Timestamping Algorithms: A Characterization and a Few Properties

Giovanna Melideo[1,2], Marco Mechelli[1], Roberto Baldoni[1], and Alberto Marchetti Spaccamela[1]

[1] Dipartimento di Informatica e Sistemistica, Universitá "La Sapienza" Via Salaria 113, 00198 Roma, Italy
{Melideo, Mechelli, Baldoni, Marchetti}@dis.uniroma1.it
[2] Dipartimento di Matematica ed Applicazioni, Universitá di L'Aquila, Via Vetoio, 67100 L'aquila, Italy

Abstract. Timestamping algorithms are used to capture the causal order or the concurrency of events in asynchronous distributed computations. This paper introduces a formal framework on timestamping algorithms, by characterizing some conditions they have to satisfy in order to capture causality. Under the proposed formal framework we derive a few properties about the size of timestamps and local informations at processes obtained by counting the number of distinct causal pasts which could be observed by an omniscient observer during the evolution of a distributed computation.

1 Introduction

Since the Lamport's seminal paper [5], that formalized the notion of causal dependency between events of an asynchronous distributed computation, a lot of work has been carried out to design distributed algorithms that capture the causal dependencies (or the concurrency) between events during a computation [7]. All these algorithms are based on timestamps associated with events and on the piggybacking of information on messages used to update timestamps. If these timestamps represent an isomorphic embedding of the partial order of the computation, the potential causal precedence or the concurrency between two events can be correctly detected just comparing their timestamps, and we say that the algorithm *characterizes causality.*

In this paper we are interested in introducing a formal framework for timestamping algorithms. At this aim, we consider some operational aspects which allow us to characterize some conditions which any timestamping algorithm has to satisfy in order to characterize causality. Under this framework we prove a bijective correspondence among the set $\mathcal{C}_n$ of causal pasts which could be observed during the execution of all distributed computations of n processes, the set $Im_n(\phi)$ of the timestamps which could be assigned by a timestamping algorithm to events in E and the set $Im_n(\mathcal{I}_\ell)$ of local informations maintained by processes.

A. Bode et al. (Eds.): Euro-Par 2000, LNCS 1900, pp. 609–616, 2000.

An interesting result concerns the reckoning of the causal pasts in $\mathcal{C}_n$, which permits to characterize also the cardinality of the sets $Im_n(\phi)$ and $Im_n(\mathcal{I}_\ell)$. This is done by counting all prefix-closed subsets of E with respect to the causal order relation [5] *which can never be causal pasts of any distributed computation.*

By analyzing the size of non-structured information (i.e. the number of bits) necessary to code elements in $Im_n(\phi)$ and in $Im_n(\mathcal{I}_\ell)$ when the timestamping algorithm has to characterize causality, we obtain a confirmation of the Charron-Bost's result [2]. Algorithms which structure timestamps as vectors of k integers characterize causality only if $k \geq n$ (being n the number of processes). Moreover, we also give a property on the minimum size of control information piggybacked on outgoing messages.

These properties partially answer a question of Schwartz-Mattern [8] about the minimum amount of information that has to be managed by a timestamping algorithm which correctly captures the causality.

The remaining of this paper is structured in 5 sections. Section 2 introduces the computation model. Section 3 presents a formal framework for timestamping algorithms. In Section 4 a few properties are given about the size of timestamps, and the amount of information managed by any timestamping algorithm characterizing causality. Finally, Section 5 relates the results obtained in this paper with timestamping algorithms presented in the literature [5,10,4,1].

2 Computation Model

A distributed computation consists of a finite set of n sequential application processes $\{P_1, P_2, \ldots, P_n\}$ which do not share a common memory and communicate solely by message exchanging, with an unpredictable but finite delay. The execution of each process P_i produces a totally ordered set of events E_i. An event may be either *internal* or it may involve communication (*send* or *receive* event).

