as possible without waiting for irrelevant global sequence numbers. We require two messages, one point to point message including one local sequence number and one multicast message including $n+2$ sequence numbers. The size of the information, for a multicast exchange is $O(n)$. We will see that the information size is reduced when using broadcast addressing instead of multicast addressing.

3.3 Result for self-addressing

Here we give a result for a protocol that performs self-addressing. In this case, the emitting protocol entity receives its own message: $\forall M \in \mathcal{M}$, $emitter(M) \in \mathcal{A}(\mathcal{G}(M))$.

The following lemma and its corollary are used to get a new formal result dealing with self-addressing. From the definition of the ordered delivery relation, we show that two conflicting ordered delivery relations occurs if and only if there is no entity that receives these two messages.

Lemma 3 $\mathcal{A}(\mathcal{G}(M)) \cap \mathcal{A}(\mathcal{G}(M')) = \emptyset \Longleftrightarrow M \rightarrow_{do} M' \wedge M' \rightarrow_{do} M$

Proof:

1) When $(\mathcal{A}(\mathcal{G}(M)) \cap \mathcal{A}(\mathcal{G}(M')) = \emptyset)$, the proposition $M \rightarrow_{do} M' \Longleftrightarrow \forall z \in \mathcal{A}(\mathcal{G}(M)) \cap \mathcal{A}(\mathcal{G}(M'))$, $D^{z}(M) \rightarrow D^{z}(M')$ is true because there is no entity z in the intersection. The opposite ordered delivery relation $M' \rightarrow_{do} M$ is true for the same reason. So, $\mathcal{A}(\mathcal{G}(M)) \cap \mathcal{A}(\mathcal{G}(M')) = \emptyset \Longrightarrow M \rightarrow_{do} M' \wedge M' \rightarrow_{do} M$.

2) Converse: By contradiction. We assume two messages such that, $M \rightarrow_{do} M' \wedge M' \rightarrow_{do} M \wedge \neg\, (\mathcal{A}(\mathcal{G}(M)) \cap \mathcal{A}(\mathcal{G}(M')) = \emptyset)$. We consider an entity $z \in \mathcal{A}(\mathcal{G}(M)) \cap \mathcal{A}(\mathcal{G}(M'))$ that satisfies the two ordered delivery relations. So, $D^{z}(M) \rightarrow D^{z}(M') \wedge D^{z}(M') \rightarrow D^{z}(M)$. Conflict. $\Box$

We get the following corollary as a contraposition,

Corollary 4 $\mathcal{A}(\mathcal{G}(M)) \cap \mathcal{A}(\mathcal{G}(M')) \neq \emptyset \iff \neg(M \rightarrow_{do} M' \wedge M' \rightarrow_{do} M)$

Lemma 5 *An identified and local order with self-addressing is an identified order that respects the local relation.*

Proof: By contradiction. We consider a protocol with self-addressing performing an identified and local order. We assume, $M \rightarrow_l M' \bigwedge M' \rightarrow_{id} M$.

1) According to the local order, $M \rightarrow_l M' \implies M \rightarrow_{do} M'$.

2) According to the identified order, $M' \rightarrow_{id} M \implies M' \rightarrow_{do} M$.

1) and 2) $\implies M \rightarrow_{do} M' \bigwedge M' \rightarrow_{do} M$

With self-addressing, $\mathcal{A}(\mathcal{G}(M)) \cap \mathcal{A}(\mathcal{G}(M')) \neq \emptyset$. According to corollary (4), $\mathcal{A}(\mathcal{G}(M)) \cap \mathcal{A}(\mathcal{G}(M')) \neq \emptyset \iff \neg(M \rightarrow_{do} M' \bigwedge M' \rightarrow_{do} M)$. Conflict. □

Lemma 6 *An identified order with self-addressing is an identified order that respects the direct causal relation.*

Proof: By contradiction. We consider a protocol with self-addressing performing an identified order. We suppose, $M \rightarrow_{dc} M' \bigwedge M' \rightarrow_{id} M$. Let $e' = emitter(M')$ be the process that emits the message M'. With self-addressing, $e' \in \mathcal{A}(\mathcal{G}(M)) \cap \mathcal{A}(\mathcal{G}(M'))$.

1) According to the direct causal relation $M \rightarrow_{dc} M'$, $D^{e'}(M) \rightarrow E^{e'}(M')$. According to self-addressing e' receives and delivers the message M', $E^{e'}(M') \rightarrow D^{e'}(M')$. So, $D^{e'}(M) \rightarrow D^{e'}(M')$. With an identified order, a total order is provided and every entity delivers the two messages according to same order, $D^{e'}(M) \rightarrow D^{e'}(M')$. That way, we get the ordered delivery relation, $M \rightarrow_{do} M'$.

