A FAULT-TOLERANT CAUSAL BROADCAST ALGORITHM TO BE APPLIED TO UNRELIABLE NETWORKS

Eduardo Lopez Dominguez[1], Jorge Estudillo Ramirez[1], Jean Fanchon[2], Saul E. Pomares Hernandez[1]
[1]Computer Science Department
National Institute of Astrophysics, Optics and Electronics (INAOE)
Luís Enrique Erro #1, 72840 Tonantzintla, Puebla, México.
{edominguez, jestudillo, spomares}@inaoep.mx
[2]Laboratory for Analysis and Architecture of Systems of CNRS
7 Av. Colonel Roche, 31077 Toulouse Cedex, France.
fanchon@laas.fr

ABSTRACT
In this work we propose an efficient real-time causal broadcast algorithm with fault tolerance to unreliable networks. The algorithm allows for the delivery of causal messages with recovering capabilities on real-time systems by using the concept of redundancy. Redundancy, in our work, is calculated based on the causal distance. The concept of causal distance was first introduced to detect the immediate dependency relation (IDR,) which presents a distance equal to one. We extend the causal distance idea in order to add minimal redundancy information (d>1). With information redundancy we ensure the causal delivery property even in the presence of lost messages. Our algorithm is suitable to be used in real-time systems because it has the characteristic of forward error recovery.

KEY WORDS
Distributed systems, causal order, causal distance, immediate dependency relation, forward error recovery, unreliable networks.

1 Introduction

Many works concerning protocols of causal delivery of messages [1-12] exist. Most of the previous causal algorithms assume a reliable transmission environment, while only some works consider the case of delay in the delivery of messages through the use of Δ-time [4]. However, some environments exist where we need to consider an unreliable transfer network in order to satisfy some temporal restrictions, as in the case of the transmission of real-time continuous media. First of all, there is no time for the re-transmission of lost messages, and/or this kind of system tolerates a certain degree messages lost, which is specified on the QoS restrictions of the system.

In this work we propose a fault-tolerance causal algorithm to be applied to unreliable transmission networks. We consider the possibility of lost messages during the transmission which could alter the causal delivery of messages. In our proposal, the algorithm is able to be re-established by itself in the presence of lost messages in a decentralized manner. We extend the minimal causal broadcast algorithm presented in [2] to be applied to unreliable networks. This minimal algorithm is based on the IDR (immediate precedence relation) relation. The IDR relation identifies the necessary and sufficient control information to be attached to each message to ensure the causal order in a *reliable network*. In order to support the loss of messages, we introduce redundancy on the control information attached per message. Redundancy is calculated based on the "causal distance" between messages. Two messages immediately precede each other when their causal distance is equal to one and an intermediate message increases this distance. Instead of restricting the causal information to its immediate predecessors, we can attach to a message all the messages up to a maximal distance in its causal past. The main benefit is that it increases the degree of tolerance of lost messages; a larger distance will increase the redundancy in the control information sent in the system. We performed an analysis of the behavior of the IDR relation on the minimal algorithm. We found that there exists inevitable and inherent redundancy information in the case of concurrent messages. Taking this into account, we only introduce extra redundancy in our causal algorithm when the number of concurrent messages sent is less than the causal distance previously established.

Our algorithm is suitable to be used in real-time systems since it presents the characteristic of recovery without the retransmission of lost messages. The present work is one of the first works on causal algorithms oriented towards the forward error recovery mechanism.

The rest of the article is structured in the following way: Section 2 presents the most relevant related works concerning the loss/recovery of information in unreliable networks. In Section 3, the system model is described and the background information is presented. Next, in Section 4 we present an informal analysis of the behavior of the causal distance. The development of the proposal and the functional algorithm is described in Section 5. Finally, the conclusions and the future works are presented in Section 6.

2 Related work

Many works concerning real-time causal algorithms [1, 2, 3, 4, 5, 6, 7] can be cited. These works can be classified into two categories. In the first category the algorithms are conceived over reliable networks, and only some of them consider transmission delay. The second category considers algorithms that work in unreliable networks; most of these consider delays and dropped messages. In this work we only present works of the second category, which are the ones we are interested in.

