A simple taxonomy for distributed mutual exclusion algorithms

Michel RAYNAL
IRISA
Campus de Beaulieu
35042 Rennes-Cedex
raynal@irisa.fr

Abstract

This short paper examines the two basic principles from which distributed mutual exclusion algorithms are designed : permission-based and token-based principles. This presentation is done in a pedagogical way and is illustrated by references to existing algorithms.

Key-words : Mutual exclusion, distributed system, permission, token.

1 Introduction

Since 1977, with Le Lann's paper[10], and 1978 with Lamport's one [9], lots of distributed algorithms have been proposed to solve the mutual exclusion problem. Only few of them were very innovative, proposing new ideas or new algorithmic techniques. The major part were designed by adapting previous and well-known solutions to specific hypotheses. The aim of this short paper is not to review all these algorithms with their underlying principles and properties (the reader may consult [16]) but to present the two simple ideas from which distributed mutual exclusion are designed. These two principles define two families for such algorithms : the permission-based family (§2) and the token-based one (§3). (These two families meet when we consider that the right to enter the critical section is managed by a central coordinator (§4).) We limit our presentation to the principles attached to each of these two families.

We consider in the following a distributed system composed of n sites : $P_1, \ldots, P_n$. Communication between processes is by messages only (there is no central memory) and transfer delays are finite but unpredictable [17]. We suppose, without loss of generality, there is one and only process per site.

2 Permission-based algorithms

This idea was first expressed by Ricart and Agrawala in 1981 [19] ; then it has been formalized by Sanders [21]. The idea is very simple : when a process wants to enter the critical section it asks the others for them to give it the permission to enter ; and then it waits until these permissions have arrived. If a process is not interested by the critical section it sends back its permission as soon as it receives the requests ; if it is interested a priority has to be established between the two conflicting requests. In the Ricart-Agrawala's proposal a timestamp mechanism (obeying the Lamport's rule [9]) is used to associated a timestamp to each request ; as timestamp are totally ordered, conflicts are easily solved in favour of the request endowed with the lowest timestamp.

In the Ricart-Agrawala's proposal a process must ask for the permissions of a statically defined set including all the other processes. Improvements have been proposed concerning the management and the

size of this set. In [4] a process P_i, which has received a permission from P_j, dont ask P_j again if it was granted the critical section, and it did not receive a request from P_j since this granting ; consequently, in this proposal, the size of the set varies dynamically for each process. (This management is the result of a systematic method used to ensure that a global assertion is always satisfied in a distributed system [3]). In [5] this is again inproved by eliminating the need for timestamps and using instead the acyclicity of a directed graph (the vertices are processes) representing request priorities (the solution obtained-namely the drinking philosophers problem- needs only bounded variables).

In the distributed database field, majority and quorum-based protocols are well-known techniques used to ensure consistency of updates [7,24] ; moreover this enable algorithms to resist faults such as crashes or partitionnings [2,6]. In fact permission-based algorithms and majority (or quorum) based protocols use the same idea : collect "enough rights" allowing to do something. So several algorithms have been proposed which differ in the size of "enough" (i.e. for each process of the set of processes granting permissions). Maekawa, using a technique based on finite projective plane, reduces it to $\sqrt{n}$ [12]. Agrawal and El Abbadi reduces it to *log(n)* by using a tree logical structure superimposed on the distributed system [1].

To sump up : in this family the safety property (that states mutual exclusion is never violated) is ensured by obtaining of a sufficient number of permissions ; and the liveness property (each request will be granted) is ensured by totally ordering the requests either by associating a timestamp to each one or by managing a distributed acyclic directed graph, the vertices of which are the requesting processes.

3 Token-based algorithms

In this case the principle used is very simple : as only one process at a time can enter the critical section, the right to enter is materialized by a special object which is unique in the whole system, namely a *token*. As one can see the safety property is trivially ensured as the token is unique. The only thing one has to manage is the movement of the token from one process to another one in order each request be satisfied (liveness property). At this point two possibilities can be considered for such a movement : the perpetuum mobile and the token-asking method.

In the perpetuum mobile the token travels from one process to another one to give them the right to enter the critical section ; and in order not to forget the request of some process the processes are put on a directed logical ring used by the token. (If a process receives the token and if it is not interested in the critical section it passes it to the next process along the ring). The perpetuum mobile on a ring ensures the liveness. This principle has been proposed by Le Lann [10] and used in several algorithms [13].

In the token-asking method, the token dœs not move by itself ; a process willing to enter the critical section asks for the token (if it dœs not own it) and waits until the token arrives. Several refinements are possible. In [20] the requests of each process are sent to all the others, each process counts the number of request received and the token carries the number of critical section uses per process ; there is sufficient information to ensure a movement of the token ensuring requests liveness. In [8] this principle is extended to a network with an arbitrary topology. In [11,14,15,25] the requesting processes are logically structured (by the requests) as a tree and the movement of the token consists in a sequential tree traversal [18,22] (a visited process is suppressed from this tree as the token leaves it).