Let E be *the disjoint union of totally ordered sets* E_i, i.e. $E = \bigcup_{i=1}^{n} E_i$. This set is structured as a partial order by Lamport's causality relation [5], denoted as $\rightarrow$ and defined as follows:

Definition 1. *The causality relation* $\rightarrow \subseteq E \times E$ *is the smallest relation satisfying the following conditions:* $e \rightarrow e'$ *if one of these conditions holds:*
(1) e *and* e' *are events in the same process and* e *precedes* e'
(2) $\exists\ m\ :\ \ e = send(m)\ \wedge\ e' = receive(m)$
(3) $\exists\ e''\ :\ \ e \rightarrow e''\ \wedge e'' \rightarrow e'$.

Two events e and e' are concurrent if $\neg(e \rightarrow e')$ and $\neg(e' \rightarrow e)$.

The partial order $\widehat{E} = (E, \rightarrow)$ constitutes a formal model of the distributed computation it is associated with. Namely:

Definition 2. *A relation* $\rightarrow \subseteq E \times E$ *is a causality relation on* E *if [2]:*
(1) $(E, \rightarrow)$ *has no cycles, and*
(2) $\forall e \in E,\ |\{(e, e') \in \rightarrow |\ e' \in E\} \cup \{(e', e) \in \rightarrow |\ e' \in E\}| \leq 1$
(i.e. for every receipt of a message m*, there is a single sending of* m*).*

According to this model, we denote as $e_{i,j} \in E$ a generic j-th event produced by the process P_i, whose type (internal/send/receive) is defined by the specific causality relation $\rightarrow$ considered on these events[1].

Moreover, we model all distributed computations of n processes as the set $\widehat{\mathcal{E}}_n = \{(E, \rightarrow) \mid \rightarrow \in \mathcal{R}_\rightarrow\}$, where $\mathcal{R}_\rightarrow \subseteq 2^{E \times E}$ denotes the set of all causality relations on E.

For a given computation $\widehat{E} \in \widehat{\mathcal{E}}_n$, the *causal past* of e in $\widehat{E}$ is the prefix-closed set of E under the causal order $\uparrow (e, \widehat{E}) = \{e' \in E \mid e' \rightarrow e\} \cup \{e\}$. Each causal past $\uparrow (e, \widehat{E}) \subseteq E$ can be decomposed in n disjoint subsets $\uparrow_1 (e, \widehat{E}), \uparrow_2 (e, \widehat{E}), \ldots, \uparrow_n (e, \widehat{E})$, where $\uparrow_i (e, \widehat{E}) = \uparrow (e, \widehat{E}) \cap E_i$.

Following Schwartz-Mattern [8], $\forall\ \widehat{E} \in \widehat{\mathcal{E}}_n$, $(\{\uparrow (e, \widehat{E}) \mid e \in E\}, \subseteq)$ is an isomorphic embedding of $(E, \rightarrow)$. In fact, different causal pasts in the same computation correspond to different events, and

$$\forall\ \widehat{E} \in \widehat{\mathcal{E}}_n,\ \forall e, e' \in \widehat{E}\ (e \neq e'), \quad e \rightarrow e' \Leftrightarrow \uparrow (e, \widehat{E}) \subset \uparrow (e', \widehat{E}). \tag{1}$$

We denote as $\mathcal{C}_n = \bigcup_{\widehat{E} \in \widehat{\mathcal{E}}_n} \{\uparrow (e, \widehat{E}) \mid e \in E\}$ the set of causal pasts which could be observed during the execution of all computations of n processes.

3 A Characterization of Timestamping Algorithms

Techniques to detect causality relations or concurrency between events are based on timestamps of events produced by the execution of a timestamping algorithm, which assigns "on-the-fly", that is during the evolution of the computation $\widehat{E}$ and without knowing its future, to each event e a value $\phi(e, \widehat{E})$ of a suitable partially ordered set $(D, <)$.

A *timestamping algorithm* $\mathcal{A}$ is usually characterized by a partially ordered set $(D, <)$ called *timestamps domain*, a *timestamping function* ϕ which establishes a correspondence between events of a computation and timestamps in D, and a set of rules implementing the algorithm which decide both local information at processes and control information piggybacked by messages.