The most important works about fault-tolerance over unreliable networks are [4, 5, 11, 12]. In [5] the author incorporates a version field into a vector clock to create the Fault-Tolerant Vector Clock (FTVC). He uses a history mechanism to detect orphan states and obsolete messages. After a failure, the process restores its last checkpoint from the stable storage. In order to inform the other processes about the failure, all received logged messages are replied in the same order in which they were received.

Other works related to fault-tolerance in unreliable networks are [11, 12]; both of these consider a causal server. In this approach, when some process fails, the server sends it its causal information to recover it. In others words, the causal server is responsible for the causal recovery of the system.

3 Background

Causal ordering delivery is based on the causal precedence relation defined by Lamport [8]. This relation is a partial order relating the sending and delivery events executed by a set of communicating sequential processes denoted c. The sending of a message *m* is denoted *send(m)*, the delivery of *m* to the process *k* is denoted *delivery(k,m)* and *p(e)* denotes the process where the event *e* occurs.

Definition 1: The causal precedence relation, denoted by $\rightarrow$, is the partial order generated by the following pairs:

1. $e \rightarrow e'$ for all e,e' such that $p(e)=p(e')$ and e occurs before e' on $p(e)$
2. $send(m) \rightarrow delivery(k, m)$ for every message m and process k

The causal precedence is extended to messages in the following way: $m \rightarrow m'$ iff $send(m) \rightarrow send(m')$.

A behaviour or a set of behaviours satisfy *Causal broadcast delivery* if when the diffusion of a message *m* causally precedes the diffusion of a message *m*', then the delivery of *m* causally precedes the delivery of *m*' for all participants that belong to *c*. It is defined as follows [2] .

Definition 2 Causal broadcast delivery :

$$\textbf{IF } send(m) \rightarrow send(m'), \textbf{ then } \forall k \in c : \\ deliver(k,m) \rightarrow delivery_k(k,m')$$

3.1 The Immediate Dependence Relation

The Immediate Dependency Relation (IDR) [2] is the propagation threshold of the control information *CI*, regarding the messages sent in the causal past that must be transmitted to ensure a causal delivery. We denote it by $\downarrow$, and its formal definition is the following

Definition 3: 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''* belonging to *M* exists (*M* is the set of messages of the system), such that *m''* belongs at the same time to the causal future of *m*, and to the causal past of *m*'.
This relationship is important since we show that if the delivery of messages respects the order of their diffusion for all pairs of messages in IDR, then the delivery will respect the causal delivery for all messages. This property is formalized as follows:

Property 1:

$$\text{If } \forall\, m,m' \in M,\ m \downarrow m' \Rightarrow \forall k \in c :$$
$$delivery(k,m) \rightarrow deliver(k,m')$$
$$\text{then } m \rightarrow m' \Rightarrow \forall k \in c :$$
$$delivery_{(k,}m) \rightarrow delivery_{(k,}m')$$

Causal information that includes the messages immediately preceding a given message is sufficient to ensure a causal delivery of such message.

3.2 Causal distance

Intuitively the causal distance between two causally dependent messages is the greatest number of pairwise dependent messages sent between them plus one. Formally it is defined as follows:
:

Definition 4. The distance $d(m,m')$ is defined for any pair of messages m and $m' \in M$ such that $m \rightarrow m'$: $d(m,m')$ 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 $m_i \downarrow m_{i+1}$ for all $i = 0 \ldots n-1$.

4 Causal Distance Analysis

There are two possible cases in the transmission of causal events: the serial case and the concurrent case.

