Snapshots et Détection de Propriétés Stables dans les Systèmes Distribués Anonymes

Jérémie Chalopin[1], Yves Métivier[2], Thomas Morsellino[2]

[1]*LIF, CNRS, Université Aix-Marseille, 39, rue Joliot Curie 13453 Marseille Cedex 13, France*
[2]*LaBRI UMR 5800, CNRS, Université de Bordeaux, 351, cours de la Libération 33405 Talence, France*

Nous étudions les problèmes du calcul d'état global (ou snapshot) et, plus généralement, de la détection de propriétés stables dans les systèmes totalement distribués et anonymes. Nous considérons le modèle classique à passage de messages dans lequel, pour une étape de calcul, chaque élément du système peut changer son état, envoyer ou recevoir un message à travers des liens de communication. La plupart des algorithmes existants pour résoudre le problème du calcul d'état global supposent que les éléments du système ont des identifiants uniques ou qu'il existe un unique noeud initiateur. Ce travail concerne le calcul d'état global dans les systèmes anonymes et plus généralement quelles sont les propriétés stables d'un système distribué qui peuvent être détectées anonymement par l'utilisation de snapshots locaux tout en autorisant des initiateurs multiples et en ne connaissant qu'une borne supérieure sur le diamètre du réseau.

Keywords: Calcul distribué, État global, Propriété stable.

1 Introduction

Many complex distributed systems require a large number of servers and machines interconnected on either local or remote networks. Such distributed applications aim at making several processes collaborate to the execution of a same task. Problems are raised concerning concurrent access to resources, critical failure detection, or even process communication strategy. The design and the validation of distributed applications depend on the analysis and understanding of underlying algorithms that must be proved, implemented and debugged. The debugging of such algorithms is precisely addressed by the snapshot computation problem.

The Problem. Given a distributed system, the aim of a snapshot algorithm is the computation of a global state, an instantaneous photography of the whole system. As is explained by Tel [Tel00] (p. 335-336), the construction of snapshots is mainly motivated by the detection of stable properties of the distributed system (properties which remain true as soon as they are verified) or if the system must be restarted (due to a failure of a component) then it may be restarted from the last known snapshot (and not from the initial configuration). Besides, it may be useful in debugging distributed algorithms. A consistent snapshot is a global state of the distributed system or a global state that the system could have reached. Since the seminal paper of Chandy and Lamport [CL85] which presents an algorithm to compute a consistent snapshot, many papers gave snapshot algorithms for different models of distributed systems. They assume that processes have unique identifiers and/or that there is exactly one initiator. Many papers [SS94, KS08] give also specific algorithms to detect some specific properties like termination or deadlock.

Besides, Guerraoui and Ruppert in [GR05], considering that a vast majority of papers on distributed computing assume that processes have unique identifiers, ask the following question : What if processes do not have unique identifiers or do not wish to divulge them for reasons of privacy ? This work addresses this question in the context of snapshots computations and by considering stable properties of a distributed system that can be detected anonymously and admiting several initiators.

The Model. Our model is the usual asynchronous message passing model [Tel00]. A network is represented by a simple connected graph $G = (V(G), E(G))$ where vertices correspond to processes and edges to direct communication links. The state of each process p is represented by a label $\lambda(v)$ associated to the corresponding vertex $v \in V(G)$; we denote by $\mathbf{G} = (G, \lambda)$ such a labelled graph. We assume the network to

be anonymous : the identities of processors are not necessarily unique or, for privacy and security reasons, processes do not share their identities during computation steps. We assume that each process knows from which channel it receives or it sends a message, i.e., for each $u \in V(G)$ there exists a bijection δ_u between the neighbors of u in G and $[1, \deg_G(u)]$, called *port-numbering*. We will denote by δ the set of functions $\{\delta_u \mid u \in V(G)\}$. Let p be a process. Let q be a neighbor of p, i.e., pq is a channel. The state associated to p is denoted by $\texttt{state}(p)$. The multiset of messages in transit associated to pq is denoted by M_{pq} (initially M_{pq} is empty). The *local snapshot*, with respect to the process p, is defined by $\texttt{state}(p)$ and the set of $\{M_{qp} \mid$ q is a neighbor of p$\}$. We consider asynchronous systems, i.e., no global time is available and each computation may take an unpredictable (but finite) amount of time. Note that we consider only reliable systems : there are no message loss or duplication. We also assume that the channels are FIFO.

