Simple distributed solutions to the readers-writers problem

Michel Raynal

▶ **To cite this version:**

Michel Raynal. Simple distributed solutions to the readers-writers problem. [Research Report] RR-1279, INRIA. 1990. inria-00075280

HAL Id: inria-00075280
https://inria.hal.science/inria-00075280

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNITÉ DE RECHERCHE
INRIA-RENNES

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

N° 1279

Programme 3
Réseaux et Systèmes Répartis

SIMPLE DISTRIBUTED SOLUTIONS TO THE READERS-WRITERS PROBLEM

Michel RAYNAL

Août 1990

RR_1279

IRISA INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALEATOIRES

Campus Universitaire de Beaulieu
35042 - RENNES CEDEX
FRANCE
Téléphone : 99.36.20.00
Télex : UNIRISA 950 473F
Télécopie : 99.38.38.32

Simple Distributed solutions to the Readers-Writers problem

Michel RAYNAL
IRISA
Campus de Beaulieu
35042 Rennes - Cédex
FRANCE

Publication Interne n° 543 - Juillet 1990 - 10 Pages

Abstract

Two solutions to the readers-writers problem in a distributed context are presented. They are derived from basic solutions to the distributed mutual exclusion paradigm. These derivations put into prominence the use of properties of well-known basic tools (token and timestamp) in order to obtain different priority strategies.

Key-words : distributed algorithms, mutual exclusion, readers-writers problem, token, timestamping.

Solutions réparties au problème des lecteurs-rédacteurs

Résumé

Deux solutions au problème des lecteurs-rédacteurs dans un contexte de systèmes répartis sont présentées. Les algorithmes obtenus sont issus de 2 solutions de base au problème qu'est l'exclusion mutuelle répartie.

1. INTRODUCTION

A great number of control problems we have to cope with in reliable distributed systems can be solved by using or generalizing solutions to one of the 3 following typical problems or paradigms : election, mutual exclusion or distributed termination detection.

The election problem consists in choosing in the set of the system sites one of them in order to have it obey a particular behaviour [6,7]. Algorithmic tools used to solve this problem have brought particular topological structures (such as ring or spanning trees) into prominence and have allowed a better understanding of their properties, from a distributed computing point of view [10, 12 (chapter 3)].

The mutual exclusion problem was and still is one of the most important control problems of concurrent programming in centralized contexts as well as in distributed ones [11]. The fondamental point solved by mutual exclusion algorithms is the construction of a total order on the executions of a set of a priori possible concurrent operations (the ones that have to be executed in the so-called critical section).

The third typical problem consists in detecting termination of distributed applications [5, 9, 12]. The common point to these problems, the termination detection constitutes the paradigm of, lies in the following observation : what has to be observed in a computation in order to extract behaviour properties? We are generally interested in the so-called stable properties : once true such a property remains true (deadlock and termination are two such properties) [3]. The difficulties to realize such observations come from two origins : first the observation has to not modify the behaviour of the observed application and secondly, in a distributed context, there is no global state which could be grasped instantaneously by some site [13] ; because of the asynchronism of sites and channels, only a previous global state can be obtained [3].

Here we are interested in the mutual exclusion problem in a distributed context. We show how algorithms solving this problem can be extended to give solutions to distributed readers-writers problems. Such an approach is nothing but stressing the paradigmatic nature of the (distributed) mutual exlusion problem. In the following we consider a distributed system of n sites : $S_1, \ldots, S_n$, with a process per site. There is no central memory ; communication between processes/sites is performed along point-to-point reliable channels ; communication delays are arbitrary but finite. We suppose, without loss of generality that there is a channel associated to each pair of sites. A site can at any time express its desire to read or to write a common data. The implementation of the data does not matter ; it can be put on one site, partitionned on several sites, duplicated, etc ; we are only interested in the control rules necessary to manage correctly the reads and the writes and not in the addressing scheme needed to access the data.