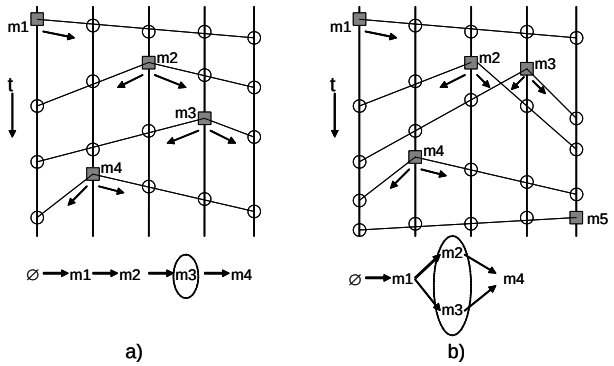

Figure 1. Example scenarios and their associated graphs

In the serial case (Fig. 1a), when the causal distance is equal to one (IDR relation), redundancy does not exist in the control information sent. In this case, it is impossible to recover the system when a message is lost. In the case of concurrent messages (Fig. 1b), the redundancy is directly proportional to the number of concurrent messages. In our case, the redundancy defines the number of times the information about a causal message is sent in the system. For example, in Figure 1b (using d=1) the concurrent messages m_2 and m_3 send information about message m_1 which has a d=1 with both of them. As we can see, the information about m_1 is sent twice. In this case, if one of the two concurrent messages is lost, the system is causally recovered through the information provided by the other concurrent message received. However, if both messages are lost, the system cannot be recovered.
In order to increment the degree of fault-tolerance, we introduce extra redundancy to be timestamped to each message. For example, in the serial case, by using d=2, the system can causally recover in the presence of one lost message. With the IDR relation, messages only send information about the immediate predecessors (d=1). For example, in Figure 1a, m_2 sends information only about m_1, m_3 sends information only about m_2, and m_4 sends information only about m_3. With d=2 the control information sent per message corresponds to the messages which have a causal distance equal to two with the current

message. For example, message m_3 must send information about messages m_2 and m_1 and lastly m_4 must send information about m_3 and m_2.

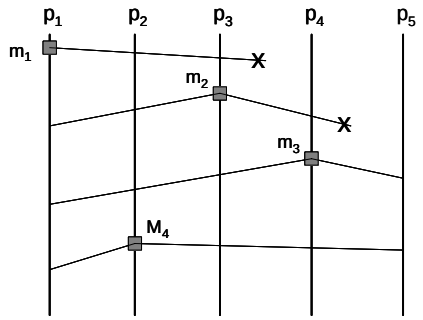

Figure 2. Faulty scenario

5 The Algorithm

In order to explain how the redundancy is increased, we present the following faulty scenario.
Consider the scenario presented in Figure 2 with a causal distance d=2. The diffusion of m_1 is not received by the processes p_4 and p_5. Process p_3, after the reception of m_1, sends m_2 with control information about m_1. Process p_4 receives m_2 and through the control information attached to m_2 about m_1 it is able to detect that message m_1 has been lost. Then p_4 proceeds to update its logical vector and delivers message m_2. Next, process p_4 sends message m_3 which carries causal information about m_2 and m_1 with distances d=1 and d=2, respectively. Process p_5 which neither received m_1 nor m_2, now receives m_3. It analyzes the control information attached to message m_3 and it can determine that two messages have been lost. It proceeds to deliver and to update its vector time clock. Finally, we can see that when m_4 arrives to p_5, the causal order $m_1 \rightarrow m_2 \rightarrow m_3 \rightarrow m_4$ is achieved.

Note. In this algorithm, for simplicity, we consider that if a message arrives out of order or if it is delayed, it is determined to be lost and is automatically discarded. In future works, this algorithm will be easily adapted to consider a deadline associated to each message.

5.1 Data structures

The structures used in this work have few changes compared to the structures used in [2]. Only the CI structure was modified by adding the distance field *d*. The data structures used in the algorithm are:

- $VT(p)[]$ is the vector timeclock. The size of the vector is equal to n. It is here that we keep track of the number of messages diffused by the participant.
- The structure of the control information $CI(p)$ is a set of entries (k, t, d). Each entry in $CI(p)$ denotes a message that is not ensured by participant p of being delivered in a causal order. The entry (k, t, d) represents a diffusion by participant k at a logical local timeclock $t = VT(p)[k]$, and d is potentially the causal distance.
- The structure of a message m , is a quadruplet $m = (i, t, message, H(m))$, where i is the participant identifier, $t=VT(p)[i]$ is the logical local timeclock at node i, *message* is the message in question, and $H(m)$ contains the set of all entries that have an IDR with m.
- The *dist_def* variable is the causal distance predetermined.