Our Contribution. We consider anonymous networks in which several processes can be initiators of computations. Thus, no process of the network can compute a snapshot (it is a direct consequence of Theorem 5.5 in [Ang80]). Furthermore we assume that each process knows an upper bound of the diameter β of the network. First we give a very simple algorithm based on the composition of an algorithm by Szymanski, Shy, and Prywes [SSP85] with the Chandy-Lamport algorithm which enables each process to detect an instant where all processes have obtained their local snapshot and to associate the same number to all local snapshots. By this way we obtain an original application to checkpoint and rollback recovery.

Then we prove that stable properties can be anonymously detected by proving we can compute a snapshot up to covering. Let (G,λ) be a labelled graph with the port-numbering δ. We will denote by $(\mathrm{Dir}(\mathbf{G}),\delta)$ the symmetric labelled digraph $(\mathrm{Dir}(G),(\lambda,\delta),s_A,t_A)$ constructed in the following way. The vertices of $\mathrm{Dir}(G)$ are the vertices of G and they have the same labels in $\mathbf{G}$ and in $\mathrm{Dir}(\mathbf{G})$. Each edge $\{u,v\}$ of G is replaced in $(\mathrm{Dir}(\mathbf{G}),\delta)$ by two arcs $a_{(u,v)}, a_{(v,u)} \in A(\mathrm{Dir}(G))$ such that $s(a_{(u,v)}) = t(a_{(v,u)}) = u$, $t(a_{(u,v)}) = s(a_{(v,u)}) = v$ (where s and t are two maps that assign to each arc two elements of $V(G)$: a source and a target), $\delta(a_{(u,v)}) = (\delta_u(v), \delta_v(u))$ and $\delta(a_{(v,u)}) = (\delta_v(u), \delta_u(v))$. A digraph D is a covering of another digraph D' if there is a surjective homomorphism φ from D to D' which is locally bijective on arcs. We give a fully distributed algorithm with termination detection which, given (G,λ) a labelled graph with the port-numbering δ, enables each process to compute a labelled digraph $\mathbf{D}$ such that $(Dir(\mathbf{G}),\delta)$ is a symmetric covering of $\mathbf{D}$. We show that stable properties such as termination, deadlock, garbage or loss of tokens detected knowing (G,λ) can be detected knowing $\mathbf{D}$, called a *weak snapshot.*

Related Works. Many notions and algorithms concerning snapshots, stable properties, checkpointing and rollback recovery can be found in [KS08]. From a theoretical point of view, it is simple to know whether the global state of a distributed system satisfies a stable property. A distinguished process starts the Chandy-Lamport algorithm, then it collects states of processes and states of channels, it computes a map of the network and finally it tests whether the labelled network satisfies the given property. To collect or to analyze local snapshots, different assumptions may be done (see [KRS95]) : processes have unique identifiers, there is exactly one initiator or one collector process. Some results have been obtained for the computation of snapshots in asynchronous shared-memory systems that are anonymous (see Section 5 of [GR05] for a survey). This paper also presents results concerning consensus and timestamping.

Many notions and algorithms concerning snapshots and global predicates In any case, it is assumed that processes have identifiers and/or that there is exactly one initiator. No such possibilities exist under our assumptions. As far as we know, our method is an emerging solution for global snapshots computation and stable properties detection over anonymous networks where the only knowledge is an upper bound on the diameter of the network.

2 Snapshot Algorithms for Anonymous Networks

The aim of the snapshot algorithm [CL85] is to construct a system configuration defined by the state of each process and the state of each channel. Once the computation of local snapshots is completed, the knowledge of the snapshot is fully distributed over the system. In this section we show how to exploit this distributed knowledge in the context of anonymous networks with no distinguished process and no particular topology.

2.1 Checkpoint Algorithm : Termination Detection of the Snapshot Algorithm

We use a combination of the Chandy-Lamport algorithm with an adaptation of the SSP algorithm. This combination of algorithms enables each process to detect an instant where all processes have completed the computation of their local snapshot.

Let G be a graph, to each process p is associated a predicate $P(p)$ (initially $P(p)$ is false). Once $P(p)$ becomes *true*, it remains *true* thereafter. Each process p knows when it has finished its local snapshot computation. We consider the termination detection of a local snapshot computation as such a predicate $P(p)$. A process p is also endowed with two other variables : $a(p) \in \mathbb{Z}$ is a counter (initially $a(p) = -1$), $a(p)$ represents the distance up to which all processes have satisfied the predicate ($P(p) = true$) (i.e., the distance up to which all processes have completed the computation of the local snapshot) ; $A(p) \in \mathcal{P}_{\text{fin}}(\mathbb{N} \times \mathbb{Z})$ [†] encodes the information p has about the values of $a(q)$ for each neighbor q (initially, $A(p) = \{(i,-1) | i \in [1, deg_G(p)]\}$). Transformations of the value of $a(p)$ are defined by the following description.