The aim is to assign values in D to events so that for each $\widehat{E} \in \widehat{\mathcal{E}}_n$, the suborder $(\{\phi(e, \widehat{E}) \mid e \in E\}, <)$ of the timestamps assigned to events is an isomorphic embedding of $(E, \rightarrow)$.

This is usually formalized [2,3,8] by requiring the function ϕ *characterizes causality*, i.e. ϕ is injective and $\forall \widehat{E} \in \widehat{\mathcal{E}}_n, \forall e, e' \in \widehat{E},\ e \rightarrow e' \Leftrightarrow \phi(e, \widehat{E}) < \phi(e', \widehat{E})$.

We denote as $Im_n(\phi) = \bigcup_{\widehat{E} \in \widehat{\mathcal{E}}_n} \{\phi(e, \widehat{E}) \mid e \in E\} \subseteq D$ the set of timestamps which could be assigned by any timestamping algorithm to events during the execution of all computations of n processes.

[1] When referring to generic events we drop subscripts and we use the following simple notation e, e' and e''.

A bijective correspondence between $Im_n(\phi)$ and $Im_n(\mathcal{I}_\ell)$. A deterministic timestamping algorithm assigns a timestamp to an event e only basing on the current local control information at the process producing e. So, it will assign the same timestamp to two events which have the same local information at process, even though they belong to two distinct distributed computations.

We denote as $\mathcal{I}_\ell(e, \widehat{E})$ the local control information associated with e by the timestamping algorithm during the execution of the computation[2]. Namely, $\mathcal{I}_\ell$ is a function mapping events to values in an ordered set $(L, \prec)$, called *local domain.* Then, we can say that

$$\forall\ \widehat{E}_1, \widehat{E}_2 \in \widehat{\mathcal{E}}_n,\ \mathcal{I}_\ell(e, \widehat{E}_1) = \mathcal{I}_\ell(e', \widehat{E}_2) \Rightarrow \phi(e, \widehat{E}_1) = \phi(e', \widehat{E}_2). \tag{2}$$

Let $Im_n(\mathcal{I}_\ell) = \bigcup_{\widehat{E} \in \widehat{\mathcal{E}}_n} \{\mathcal{I}_\ell(e, \widehat{E}) \mid e \in E\}$ be the set of local informations which could be associated with events by any timestamping algorithm, during the execution of all computations of n processes. The following proposition shows that any timestamping algorithm which characterizes causality is characterized by a bijective correspondence between $Im_n(\mathcal{I}_\ell)$ and $Im_n(\phi)$. In fact, it proves that if the timestamping function characterizes causality then the converse of (2) is also true.

Proposition 1. *If a timestamping function characterizes causality, then*

$$\forall\ \widehat{E}_1, \widehat{E}_2 \in \widehat{\mathcal{E}}_n,\ \mathcal{I}_\ell(e, \widehat{E}_1) = \mathcal{I}_\ell(e', \widehat{E}_2) \Leftrightarrow \phi(e, \widehat{E}_1) = \phi(e', \widehat{E}_2). \tag{3}$$

Proof. Sufficiency is given by equation (2). To prove necessity, let $e \in \widehat{E}_1,\ e' \in \widehat{E}_2$ be two events with the same timestamp (i.e. $\phi(e, \widehat{E}_1) = \phi(e', \widehat{E}_2)$) and $\mathcal{I}(e, \widehat{E}_1) \neq \mathcal{I}(e', \widehat{E}_2)$. Since a timestamping algorithm cannot predict the progress of the computation, there could exist an event $e'' \in \widehat{E}_1$ such that $\mathcal{I}(e'', \widehat{E}_1) = \mathcal{I}(e', \widehat{E}_2)$. In this case, condition (2) implies that $\phi(e'', \widehat{E}_1) = \phi(e', \widehat{E}_2)$, that is the algorithm must assign to event e'' the same timestamp as e'. By hypothesis $\phi(e, \widehat{E}_1) = \phi(e', \widehat{E}_2)$, so $\phi(e, \widehat{E}_1) = \phi(e'', \widehat{E}_1)$ holds, that is in the same computation two events have the same timestamp. As ϕ characterizes causality, $\phi(e, \widehat{E})$ and $\phi(e'', \widehat{E})$ must be distinct.