2. FROM EATING PHILOSOPHERS TO READERS AND WRITERS.

2.1 Recall of the Chandy-Misra's algorithm.

Chandy and Misra have designed a very general solution to the mutual exclusion problem [4]. An exclusion relation between two sites is expressed by an edge with a token associated to it (the token is called fork or bottle in [4]) ; the set of the sites and of these edges define the conflict graph. The site owner of the token shared by two sites is the priviledged one from the point of view of these two sites. For the mutual exclusion problem we have to consider a fully connected conflict graph (this graph is essentially a logical structure) with the associated tokens (which are concrete objects). As one can see, if a site S_i owns the $n-1$ tokens it shares individually with each of its $n-1$ neigbhours in the conflict graph, then S_i is the only one to own the privilege to enter the critical section.

An initial configuration in the possession of tokens (each token is shared by 2 neighbours in the conflict graph) and a rule in the transmission of tokens are defined in order to ensure fairness (and consequently no deadlocks) for critical section requests. A site wanting to enter the critical section asks for the tokens it does not hold. A token can be in one of two states : clean or dirty. When a site leaves the critical section it dirties the $n-1$ tokens it owns. When S_i receives from S_j a request for the token they share, S_i sends (respt. will send) the token back to S_j if (respt. when) the token is dirty ; the sending of a token cleans it (such a sending can be followed by a request for the token if S_i is requesting the critical section).

From the mutual exclusion point of view the state of the system can be represented by a directed conflict graph. The conflict edge between S_j and S_i is directed from S_j to S_i iff the token is clean and is located at (or moving to) S_i or the token is dirty and is located at S_j. It is easy to see that if this graph is initially acyclic (tokens are initially dirty), the rule governing the state changes and the transferts of the tokens leaves it acyclic. The fairness is deduced from this property (see [4] for a formal proof) : each site wanting to enter the critical section will enter it.

This distributed algorithm is very interesting on several accounts ; in particular in its generality (it solves the so-called concurrent mutual exclusion problem if the conflit graph is not fully connected) and in the fact it needs only bounded variables in its implementation.

2.2 Extension to solve a readers-writers problem.

Two types of exclusion occur in readers-writers problems. First write operations are mutually exclusive ; this exclusion is a simple one : we associate a shared write token *wtok (i, j)* to every pair of sites S_i and S_j and we use the preceding algorithm with these tokens to solve conflicts between writes (*wtok (i, j)* and *wtok (j, i)* are synonyms for the same token).

The second kind of exclusion is between a reader and a writer. To solve it one can use the same principle as above. First we associate to each site of every pair of sites S_i and S_j a read token : $rtok_i\ (i, j)$ is associated to S_i and $rtok_j\ (j, i)$ to S_j. Secondly we define the rules governing exclusion between a read and a write. There are now 3 tokens travelling between every pair of sites : a shared write token (with 2 internal states : clean or dirty) and two read tokens (without internal states). The conditions are :

Condition (read by S_i) : S_i owns the (n-1) read tokens associated to it :
$\forall j \neq i$: $rtok_i\ (i, j)$ *are at* S_i.

Condition (write by S_i) : S_i owns :

1. the (n-1) write tokens it shares with its neigbhours :
$\forall j \neq i$: $wtok\ (i,j)$ *are at* S_i

and

2. the (n-1) read tokens associated to its neigbhours :
$\forall j \neq i$: $rtok_j\ (j, i)$ *are at* S_i.

It is easy to see that these two conditions ensures the safety property of the readers-writers problem. In order to avoid deadlocks (some kind of lineness property) one can give some priority to the writers over the readers. A writer first asks for and obtains all the write tokens it shares with its $n-1$ neigbhours and only afterwards it asks for the $n-1$ read tokens it needs. Write tokens are released as in the Chandy-Misra 's algorithm ; read tokens are sent to their callers when they are not used and are requested. Others strategies can be devised to give the readers some priority.

This use of two kind of token processes have to acquire is, in some respect, similar to multi-dimensionnal voting schema [2].

3. A BAKERY ALGORITHM.

In the preceding algorithm to give the writes priority is much easier than to design a strategy ensuring reads and writes are scheduled in a fifo order (fifo meaning here write/write and read/write conflicts being solved according to an order consistent with the Lamport's causality relation [8]). In this case there is no "absolute" priority of one operation over the other. A common way to ensure such a fifo order is to associate a timestamp to every read or write request, these timestamps building a total order over the set of the requests [8, 13, 14].