5.2 Algorithm codification

1. **Initially,**
2. $VT(p)[j] = 0 \ \forall \ j:1\ldots n$.
3.. $CI(p) \leftarrow \varnothing$
4. **For each diffusion of message** send(m) **at** p
5. $VT(p)[i] = VT(p)[i] +1$
6. For each $(k,t,d) \in CI(p)$
7. $(k,t,d) \leftarrow (k,t,d+1)$
8. $H(m) \leftarrow H(m) \cup \{k,t\}$
9. endfor
10. $m = (i, t=VT(p)[i], message, H(m))$

11. **Diffusion :** send(m)
12. CI(p)←CI(p) ∪ {(k,t,d=0)}
13. For all (k,t,d) ∈ CI(p) if $d = dist\_def$.
14. then
15. CI(p)←CI(p) / (k,t,d)
16. endif
17. endfor.
18. **For each reception** receive(m) **at** p
    m=(k,t,*message*, H(m))
    /* *To enforce a causal delivery of m* */
    /**Delivery condition**/
19. if not (t =VT(p)[k] +1 and ∀ (l,x)∈ H(m):
    x≤ VT(p)[l])
20. then
    /* *Detection of lost message and update of vectors*/
21. For all (l,x) ∈ H(m))
22. if (x >VT(p)[l] then VT(p)[l]=x
23. endif
24. endif
    /* causal delivery of message*/
25. **Delivery**: delivery(m)
26. VT(p)[k] = VT(p)[k] +1
27. For all (l,x) ∈ H(m) if ∃d : (l,x,d) ∈ CI(p)
28. then
29. (l,x,d) ←(l,x,d+1)
30. endif.
31. CI(p)←CI(p) ∪ {(k,t,d=0)}
32. For all (l,x,d) ∈ CI(p) if d=dist_def.
33. then
34. CI(p)←CI(p) / (l,x,d)
35. endif

5.3 Example

Consider the group of participants g={p_1, p_2, p_3, p_4, p_5} and the diffusion of message m_4 to p_1. Before the delivery of m_4 to p_1, CI_1={(1,1,0)} and VT(p_1)=(1,0,0,0,0) (See Fig. 3). These values are deduced from the fault-tolerance broadcast protocol (Figure 3). The complete run is shown in Appendix A.

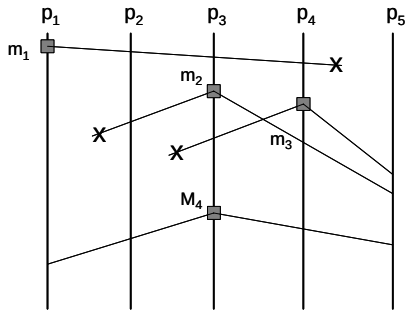

Figure 3. Scenario with lost messages

Difussion of message m_4 by p_3.

Line 5. The value of vector time VT(p_3)[3] is incremented by one. This ensures the sequential delivery of all messages m diffused by the same participant p_k.

Lines 6-9. We increase the distance value d for each triplet of CI_3={(3,1,1),(4,1,1)}. This is done in order to know how many times the messages which have causal order at m_4 have been diffused. In the same loop (**2iv**) we assign at H(m) only the tuple (k,t) from each element of CI_3, H(m)={(3,1),(4,1)}. These entries in H(m) are responsible for the detection of lost of messages, and afterwards, they are used to update the vector time in order to maintain a consistent causal state.

Line 11. Diffusion of message m_4=(3,2,*event*, {(3,1),(4,1)}), send(m_4).

Line 12. Now CI_3 is updated with control information from message m_4, CI_3={(3,1,1),(4,1,1),(3,2,0)}.

Later on, in **Line 13.** we search for a triplet with distance d=2 and we deleted it in **Line 15** of CI_3. When a triple in CI_k has a distance d=2, it indicates that it has send its control information in the network twice and it is not necessary to send this control information again. In CI_3 there is not a triplet with d=2; therefore, no action is necessary.

Delivery of message m_4 to p_1.