A bijective correspondence between $\mathcal{C}_n$ and $Im_n(\phi)$. If ϕ characterizes causality, the condition (1) implies that $\forall \widehat{E} \in \widehat{\mathcal{E}}_n$, $(\{\phi(e, \widehat{E}) \mid e \in E\}, <)$ is an isomorphic embedding of $(\{\uparrow (e, \widehat{E}) \mid e \in E\}, \subseteq)$, i.e.

$$\forall \widehat{E} \in \widehat{\mathcal{E}}_n,\ \forall e, e' \in \widehat{E}, (e \neq e'),\ \uparrow (e, \widehat{E}) \subseteq \uparrow (e', \widehat{E}) \Leftrightarrow \phi(e, \widehat{E}) < \phi(e', \widehat{E}). \tag{4}$$

We consider an omniscient observer whose role is to instantaneously detect if a pair of events is causally related or concurrent only by comparing their timestamps.

[2] We suppose there is no redundant local information at processes, i.e. the local information is minimal.

The condition (1) implies *the observer must have perfect knowledge of all causal pasts at any time*, so it can be argued that the timestamps known by the observer $(Im_n(\phi), <)$ have to form an isomorphic embedding of $(\mathcal{C}_n, \subseteq)$. This implies the *decoding function* $\varphi : Im_n(\phi) \rightarrow \mathcal{C}_n$, which characterizes the algorithm executing by the observer, is *bijective* and satisfies the following condition: $\forall d_1, d_2 \in Im_n(\phi),\ d_1 < d_2 \Leftrightarrow \varphi(d_1) \subseteq \varphi(d_2)$.

Previous condition directly implies (4). Moreover, since φ is bijective and $\phi = \varphi^{-1} \circ \uparrow$, we can assert that causal pasts and timestamping functions characterizing causality are also related as follows:

$$\uparrow (e, \widehat{E}_1) = \uparrow (e', \widehat{E}_2) \Leftrightarrow \phi(e, \widehat{E}_1) = \phi(e', \widehat{E}_2). \tag{5}$$

The operational aspects of timestamping algorithms analyzed in the previous paragraphs allow us to argue that both $Im_n(\phi) \subseteq D$ and $Im_n(\mathcal{I}_\ell) \subseteq L$ are actually a coding of the set $\mathcal{C}_n$. Then, we can characterize a timestamping algorithm as a sequence $\mathcal{A}(\overline{D}, \overline{L}, \chi_D, \chi_L)$ where:

- $\overline{D} = (D, <)$ is a partial order called *timestamps domain*;
- $\overline{L} = (L, \prec)$ is a partial order called *local domain*;
- $\chi_D : \mathcal{C}_n \rightarrow D$ is a mapping from causal pasts to timestamps;
- $\chi_L : \mathcal{C}_n \rightarrow L$ is a mapping from causal pasts to local informations.

Definition 3. *A timestamping algorithm* $\mathcal{A}(\overline{D}, \overline{L}, \chi_D, \chi_L)$ characterizes causality *if (i)* χ_D *and* χ_L *are both injective functions, and (ii) the function* $\phi = \chi_D \circ \uparrow$ *characterizes causality (i.e.* $\mathcal{A}$ *characterizes causality if* ϕ *characterizes causality and it timestamps events according to (3) and (5)).*

An Example of Timestamping Algorithm: Vector Clocks [3,6]. The Vector Clocks algorithm codifies causal pasts as integer vectors of size n. Let VC_i be the vector clock endowed by the process P_i. $VC_i[j]$ represents the number of events on P_j in the causal past known by P_i. In this case:

(1) $\overline{D} \equiv \overline{L} \equiv (I\!N^n, \leq)$, where $\forall\ V, V' \in I\!N^n,\ V \leq V'$ iff $\forall i,\ V[i] \leq V'[i]$;
(2) $\chi_D \equiv \chi_L : \mathcal{C}_n \rightarrow I\!N^n$ is defined as: $\forall i,\ \forall S \in \mathcal{C}_n,\ \chi_D(S)[i] = |S \cap E_i|$.