2) According to the identified order, $M' \rightarrow_{id} M \implies M' \rightarrow_{do} M$.

1) and 2) $\implies (M \rightarrow_{do} M' \bigwedge M' \rightarrow_{do} M)$.

According to lemma (3), $(M \rightarrow_{do} M' \bigwedge M' \rightarrow_{do} M) \iff \mathcal{A}(\mathcal{G}(M)) \cap \mathcal{A}(\mathcal{G}(M')) = \emptyset$. With $e' \in \mathcal{A}(\mathcal{G}(M)) \cap \mathcal{A}(\mathcal{G}(M'))$, $\mathcal{A}(\mathcal{G}(M)) \cap \mathcal{A}(\mathcal{G}(M')) \neq \emptyset$. Conflict. □

Theorem 7 *An identified and local order with self-addressing provides a total and causal order*

Proof: Assume an identified and local order with self-addressing. According to lemma (5), an identified and local order with self-addressing is an identified order that respects the local relation. According to lemma (6), an identified order with self-addressing is an identified order that respects the direct causal relation. So, the identified ordering respects both the local and direct causal relations. According to theorem (2), we get a causal and total ordering. □

How to use the result:

a) With theorem (7), a protocol using a self-addressing avoids the management of $\rightarrow_{dc}$. Thus, the causal history is shortened. The only information required concerns the local relations. Generally, we need a single local sequence to provide a local order.

b) As previously explained, in [18], we show that a protocol like [9] performs an identified ordering. But, [9] does not provide a local order and does not perform self-addressing.

So, this protocol cannot provide a causal order. In [18][20], a new protocol uses a tree of servers and provides a total and local order through a tree of servers. The total order requires the multicasting of one message that carries only one sequence number. As previously explained, we get an identified order but the identifiers are not used by the protocol entity for the ordering. The protocol entities do not know the identifiers. Our result allows to treat the direct causal relations by self-addressing. It avoids managing the transitive closure of local and direct causal relations.

3.4 Result for recording entity and self-addressing

We consider a protocol where a particular protocol entity r receives all the messages: $\forall M \in \mathcal{M}$, $r \in \mathcal{A}(\mathcal{G}(M))$. As it receives all the messages, r reorders them. Moreover, an emitting protocol entity performs self-addressing.

Lemma 8 *With a recording entity r , we can define an identified relation such that,* $D^r(M) \rightarrow D^r(M') \iff M \rightarrow_{id} M'$.

Proof: Let r be the recording entity that receives and delivers all the messages M_0, $\ldots$, M_n existing in the system.

1) The messages M_0, $\ldots$, M_n can be ordered on r according to their delivery events: $D^r(M_0) \rightarrow D^r(M_1) \rightarrow \ldots \rightarrow D^r(M_n)$. We can define a function id that associates each message with an identifier according to their delivery position on r. Two different messages never get the same identifier because it would mean that they were associated with the same delivery event on r. The general definition of $\rightarrow_{id}$ is, $M \rightarrow_{id} M' \iff [M \neq M' \wedge id(M) \leq_{id} id(M')]$. As two messages cannot get the same identifier, $M \rightarrow_{id} M' \iff [M \neq M' \wedge id(M) <_{id} id(M')]$. By definition of id, two different messages M, M' such that $D^r(M) \rightarrow D^r(M')$ satisfy $id(M) <_{id} id(M')$. So, $D^r(M) \rightarrow D^r(M') \implies M \rightarrow_{id} M'$.

2) The converse is proved by contradiction. Assume $M \rightarrow_{id} M'$ and $D^r(M') \rightarrow D^r(M)$. According to 1) $M \rightarrow_{id} M' \iff [M \neq M' \wedge id(M) <_{id} id(M')]$. According to 1), $D^r(M') \rightarrow D^r(M) \implies M' \rightarrow_{id} M \implies [M' \neq M \wedge id(M') <_{id} id(M)]$. We get, $[id(M) <_{id} id(M')]$ and $[id(M') <_{id} id(M)]$. Conflict.

1) and 2): $D^r(M) \rightarrow D^r(M') \iff M \rightarrow_{id} M'$. $\Box$

Lemma 9 *A total ordering with a recording entity is an identified ordering*

Proof: According to lemma (8), we can define a relation $\rightarrow_{id}$ with a recording entity r such that, $D^r(M) \rightarrow D^r(M') \iff M \rightarrow_{id} M'$. With a total ordering, $D^r(M) \rightarrow D^r(M') \implies M \rightarrow_{do} M'$. So, $M \rightarrow_{id} M' \implies M \rightarrow_{do} M'$. $\Box$

Theorem 10 *A total and local ordering with recording entity and self-addressing provides a total and causal ordering*

Proof: According to lemma (9), a total ordering with a recording entity is an identified ordering. According to theorem (7), an identified and local ordering with self-addressing provides a total and causal ordering. So, a total and local ordering with a recording entity and self-addressing provides a total and causal ordering. □

We do not know any multicast protocol that uses this result. But, the result of the next section is a consequence of (10). So, the use of the following result can be considered as an implementation of (10).