In order to obtain a very general solution we assume a site S_i can call any operation belonging to some type : c_1, c_2,... or c_m. Exclusion relations are expressed by a symmetric compatibility boolean matrix ; *compat* (c_{k1}, c_{k2}) is true iff executions of type c_{k1} and c_{k2} are not exclusive. Concerning the readers-writers problem we have *compat (read, read) = true*, the matrix values being set to *false* anywhere else.

From the set of the timestamp-based mutual exclusion algorithms we choose the Ricart-Agrawala 's one [14] as a skeleton to express the readers-writers rules with a fifo ordering. The underlying principle is simple. When a site wants to execute an operation of type c_k, it sends a request to the *n-1* others sites and waits for the *n-1* corresponding authorizations. Every request is composed of the operation type c_k and of a timestamp ; a timestamp is a pair : " value of the local logical clock + 1, identity of the site". When it receives such a timestamped request *req* $(c_k, (h, j))$ the site S_i sends back its authorization to S_j if it is not involved in an operation of type c_1, ..., c_m ; it also sends back its authorization if it is involved in an operation of type c_p such that *compat* (c_k, c_p) or if $\neg$ *compat* (c_k, c_p) and its request *req* $(c_p, (lr, i))$ has a higher timestamp than the S_j's request : $(lr, i) > (h, j)$ ie. :

$$lr > k \text{ or } (lr = k \text{ and } i > j)$$

In the other case (the request of S_i has priority over S_j 's one and the two requests are conflicting) the site S_i delays its authorization sending (it will send it when the execution of its operation ends).

It is easy to see that this algorithm ensures a fifo order [8] between writes and between reads and writes and that reads between two writes can be executed concurrently (consequently it satisfies the liveness property).

A more formal description of the algorithm follows. Each site S_i is endowed with the following local variables :

```
dem : boolean init false ;
expected, delayed : set of ( 1, 2, ..., n ) init Φ ;
lh : 0..+∞ init 0 ; % local clock of S_i %
lr : 0..+∞ init 0 ; % last request's clock value %
typop : ( c_1 ..., c_m ) ;
```

The site S_i behaves as follows with respect to the readers-writers problem :

upon a call to a c_p *operation*

```
dem      : = true ;
lr       : = lh + 1 ; typop : = c_p ;
expected : = (1, 2, ..., n) - (i) ;
∀ j ≠ i  : send req ( c_p, (lr.i)) to S_j ;
wait (expected = Φ) ;
% this operation is interruptable by message reception %
< execute the c_p operation > ;
dem      : = false ;
∀ j ∈ delayed : send ok (i) to S_j ;
delayed : = Φ ;
```

upon receiving $ok(j)$

```
expected : = expected - (j)
```

upon receiving $req\ (c_k, (h,j))$

```
lh := max (lh, h) ;
if (¬ dem) or
   (dem and compat (c_k, typop)) or
   (dem and ¬ compat (c_k, typop) and (h,j) < (lr,i))
then          send ok(i) to S_j
else % dem and ¬ compat (c_k, typop) and (lr,i) < (h,j) %
        delayed : = delayed ∪ (j)
fi
```

If we consider the special case of only one (type of) operation c_i such that $\neg\ compat\ (c_i, c_i)$ the algorithm we obtain is the original Ricart-Agrawala's one ensuring distributed mutual exclusion between calls to the c_i operation. 2 (n-1) messages are needed to execute an operation ; this number can be reduced using techniques such as the ones described in [1, 13].

4. CONCLUSION.

These extensions of basic mutual exclusion algorithms to solve distributed readers-writers problems show that the mutual exclusion is a paradigm of distributed control. Moreover the two algorithms presented show the kind of basic algorithms one needs in order to ensure some priority rules : timestamp-based algorithms are well suited for fifo ordering whereas multi-token-based algorithms seem best suited for other kinds of priority rules.

REFERENCES :

[1] CARVALHO O., ROUCAIROL G.
On mutual exclusion in computer networks.
Comm. ACM, Vol. 26, (Feb. 1983), pp. 146-147.