Each time a participant receives a message, it verifies that the message satisfies the delivery condition, which verifies that the reception message fulfills the causal property. In this case, message m_4=(3,2,*event*,{(3,1),(4,1)} does not satisfy the delivery condition **Line 19** because t=2 and VT(p_1)[3]=1 and the conditions t=VT(p)[k] and as $t \leq$ VT(p)[l] are not satisfied, as same corresponding to the information of H(m). Then, the algorithm executes the procedure of lost messages and proceeds to update the vector time of p_1. In this step, the vector VT(p_1) is updated with de causal information from H(m), resulting in VT(p_1)=(1,0,1,1,0).

Line 25. Now message m_4 is causally delivered and in **Line 26** the vector is increased by one in VT(p_1)[3] resulting in VT(p_1)=(1,0,2,1,0).

CI is updated in the following manner.

Line 27. If (l,x) exists in H(m) which correspond to a triplet (k,t,d) on CI_1 where l=k and t=x, we increase the value of d by 1. In this case, the message m_4 do not have in its H(m) any tuple which corresponds with the values of CI_1.

Line 31. We updated CI_1 adding control information from m_4, resulting in CI_1={(1,1,0),(3,2,0)}.

Linea 32-35. Lastly, we check that there does not exist any triplet with d=2 in CI_1. If it exists, we delete this triplet. In this example in the CI_1 there is not a triplet with d=2; therefore, no action is taken.

6 Conclusions and future works

We presented an algorithm that tolerates the loss of messages by increasing the redundancy in the control information timestamped per message. The redundancy is calculated by the causal distance between events. The importance of this proposal is the recovery of the system in the presence of lost messages in a forward error correction manner. The algorithm presented does not considered restrictions of time, such as transmission delays or lifetime constraints. For future works, it can incorporate restrictions of time to consider the cases previously mentioned.

7 References

[1] R. Baldoni, R. Prakash, M. Raynal, M. Singhal, Efficient causally ordered comunications for multimedia real-time applications, In Proceedings of the 4th International Symposium on Hight performance Distributed computing, pp 140-147,Whasington, D.C. Aug. 1995.

[2] S. Pomares Hernández, J. Fanchon, K. Drira, The Inmediate Dependency Relation: An Optimal Way to Ensure Causal Group Communication. annual Review of Scalable Computing , Vol. 6, Series on Scalable Computing. Ed. Y.C. Kwong, World Scientific, No. ISBN 981-238-902-4, 2004, Chapter 3, pp. 61-79.

[3] L. Rodríguez, P. Verissimo, Causal Separators and Topological Timestamping: an Aproach to Support Causal Multicast in Large-Scale Systems, proc. 15th International Conference on Distributed Computing Systems, Vancouver, British Columbia, Canada, 1995.

[4] Roberto Baldoni and Achour Mostefaoui and Michel Raynal, Causal Deliveries in Unreliable Networks With Real-Time Delivery Constraints, Journal of Real-Time Systems, volume 2, Issue 1, Pages 308--321, 1996
[5] O. Damani, V. K. Garg, How to Recover Efficiently and Asynchronously when Optimism Fails, 'Proc. IEEE International Conference on Distributed Computing Systems,Hong Kong, May 1996, pp. 108 – 115.
[6] Paulo Veríssimo, Causal Delivery Protocol in Real-Time Systems: a Generic Model. Instituto de Engenharia de Sistemas e Computadores, Lisboa-Portugal, Technical University of Lisboa.
[7] Takayuki Tachikawa, Makoto Takizawa, Δ-Causality in Wide – Area Group Communications. Proc. 1997 Int'l Conf. on Parallel and Distributed Systems (ICPADS-97), 260-267, 1997.
[8] L. Lamport, Time Clocks and the Ordering of Messagesin Distributed Systems, Communications ACM 21(7), pp. 558-565, 1978.
[9] K. Birman, The Process Group Approach to Reliable Distributed Computing, Communications of the ACM, Vol. 36, No. 12, pp 36-53, 1993.
[10] R. Prakash, M. Raynal, M. Singhal, An Adaptive Causal Ordering Algorithm Suited to Mobile Computing Environment, Journal of Parallel and Distributed Computing , pp. 190-204, Mar. 1997.
[11] Roberto Baldoni, Roy Friedman, Robbert van Renesse, The Hierarchical Daisy Architecture for Causal Delivery, In Proceedings of the 17 th IEEE International Conference on Distributed Systems, pp 570-577, May 1997.
[12] Kenneth P. Birman, Thomas A. Joseph, Reliable Communication in the Presence of Failures, ACM Transactions on Computer Systems (TOCS), Volume 5, Issue 1, February 1987, Pages: 47 – 76, ISSN:0734-2071