Another distinction can be made according to the ways requests are disseminated in the system to reach the token. [8] uses a parallel floading technique [22] ; [14,15] use an underlying logical structure (a tree) to allow requests to join more quickly the token ; in [23] a process can use some heuristic to guess with a good likelihood where the token is, in order to send less requests.

4 A special case

In the case of a central coordinator, statically defined, the two principles meet. Processes ask only the coordinator for the permission before entering, and this unique permission can be assimilated to a token managed by this coordinator. So this case can be seen as the separation point between the two principles.

5 Conclusion

The aim of this short paper was not to pretend to exhaustiveness about particular characteristics and properties of distributed mutual exclusion algorithms. (References also are not exhaustive ; refer to [16] for a more appropriate list). Its only goal was to state basic principles from which such algorithms are designed and consequently can be understood and better mastered. Such an understanding can help when, more generally, one has to implement or design distributed control algorithms.

References

[1] AGRAWALA D., El ABBADI A. *An efficient solution to the distributed mutual exclusion problem.* Proc. 8th ACM Symposium on PODC, (August 1989), pp. 193-200

[2] AGRAWALA D., El ABBADI A. *Exploiting logical structures in replicated databases.* Inf. Proc. Letters, vol.33, (1990), pp. 255-260

[3] CARVALHO O.S.F., ROUCAIROL G. *On the distribution of an assertion.* Proc. 2d ACM Symposium on PODC, (1982), pp. 121-131

[4] CARVALHO O.S.F., ROUCAIROL G. *On mutual exclusion in computer networks.* Comm. ACM, vol.26,2, (1983), pp. 145-147

[5] CHANDY K.M., MISRA J. *The drinking philosophers problem.* ACM Trans. on Prog. Languages and Systems, vol.6,4, (1984), pp. 632-646

[6] GARCIA-MOLINA H., BARBARA D. *How to assign votes in a distributed system.* Journal of the ACM, vol.32,4, (1985), pp. 841-860

[7] GIFFORD D.K. *Weighted voting for replicated data.* Proc. 7th ACM Symposium on Op.Systems Principles, (1989), pp. 150-159

[8] HELARY J.M., PLOUZEAU N., RAYNAL M. *A distributed algorithm for mutual exclusion in an arbitrary network.* The Computer Journal, vol.31,4, (1988), pp. 289-295

[9] LAMPORT L. *Time, clocks and the ordering of events in a distributed systems.* Comm. ACM, vol.21,7, (1978), pp. 558-565

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

[11] LYNCH N.A., TUTTLE M. *Hierarchical correctness proofs for distributed algorithms.* Proc. 7th ACM Symposium on PODC, (1987), pp. 137-151

[12] MAEKAWA M. *A $\sqrt{n}$ algorithm for mutual exclusion in decentralized systems.* ACM Trans. on Comp. Systems, vol.3,2, (1985), pp. 145-159

[13] MARTIN A.J. *Distributed mutual exclusion on a ring of processors.* Science of Computer Programming, vol.5, (1985), pp. 256-276

[14] NAIMI M., TREHEL M. *A distributed algorithm for mutual exclusion based on data structures and fault tolerance.* Proc. 6th Int. Phoenix IEEE Conf. on Comp. and Comm., Scottsdale, (1987), pp. 35-39

[15] RAYMOND K. *A tree-based algorithm for distributed mutual exclusion.* ACM Trans. on Comp. Systems, vol.7,1, (1989), pp. 61-77

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

[17] RAYNAL M. *Networks and distributed computations : concepts, tools and algorithms.* The MIT Press, (1988), 166 p.

[18] RAYNAL M., HELARY J.M. *Synchronization and control of distributed systems and programs.* Wiley and sons, (1990), 200 p.

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

[20] RICART G., AGRAWALA A.K. *Author response to "on mutual exclusion in computer networks" by Carvalho and Roucairol.* Comm. ACM, vol.26,2, (1983), pp. 147-148

[21] SANDERS B. *The information structure of distributed mutual exclusion algorithms.* ACM Trans. on Comp. systems, vol.5,3, (1987), pp. 284-299

[22] SEGALL A. *Distributed network protocols.* IEEE Trans. on Inf. Theory, vol. IT 29,1, (1983), pp. 23-35

[23] SINGHAL M. *A heuristically-aided algorithm for mutual exclusion in distributed systems.* IEEE Trans. on Computers, vol.38,5, (May 1989), pp. 651-662

[24] THOMAS R.H. *A majority consensus approach to concurrency control for multiple copy databases.* ACM Trans. on Database Systems, vol.4,2, (1979), pp. 180-209

[25] Van de SNEPSCHEUT J.L. *Fair mutual exclusion on a graph of processes.* Distributed Computing, vol.2, (1987), pp. 113-115