Outline of the algorithm. First, if $P(p) = false$ then $a(p) = -1$. Otherwise, if a process p has completed the computation of its local snapshot ($P(p) = true$) then it changes the value of $a(p)$ to 0 and it informs its neighbors. When a process p receives a value $a(q)$ for some neighbor q via the port i then it substitutes the new value $(i, a(q))$ to the old value (i,x) in $A(p)$. Finally, p computes the new value $a(p) = 1 + Min\{x | (i,x) \in A(p)\}$. A process p knows that each process has completed the computation of its local snapshot as soon as $a(p) \geq \beta$.

When the local snapshot is computed for each process p, it enables p to restart a system if there is a failure. As explained in [KS08] p. 456, the saved state is called a checkpoint, and the procedure of restarting from a previously checkpointed state is called rollback recovery.

2.2 Computing Anonymously a Weak Snapshot

The notion of coverings is fundamental in this work ; a labelled digraph $\mathbf{D}$ is a *covering* of a labelled digraph $\mathbf{D}'$ via φ if φ is a homomorphism from $\mathbf{D}$ to $\mathbf{D}'$ such that for each arc $a' \in A(D')$ and for each vertex $v \in \varphi^{-1}(t(a'))$ (resp. $v \in \varphi^{-1}(s(a'))$, there exists a unique arc $a \in A(D)$ such that $t(a) = v$ (resp. $s(a) = v$) and $\varphi(a) = a'$.

Given a simple connected labelled graph $\mathbf{G} = (G, \lambda)$ with a port-numbering δ which defines a snapshot of a network G. Let $\mathbf{D} = (\text{Dir}(\mathbf{G}), \delta)$ be the corresponding labelled digraph $(\text{Dir}(G), (\lambda, \delta))$. Let $\mathbf{D}'$ be a labelled digraph such that $\mathbf{D} = (\text{Dir}(\mathbf{G}), \delta)$ is a covering of $\mathbf{D}'$. The labelled digraph $\mathbf{D}'$ is called a *weak snapshot* of G.

Proposition 1. *Let G be a distributed system. From any weak snapshot $\mathbf{D}$ of G, one can detect deadlock and termination and one can perform garbage collection. Furthermore, if the processes know the size of G then loss of tokens can also be detected from a weak snapshot.*

Hence, knowing an upper bound of the diameter β of an anonymous network $\mathbf{D}_1$, there exists a fully distributed algorithm (it may admit several initiators) with termination detection, which computes $\mathbf{D}_2$ such that $\mathbf{D}_1$ is a covering of $\mathbf{D}_2$. In anonymous networks and considering algorithms with multiple initiators, we cannot compute a snapshot nevertheless from Proposition 1, we can solve stable properties detection. Our algorithm is as follows :

1. at least one process initiates the Chandy-Lamport algorithm [CL85] ;
2. each process detects an instant where the computation of all local snapshots is completed (see Section 2.1) ;
3. at least one process initiates the computation of a weak snapshot (Algorithm 1) ;
4. each process detects an instant where the computation of the weak snapshot is completed and decides about the stable property.

Now, we present the outline of the fully distributed algorithm which computes anonymously a weak snapshot with termination detection (Step 2). It is an adaptation of the work done by Mazurkiewicz [Maz97].

---

†. For any set S, $\mathcal{P}_{\text{fin}}(S)$ denotes the set of finite subsets of S.

Outline of the algorithm. In Algorithm 1, for each vertex $v \in V(\mathbf{G})$, $\lambda(v)$ corresponds to the label of v obtained after the termination of the Chandy-Lamport algorithm. During the execution, each vertex v attempts to get an identity which is a number between 1 and $|V(G)|$. Once a vertex v has chosen a number $n(v)$, it sends it to each neighbor u with the port-number $\delta_v(u)$. When a vertex u receives a message from one neighbor v, it stores the number $n(v)$ with the port-numbers $\delta_u(v)$ and $\delta_v(u)$. From all information it has gathered from its neighbors, each vertex can construct its *local view* (which is the set of numbers of its neighbors associated with the corresponding port-numbers). Then, a vertex broadcasts its number, its label and its mailbox (which contains a set of *local views*). If a vertex u discovers the existence of another vertex v with the same number then it should decide if it changes its identity. To this end it compares its local view with the local view of v. If the label of u or the local view of u is "weaker", then u picks another number — its new temporary identity — and broadcasts it again with its local view. At the end of the computation, each vertex has computed a graph $(\mathbf{D}, \delta')$ such that $(Dir(\mathbf{G}), \delta)$ is a symmetric covering of $(\mathbf{D}, \delta')$.