3.5 Result for addressing everybody

Each message is addressed to every protocol entity: $\forall M \in \mathcal{M}$, $\mathcal{A}(\mathcal{G}(M)) = \mathcal{E}_p$. Each message is sent through a broadcasting on the underlying network and all the protocol entities receive and reorder it. But, each message is not necessarily delivered to every application entity. The method of addressing is a broadcast but the application performs a multicast with different multicast groups. The protocol reorders and filters the messages for the application.

Theorem 11 *A total and local order with addressing to everybody provides a total and causal order*

Proof: With a protocol performing addressing to everybody, any protocol entity z is a recorded entity because it receives all the messages. Moreover, the protocol performs self-addressing because the emitter belongs to $\mathcal{A}(\mathcal{G}(M)) = \mathcal{E}_p$. We get a total and local order with a recording entity and self-addressing. According to (10), a causal and total order is provided. □

How to use the result:

a) It provides a simple way to get a causally and totally ordered multicast. It avoids building a causal layer. Locally and totally ordered broadcast protocols can be extended to a multicast service by filtering the messages. The causal property is then got by our result. There is a counter part to the simplicity because every station is quite loaded by broadcast.

b) The protocol that we defined in [8] can be modified using this result. The second message is then broadcast by the server and carries only two sequence numbers (one local sequence number LSN and one server sequence number GSN). This way, with broadcast addressing, no vector $LastGSN[n]$ is required to distinguish between missing and non-addressed messages. Thus, a broadcast addressing reduces the size of the information per broadcast exchange to 3 (one local sequence number in a point to point message plus one local sequence number and one global sequence number in a broadcast message). In contrast, we explained that a multicast addressing costs $O(n)$. Other local and total broadcasts can be modified in the same way to get a causal and total multicast. For example, [6] [2] use a logical ring to perform a total and local broadcast. They can use the result to get a causal and total order for multicast within a broadcast domain. It requires simply the protocol to carry out a total and local order. When a protocol entity decides to deliver a message, the modification consists in filtering the messages to the recipient application entities (possibly none or multiple recipients).

In another way, a sequencer technique like [11] can also make use of it. In that case, the framework must be changed because the authors consider one sequencer that performs a total and local order for one group. One sequencer for all the groups must be used to get benefit of our result.

3.6 A broadcast result

Theorem 12 *A totally and locally ordered broadcast provides a total and causal order*

Proof: With a broadcast, every application entity receives each message M, $\mathcal{G}(M) = \mathcal{E}_a$. For this purpose, every protocol entity receives each message, $\mathcal{A}(\mathcal{G}(M)) = \mathcal{E}_p$. So, a broadcast is performed with broadcast addressing. According to (10), we get a causal and total order. $\Box$

How to use the result:

a) This result was first proved in [8]. Here, it is simply another kind of proof as a consequence of more general results for multicast. In [8], we proved the causal property of many totally ordered broadcast previously published, like [6] [11], because generally a local order is also provided. And obviously, it proves the causal property of protocol like [1] published after this result. In this latter case, no specific proof is required because the protocol provides a causal order as a consequence of [8] that is a very general result for broadcast systems.

b) In [19][22], this result is applied to the FIP (Factory Instrumentation Protocol) field bus. We show that the FIP broadcast provides a total and local order at low speed. So, we get a causal and total property. At high communication speeds, our result is used for defining an extension on top of the FIP system. Thus, causal and total order is also provided at high speed. This is an interesting ability for the FIP field bus that was standardized for providing time properties but not order properties. Moreover, this result fits well with the context of real time systems addressed by FIP because we do not introduce extra overhead, in particular, for low communication speeds. The extension for high speeds do not require to modify the FIP network controller.

4 Conclusion

In this paper, we established formal results that define a set of circumstances under which a total order satisfies the causal relations between the messages. These results allow designers to prove the causal property of multicast systems. They can be used to see the relationships between total ordering and causal ordering. They can be used to give extensions and define new protocols. The causal layer is avoided and the information piggy-backed in the message is reduced. So, the paper defines a formal method to define whether or not a totally ordered protocol provides a causal order. That way, the causal property is got without using a specific layer of protocol. This paper mainly presents the results and the proofs. The uses are briefly described in order to show the relevance of the results.