[2] CHEUNG S.Y., AHAMAD M., AMMAR M.H.
Multidimensionnal voting : a general method for Implementing synchronization in distributed systems.
Proc. oth IEEE Int. Conf. on DCS, Paris (June 1990), pp. 362-369.

[3] CHANDY K.M., LAMPORT L.
Distributed snapshots : determining global states in distributed systems.
ACM TOCS, Vol. 3, 1, (Feb. 1985), pp. 63-75.

[4] CHANDY K.M., MISRA J.
The Drinking philosophers problem.
ACM Toplas, Vol. 6, 4, (Oct. 1984), 632-646.

[5] FRANCEZ N.
Distributed Termination.
ACM Toplas, Vol. 2, 1, (Janv. 1980), pp. 42-55.

[6] GARCIA-MOLINA H.
Elections in a distributed computing system.
IEEE Trans. on Computers, Vol. C31, 1, (Jan. 1981), pp. 48-59.

[7] LE LANN G.
Distributed systems : towards a formal approach.
IFIP Congress, North-Holland, Toronto, (1977), pp. 155-160.

[8] LAMPORT L.
Time, clocks and the ordering of events in a distributed system.
Comm. ACM, Vol. 21, 7, (July 1978), pp. 558-565.

[9] MATTERN F.
Algorithms fo distributed termination detection.
Distributed Computing, Vol. 2, (1987), pp. 161-175.

[10] PETERSON G.L.
An O (n logn) Unidirectional ring algorithm for the circular extrema problem.
ACM, Toplas, Vol. 4, 4, (October 1982), pp. 758-762.

[11] RAYNAL M.
Algorithms for mutual exclusion.
The MIT Press, (1986), 107 p.

[12] RAYNAL M.
Distributed Algorithms and Protocols.
Wiley and sons, (1988), 142 p.

[13] RAYNAL M.
Distributed computations and Networks : Concepts, Tools and Algorithms.
The MIT Press, (1988), 166 p.

[14] RICART G., AGRAWALA A.K.
An optimal algorithm for mutual exclusion in computer networks.
Comm. ACM, Vol. 24, 1, (Jan. 1981), pp. 9-17.

LISTE DES DERNIERES PUBLICATIONS INTERNES

PI 536 **TOWARDS DOCUMENT ENGINEERING**
Vincent QUINT, Marc NANARD, Jacques ANDRE
Mai 1990, 20 Pages.

PI 537 **YALTA : YET ANOTHER LANGUAGE FOR TELEOPERATE APPLICATIONS**
Jean-Christophe PAOLETTI, Lionel MARCE
Juin 1990, 32 Pages.

PI 538 **SYNCHRONOUS DISTRIBUTED ALGORITHMS : A PROOF SYSTEM**
Michel ADAM, Jean-Michel HELARY
Juin 1990, 20 Pages.

PI 539 **CONCEPTION DE DESCRIPTEURS GLOBAUX EN ANALYSE DU MOUVEMENT A PARTIR D'UN CHAMP DENSE DE VECTEURS VITESSES APPARENTES**
Henri NICOLAS, Claude LABIT
Juin 1990, 38 Pages.

PI 540 **DOCUMENT DESCRIPTION LANGUAGE INTERPRESS**
Nenad MAROVAC
Juin 1990, 26 Pages.

PI 541 **UN SIMULATEUR QUALITATIF DE CHAINES BIOLOGIQUES POUR L'AIDE AU DIAGNOSTIC MEDICAL**
Monique LE COZ, Daniel PICART
Juillet 1990, 54 Pages.

PI 542 **A NEW APPROACH TO VISUAL SERVOING IN ROBOTICS**
Bernard ESPIAU, François CHAUMETTE, Patrick RIVES
Juillet 1990, 44 Pages.

PI 543 **SIMPLE DISTRIBUTED SOLUTIONS TO THE READERS-WRITERS PROBLEM**
Michel RAYNAL
Juillet 1990, 10 Pages.

PI 544 **IMPLEMENTATION AND EVALUATION OF DISTRIBUTED SYNCHRONIZATION ON A DISTRIBUTED MEMORY PARALLEL MACHINE**
André COUVERT, René PEDRONO, Michel RAYNAL
Juillet 1990, 14 Pages.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique

ISSN 0249 - 6399

ISSN 0249-6399