Appendix A

p_1	p_2	p_3	p_4	P_5
VT_1=(0,0,0,0,0) $CI_1 \leftarrow \varnothing$	VT_2=(0,0,0,0,0) $CI_2 \leftarrow \varnothing$	VT_3=(0,0,0,0,0,) $CI_3 \leftarrow \varnothing$	VT_4=(0,0,0,0,0) $CI_4 \leftarrow \varnothing$	VT_5=(0,0,0,0,0) $CI_5 \leftarrow \varnothing$
diffusion(m_1) VT_1=(1,0,0,0,0) m_1=(1,1,event, $\varnothing$) send(m_1) CI_1={(1,1,0)}				
	reception(m_1) delivery(m_1) VT_2=(1,0,0,0,0) CI_2={(1,1,0)}	reception(m_1) delivery(m_1) VT_3=(1,0,0,0,0) CI_3={(1,1,0)}	reception(m_1) delivery(m_1) VT_3=(1,0,0,0,0) CI_3={(1,1,0)}	**lost message m_1**
		diffusion(m_2) VT_3=(1,0,1,0,0) CI_3={(1,1,1)} H(m)={(1,1)} m_2=(3,1,event,(1,1)) send(m_2) CI_3={(1,1,1),(3,10)}	diffusion(m_3) VT_4=(1,0,0,1,0) CI_4={(1,1,1)} H(m)={(1,1)} m_2=(4,1,event,(1,1)) send(m_3) CI_4={(1,1,1),(4,10)}	
lost message m_2	reception(m_2) delivery((m_2) VT_2=(1,0,1,0,0) CI_2={(1,1,1),(3,1,0)}		reception(m_2) delivery(m_2) VT_4=(1,0,1,1,0) CI_4={(4,1,0),(3,1,0)}	reception(m_3) **/*Detection lost message m_1 and update of vectors)*/** VT_5=(1,0,0,0,0) **/*causal devlivery*/** VT_5=(1,0,0,1,0) CI_5={(4,1,0)}
	lost message m_3	reception(m_3) delivery(m_3) VT_3=(1,0,1,1,0) CI_3={(3,1,0),(4,1,0)}		reception(m_2) delivery(m_2) VT_5=(1,0,1,1,0) CI_5={(3,10),(4,1,0)}
Lost message m_3		Diffusion(m_4) VT_3=(1,0,2,1,0) CI_3={(3,1,1),(4,1,1)} H(m)={(3,1),(4,1)} m_4=(3,2,event,{(3,1), (4,1)} send(m_4) CI_3={(3,1,1),(4,1,1), (3,2,0)}		
reception(m_4) **/*Detection lost message m_2 and m_3 and update of vectors)*/** VT_1=(1,0,1,1,0) **/*causal devlivery*/** VT_1=(1,0,2,1,0) CI_1={(1,1,0),(3,2,0)}	reception(m_4) **/*Detection lost message m_3 and update of vectors)*/** VT_2=(1,0,1,1,0) **/*causal devlivery*/** VT_2=(1,0,2,1,0) CI_2={(1,1,1),(3,1,1) (3,2,0)}		reception(m_4) delivery(m_4) VT_4=(1,0,2,1,0) CI_4={(4,1,1),(3,1,1), (3,2,0)}	reception(m_4) delivery(m_4) VT_5=(1,0,2,1,0) CI_5={(3,1,1),(4,1,1)} (3,2,0)}

Table 1.- Running Example related to Figure 3.