4 Causal Pasts of a Set of Events *E*

In this section we provide some interesting properties on the set of causal pasts $\mathcal{C}_n$. Moreover, being $|Im_n(\phi)| = |Im_n(\mathcal{I}_\ell)| = |\mathcal{C}_n|$, we are interested in the reckoning of elements in $\mathcal{C}_n$, obtained as a corollary of Propositions 2 and 3.

The following proposition gives necessary conditions so that a prefix-closed subset of events $S \subseteq E$ could be a causal past. We recall that each causal past $S \in \mathcal{C}_n$ can be decomposed in n subsets $S_1, S_2, \ldots, S_n$, where $S_i = S \cap E_i$.

Proposition 2. *Let* $S \subseteq E$ *be a prefix-closed subset of events generated by* n *processes.* S *is a causal past* ($S \in \mathcal{C}_n$) *only if* $S \neq \emptyset$ *and when the number* k *of nonempty subsets in its decomposition is at least* 3, $|S| \geq 2(k-1)$.

Proof. The first claim easily follows by the definition of causal past, because at least e belongs to $\uparrow (e, \widehat{E})$, so $S \neq \emptyset$. If k processes have events in the causal past S, then at least $k-1$ processes have to send messages in order to establish a dependency. Each of $k-1$ messages contributes 2 events to S. Hence, $2(k-1)$ is the minimum number of events in S when $k \geq 3$.

The previous proposition proved that if k subsets, with $3 \leq k \leq n$, are nonempty in the decomposition of a prefix-closed subset S and $|S| \leq 2k-3$, then $S \notin \mathcal{C}_n$. In the following proposition we count the number of these sets, in order to obtain a precise reckoning of $\mathcal{C}_n$.

Proposition 3. *The number of prefix-closed subsets $S \subseteq E$ of size at most $2k-3$ which can be decomposed in $k \geq 3$ nonempty subsets is:*

$$\sum_{k=3}^{n} \binom{n}{k} \binom{2k-3}{k}. \tag{6}$$

Proof. By applying basic mathematical enumeration results, since (i) k nonempty subsets can be on any of n processes and (ii) the number of prefix-closed sets S of size h which can be decomposed in k *nonempty* subsets is $\binom{h-1}{h-k}$, we have that the number required is: $\sum_{k=3}^{n} \binom{n}{k} \sum_{h=k}^{2k-3} \binom{h-1}{h-k} = \sum_{k=3}^{n} \binom{n}{k} \sum_{h=0}^{k-3} \binom{h+k-1}{h}$.

The value $\binom{h+k-1}{h}$, denoted as $N(h,k)$, represents the number of prefix-closed subsets of size h which can be decomposed in *at most* k subsets. It can be easily proved that $N(h,k) = \sum_{i=0}^{h} N(i, k-1)$, so the thesis (6) follows by $\sum_{h=0}^{k-3} \binom{h+k-1}{h} = \sum_{h=0}^{k-3} N(h,k) = N(k-3, k+1) = \binom{2k-3}{k}$.

For simplicity's sake and wlog we assume processes generate m event each.

Corollary 1. *If n processes generate m events each, then*

$$|\mathcal{C}_n| = (m+1)^n - 1 - \sum_{k=3}^{n} \binom{n}{k} \binom{2k-3}{k}.$$

Proof. If each process generates m events, we have $(m+1)^n$ different prefix-closed subsets of events. The thesis follows by considering that $\emptyset \notin \mathcal{C}_n$ and there are $\sum_{k=3}^{n} \binom{n}{k} \binom{2k-3}{k}$ prefix-closed subsets which cannot be causal pasts (Eq. 6).