Algorithm 1: Algorithm $\mathcal{M}_{W-S}$.

I : $\{n(v_0) = 0$ and no message has arrived at $v_0\}$
begin
  $n(v_0) := 1$;
  $M(v_0) := \{(n(v_0), \lambda(v_0), \emptyset)\}$;
  for $i := 1$ **to** $\deg(v_0)$ **do**
    send $< (n(v_0), \lambda(v_0), M(v_0), a(v_0)), i >$ through port i ;

R : $\{$A message $< (n_1, \ell_1, M_1, a_1), i_1 >$ has arrived at v_0 through port $j_1\}$
begin
  $M_{old} := M(v_0)$;
  $a_{old} := a(v_0)$;
  $M(v_0) := M(v_0) \cup M_1$;
  if $n(v_0) = 0$ *or* $\exists (n(v_0), \ell', N') \in M(v_0)$ *such that* $(\lambda(v_0), N(v_0)) \prec (\ell', N')$ **then**
    $n(v_0) := 1 + \max\{n' \mid \exists (n', \ell', N') \in M(v_0)\}$;
  $N(v_0) := N(v_0) \setminus \{(n', \ell', i_1, j_1) \mid \exists (n', \ell', i_1, j_1) \in N(v_0)\} \cup \{(n_1, \ell_1, i_1, j_1)\}$;
  $M(v_0) := M(v_0) \cup \{(n(v_0), \lambda(v_0), N(v_0))\}$;
  if $M(v_0) \neq M_{old}$ **then**
    $a(v_0) := -1$;
    $A(v_0) := \{(i', -1) \mid 1 \leq i' \leq deg(v_0)\}$;
  if $M(v_0) = M_1$ **then**
    $A(v_0) := A(v_0) \setminus \{(j_1, a') \mid \exists (j_1, a') \in A(v_0)\} \cup \{(j_1, a_1)\}$;
  if $\forall (i', a') \in A(v_0), a(v_0) \leq a'$ *and* $a(v_0) \leq \beta$ **then** $a(v_0) := a(v_0) + 1$;
  if $M(v_0) \neq M_{old}$ *or* $a(v_0) \neq a_{old}$ **then**
    for $i := k$ **to** $\deg(v_0)$ **do**
      send $< (n(v_0), \lambda(v_0), M(v_0), a(v_0)), k >$ through port k ;

Conclusion. We adressed the problem of stable properties which can be detected in a fully distributed system assuming that processes are anonymous and only know an upper bound of the diameter of the network. We proved that termination detection of the Chandy-Lamport algorithm can be detected. Then, we applied this technique and we proposed a checkpoint (and rollback recovery) algorithm. We can adapt this technique to detect termination of the execution of a distributed algorithm. We defined a weak snapshot as the maximal information a process can compute anonymously on a network knowing an upper bound of its diameter. We proved that classical stable properties as termination, deadlock, loss of tokens or garbage collection can be still detetected with a weak snapshot. We gave an algorithm which enables each process to compute anonymously a weak snapshot. The next question concerns the efficiency of algorithms presented in this paper to detect such properties.

Références

[Ang80] D. Angluin. Local and global properties in networks of processors. In *Proceedings of the 12th Symposium on Theory of Computing*, pages 82–93, 1980.

[CL85] K. M. Chandy and L. Lamport. Distributed snapshots : Determining global states of distributed systems. *ACM Trans. Comput. Syst.*, 3(1) :63–75, 1985.

[GR05] R. Guerraoui and E. Ruppert. What can be implemented anonymously ? In *DISC*, pages 244–259, 2005.

[KRS95] A. D. Kshemkalyani, M. Raynal, and M. Singhal. An introduction to snapshot algorithms in distributed computing. *Distributed Systems Engineering*, 2(4) :224–233, 1995.

[KS08] A. D. Kshemkalyani and M. Singhal. *Distributed computing*. Cambridge, 2008.

[Maz97] A. Mazurkiewicz. Distributed enumeration. *Inf. Processing Letters*, 61 :233–239, 1997.

[SS94] A. Schiper and A. Sandoz. Strong stable properties in distributed systems. *Distributed Computing*, 8(2) :93–103, 1994.

[SSP85] B. Szymanski, Y. Shy, and N. Prywes. Synchronized distributed termination. *IEEE Transactions on software engineering*, SE-11(10) :1136–1140, 1985.

[Tel00] G. Tel. *Introduction to distributed algorithms*. Cambridge University Press, 2000.