References

[1] L.E. Moser, P.M. Melliar-Smith, D.A. Agarwal, R. Budhia and C. Lingley-Papadopoulos. Totem: A Fault Tolerant Multicast Group Communication System. Communications of the ACM, 39(4):54-63, April 1996.

[2] Y. Amir, L.E. Moser, P.M. Melliar-Smith, D.A. Agarwal, and P. Ciarfella. Fast Message Ordering and Membership Using a Logical Token-Passing Ring. In International Conference on Distributed Computing Systems, May 1993.

[3] Y. Amir, D. Dolev, S. Kramer, D. Malki. Transis: A communication subsystem for high availability. In Annual Symposium on Fault Tolerant Computing, number 22, pages 76-78, July 1992.

[4] K. Birman, A. Schiper, P. Stephenson. Lightweight Causal and Atomic Group Multicast. ACM TOCS Vol 9, Number 3, August 1991.

[5] R. Baldoni, R. Friedman, R. Van Renesse. The Hierarchical Daisy Architecture for Causal Delivery. Cornell University Technical Report TR96-1610, September 1996.

[6] J. Chang, N. Maxemchuck. Reliable broadcast protocols. ACM Transactions on Computer Systems, 2(3):251-273, August 1984.

[7] P. Ezhichelvan, R. Macedo an S. Shrivastava. Newtop: A fault-tolerant group communication protocol. Proceed. 15th Int. Conf. on Distributed Computing Systems, pages 296-306. IEEE 1995.

[8] G. Florin, C. Toinard. A new way to design causally and totally ordered multicast protocols. ACM Operating Systems Review, Volume 26, Number 4, October 1992.

[9] H. Garcia-Molina, A. Spauster. Ordered and Reliable Multicast Communication. ACM TOCS Vol 9, Number 3, August 1991.

[10] N. Huleilhel. Efficient Ordering of Messages in Wide Area Networks. Master Thesis, http://www.cs.huji.ac.il/labs/transis/publications.html, March 1996.

[11] M.F. Kaashoek, A.S. Tanenbaum, S.F. Hummel, H.E. Bal An efficient reliable broadcast protocol. Operating Systems Review, 23(4):5-19, October 1989.

[12] L. Lamport. Time, clocks and the ordering of events in a distributed system. CACM Vol 21, Number 7, July 1978.

[13] F. Mattern, S. Fünfrocken. A Non-Blocking Lightweight Implentation of Causal Order Message Delivery. Lecture Notes in Computer Science, 938:197-213, 1995.

[14] S. Mishra, L.L. Peterson, R.D. Schlichting. Consul: a communication substrat for fault-tolerant distributed programs. Distrib. Syst. Engng 87-103, 1993.

[15] R. Van Renesse, T.M. Hickey, K.P. Birman. Design and Performance of Horus: A lightweight Group Communications System. Technical Report TR-94-1442, August 1994.

[16] R. Van Renesse, K.P. Birman, R. Friedman, M. Hayden, and D. Karr. A Framework for Protocol Composition in Horus. In proceedings of ACM Symposium on Principles of Distributed Computing, August 1995.

[17] L.E.T. Rodrigues, H. Fonseca, P. Verissimo. A dynamic hybrid protocol for total order in large-scale systems. In proceedings of 16Th International Conference on Distributed Computing Systems, Hong-Kong, May 1996.

[18] C. Toinard. Etude des diffusions ordonnées vers des groupes. PhD Thesis, University of Paris 6, February 1993.

[19] C. Toinard, N. Chevassus. De nouvelles architectures de communication pour le contrôle de procédés. Real Time Systems (RTS'96), Paris 1996, Proceedings in Teknea Editors France.

[20] C. Toinard, M. Dumas, R. Dirlewanger. Une architecture répartie et réaliste de communication vers des groupes garantissant un ordre causal et total grâce à une arborescence de serveurs. Technical Report LaBRI. Juin 1996.

[21] C. Toinard, N. Chevassus. CORBA for distributed and fault-tolerant process control. ECOOP97, Workshop on CORBA Implementation, Use and Evaluation, Jyvaskyla, Finland, 10 june 1997.

[22] C. Toinard, N. Chevassus. An object communication architecture for real time distributed process control. In Proceedings of ECOOP97, Workshop on OO Technology and Real Time Systems, Jyvaskyla, Finland, 10 june 1997.

[23] P. Verissimo, L. Rodrigues, M. Baptista. AMp: A highly parallel atomic multicast protocol. In Proceedings of SIGCOM'89 Symposium, pages 83-93. ACM, 1989.