4.1 Properties

The Corollary 1 and the Properties 1 and 2 directly imply that:

Property 1. A timestamping algorithm characterizes causality only if $|Im_n(\phi)| = |Im_n(\mathcal{I}_\ell)| = (m+1)^n - 1 - \sum_{k=3}^{n} \binom{n}{k} \binom{2k-3}{k}$.

As a consequence, the coding of each element in $Im_n(\phi)$ and $Im_n(\mathcal{I}_\ell)$ requires at least $\lceil \log_2 |\mathcal{C}_n| \rceil = \lceil \log_2 ((m+1)^n - 1 - \sum_{k=3}^{n} \binom{n}{k} \binom{2k-3}{k}) \rceil$ bits. Regarding local information at processes, from an operational point of view, the empty set

is usually used in the initial step, so in practice it is necessary to locally use at least $\lceil \log_2 (|Im_n(\phi)| + 1) \rceil = \lceil \log_2 ((m+1)^n - \sum_{k=3}^{n} \binom{n}{k} \binom{2k-3}{k}) \rceil$ bits.

Property 2 gives the necessary amount of information piggybacked on messages, when the timestamping algorithm characterizes causality locally maintaining only minimal control informations, that is codings of causal pasts.

Let $\mathcal{I}_p(send(m), \widehat{E})$ be the control information piggybacked upon message m, and $Im_n(\mathcal{I}_p) = \bigcup_{\widehat{E} \in \widehat{\mathcal{E}}_n} \{\mathcal{I}_p(e, \widehat{E}) \mid e \in E\}$ be the set of control informations which could be piggybacked upon messages during the execution of all distributed computations on n processes (if e is not a send event we assume $\mathcal{I}_p(e, \widehat{E}) = \emptyset$).

Property 2. A timestamping algorithm characterizes causality only if $|Im_n(\mathcal{I}_p)| = (m+1)^{n-1} - 1 - \sum_{k=3}^{n-1} \binom{n-1}{k} \binom{2k-3}{k}$.

Proof. Let $e_{u,h}$ be a send event and $e_{i,j}$ the corresponding receive event in any computation $\widehat{E}$. By definition, $\uparrow (e_{i,j}, \widehat{E}) = \uparrow (e_{i,j-1}, \widehat{E}) \cup \uparrow (e_{u,h}, \widehat{E}) \cup \{e_{i,j}\}$. If we denote $S_k = \uparrow_k (e_{i,j-1}, \widehat{E})) \cup \uparrow_k (e_{u,h}, \widehat{E})$, we have $\uparrow (e_{i,j}, \widehat{E}) = \uparrow_i (e_{i,j}, \widehat{E}) \cup (\bigcup_{k \neq i} S_k)$. Then, distinct values of $\uparrow (e_{i,j}, \widehat{E})$ are associated to different values of $\bigcup_{k \neq i} S_k$, which are as many as all possible causal pasts which involve events in $n-1$ processes. Consequently their number is $(m+1)^{n-1} - 1 - \sum_{k=3}^{n-1} \binom{n-1}{k} \binom{2k-3}{k}$.

As a consequence the coding of each element in $Im_n(\mathcal{I}_p)$ requires at least $\lceil \log_2 ((m+1)^{n-1} - 1 - \sum_{k=3}^{n-1} \binom{n-1}{k} \binom{2k-3}{k}) \rceil$ bits.

A remark on the Vector Clock algorithm. If n processes generate m events each, $D = L = \{0, \ldots, m\}^n$, so $|D| = |L| = (m+1)^n$. By Proposition 2, D and L are redundant. In fact, $D, L \supset Im_n(\phi) = Im_n(\mathcal{I}_\ell) = \{V \in \{0, \ldots, m\}^n \mid k_V \geq 3 \Rightarrow \sum_{i=1}^{n} V[i] \geq 2(k_V - 1)\}$, where k_V denotes the number of indices i such that $V[i] \neq 0$, implying $|D|, |L| > \mathcal{C}_n$. Namely, D and L include vectors (as [1,1,1]) which can never be associated with events, codifying prefix-closed subsets which are not causal pasts.

To codify all elements in D and L it is necessary to use at least $n\lceil \log_2(m+1) \rceil$ bits, that is more than necessary information. However, from an operational point of view, a coding that excludes non-potential causal pasts seems to be not practicable. As a consequence, a timestamping algorithm based on vector clocks provides the closest coding to the minimal quantity of information required.

5 Related Work

Previous properties give a theoretical confirmation to the fact that some timestamping algorithms such as *scalar clock* [5], *plausible clocks* [10] and *direct dependency vectors* [4] are not able to characterize causality on-the-fly[3].

[3] A deep discussion about these timestamping algorithms is out of the scope of this paper. Nice surveys can be found in [7,8].

Plausible clocks maintain locally at each process a vector of $k < n$ entries, that is less than the quantity required by property 1. The *scalar clocks* are a particular case of plausible clocks when considering $k = 1$.

A timestamping algorithm based on *direct dependency* tracking meets the requirement of property 1 as each process maintains locally a vector of integers of size n. However, each message piggybacks only one integer, that is the index of the send event of the sender process, so by Property 2, it is not appropriate to characterize causality. On the other hand, it is well-known that the timestamping algorithm based on *direct dependency* can off-line (i.e., with some additional computation) reconstruct all causality relations between events [8]. This give rise to an interesting remark: *if a timestamping algorithm satisfies Property 1 but not Property 2, it has the necessary information for characterizing the causality relation but it needs some extra (off-line) computation.*

Previous observation is the baseline of the *k-dependency vector* algorithm introduced in [1], where, given an integer $k \leq n$, each process piggybacks a subset of size k of the local vector, including the current index of the send event of the sender process (as in direct dependency algorithm) and other $k - 1$. The choice of the other $k - 1$ values is left to a scheduling policy of the algorithm.

Acknowledgments
We acknowledge the support of the EU ESPRIT LTR Project "ALCOM-IT" under contract n. 20244.

References

1. R. Baldoni, A., M. Mechelli and G. Melideo. A General Scheme for Dependency Tracking in Distributed Computations, *Technical Report n. 17.99*, Dipartimento di Informatica e Sistemistica, Roma, 1999.
2. B. Charron-Bost. Concerning the size of logical clocks in distributed systems, *Information Processing Letters*, 39, 11–16, 1991.
3. C. Fidge. Timestamps in message passing system that preserve the partial ordering, *Proc. 11th Australian Computer Science Conf.*, 55–66, 1988.
4. J. Fowler and W. Zwaenepoel. Causal distributed breakpoints, *Proc. of 10th IEEE Int'l. Conf. on Distributed Computing Systems*, 134–141, 1990.
5. L. Lamport, *Time, clocks, and the ordering of events in a distributed system*, *Comm. ACM*, **21**7, 558-564, 1978.
6. F. Mattern. Virtual time and global states of distributed systems, M. Cosnard and P. Quinton eds. *Parallel and Distributed Algorithms* 215-226, 1988.
7. M. Raynal and M. Singhal, Logical Time: Capturing Causality in Distributed Systems, *IEEE Computer*, 29(2):49-57, 1996.
8. R. Schwarz and F. Mattern, *Detecting causal relationships in distributed computations: in search of the holy grail*, Distributed Computing 7(3), 149–174, 1994.
9. M. Singhal and A. Kshemkalyani, An Efficient Implementation of Vector Clocks. *Information Processing Letters*, 43:47-52, 1992.
10. F. J. Torres-Rojas and M. Ahamad, Plausible Clocks: constant size logical clocks for distributed systems, *Proceedings of the International Workshop on Distributed Algorithms*, 71–88, 1996.