Journal of Computer Science 3 (1): 43-46, 2007
ISSN 1549-3636
© 2007 Science Publications

Towards an Exclusion Mutual Tolerant Algorithm to Failures

[1]M. Senouci, [2]A. Liazid and [1]D. Benhamamouch
[1]Informatic Department Es-Sénia-University, BP. 1524, El Mnaouer, 31000 Oran, Algeria
[2] LTE Laboratory; ENSET-Oran; BP. 1523, El Mnaouer, 31000 Oran, Algeria

Abstract: The distributed algorithmic is widely used in various economic and industrial fields as finance, medical, industry, telecommunication. New technologies are increasing rapidly. From now on, applications must associate two conditions: auto-execution on heterogeneous grid computers, on the one hand and on the other hand, satisfaction of the temporal and safety constraints. Hence this paper deals with performing an algorithm based on an innovative idea based on the mutual exclusion which permits to adjust the access conflict to the shared resources and to synchronize sites in a distributed system. The performed algorithm was tolerant to breakdowns. A site wanting to enter in critical section demands the permission of all sites of it set called quorum. This quorum notion assures the mutual exclusion even in the case of breakdowns. The presented algorithm was exempt of deadlock and assures the equity.

Key words: Mutual exclusion, distributed system, logic arborescence, quorum

INTRODUCTION

The computer systems designed to manage the high level security and inaccessible processes must not fall in breakdown. This is the case of the aerial transportation, atomic energy, distributed industrial processes and other process.

Indeed, it is necessary to consider in the computer distributed systems the breakdown possibility of a site. This likely failing must not affect the working of the whole system. Consequently the fault tolerance must be studied to ensure a very strong safety of working in the distributed architectures. Tolerance to faults can be assured without redundancy and the distributed concept of processing and data is an interesting solution to solve this aspect. Consequently, tolerance to faults is a motivating factor to design the distributed algorithm. Cooperation between the various tasks of a distributed system implies the conception of access protocols to the shared resources to get at a given instant the global system state. The global network group security must be considered[1]. Indeed, the mutual exclusion is an important tool of cooperation between distributed systems. Several attempts have been achieved to conceive a distributed algorithm of mutual exclusion tolerating to breakdowns. Many authors have tried to decrease the messages amount that demand access to the critical section in order to reduce the algorithm complexity and to improve its breakdown tolerance.

This paper deals with the different strategies proposed for development of distributed algorithm. In this paper an algorithm is proposed using the quorum notion which is relatively well adapted to the previous proposed algorithms[2-5] and for their optimisation . The hypotheses of the model as well as its algorithm are presented. The algorithm principle and its implementation are then presented. Finally, the complexity of the algorithm is discussed.

Strategies adopted until now: The mutual exclusion principle consists at each instant in assuring for each system site the possibility to obtain a privilege. This privilege permits to execute an action called critical section.

Let's consider for a context, a distributed network where the process of every site interacts only while transmitting messages to itself (several algorithms are described by Raynal[6]). Let's recall some principles; Le Lann[7] presents an algorithm of mutual exclusion for the logically on-line sites according to a ring structure. Lamport[8] introduced the logical clock concept to date requests and to order and satisfy them sequentially. To satisfy every access demand in critical section, $3*(N-1)$ messages are necessary (N: is the number of network sites). Ricart and Agrawala[9] proposed an improvement while bringing back the number of messages to $2*(N-1)$.

Let's note other propositions of improvement[10-14]. All theses propositions use the principle of the token: the demand of authorization from another site, in order to satisfy access demands in critical section. The loss of the token is a problem that requires an algorithm of regeneration which is complicated. When the access demand in critical section is significant then the amount

Corresponding Author: M. Senouci, Informatic Department Es-Sénia-University, BP. 1524, El Mnaouer, 31000 Oran, Algeria

of the exchanged messages becomes important (in *O (N)*).

The proposed algorithm permits to reach two objectives:

* To find a minimal set of sites which allows an applicant to enter in critical section with respect of the mutual exclusion.
* To decrease the message amount and to increase fault tolerance.

Model and hypotheses: A distributed system is considered as a finite set of sites that evolves in parallel and communicates between them by messages exchanges. The possibility to communicate between sites is modelled by a non oriented associative graph called communications graph.

Nodes of this graph represent the sites. Each site is identified by only one number. In the present model, no site plays the role of a main agent then no site has the global state of the system.

Three hypothesis types are required. The first hypothesis concerns the network which is considered complete (the *N* sites can communicate directly between them) and the links between sites are bi-directional. The second hypothesis concerns the communication system. It is supposed that:

* No loss of messages occurs.
* The transmission delay is arbitrary but finite.
* The messages are not duplicated and not altered.
* If messages of different sites arrive simultaneously, they are taken sequentially.

The third hypothesis deals with the critical section. Any site in critical section must keep away from after a finite time.

Algorithm principle: To avoid inconveniences of the algorithm presented by Senouci[15], Naïmi *et al.*[16] and Trehel *et al.*[4] and that consists in providing a system deadlock in case of quorum breakdown and the risk in its construction, we propose a new approach based on a distributed algorithm which is tolerant to faults and solve the problem of the mutual exclusion. The characteristics of the algorithm are:

* The use of a new function for the quorum construction.
* The reduction of the exchanged messages amount to access in the critical section.
* The tolerance to faults

The 'coterie' *C* is a family of sets where each set q of *C* is called quorum $C = \{q_i\}$. Quorums[17] verify the following properties:

Intersection: if *p* and *q* are two quorums, then *p* and *q* must have a non emptiness intersection ($p \cap q \neq \varnothing$)

Minimality: there are not two quorums *p* and *q* in a coterie *C* as *p* is a subset of *q*.

The coterie notion has been used to develop protocols that guarantee the mutual exclusion in a distributed system. The mutual exclusion is assured knowing that every two quorums have a common site. It is supposed that every site has and manages a waiting line that contains the request messages. These requests are ordered according to their stamping. A site that wants to execute its critical section sends a demand to all quorum sites and waits their permission. When a request is at the top of the line of a site, then this site sends a message called *Replay* to the claimant site. When a site leaves the critical section, it sends an acquittal message named *Relinq* to all quorum sites; therefore they can remove the corresponding requests. In the case where a request arrives with a stamp lower than the stamp of the request which is at the top of the line, a message called *Inquire* is sent the site whose request is in the top of the line and awaits the reception of either message: *Yield* or *Relinq*. If a site receives the *Inquire* message and if it received all answers of the quorum, then it ignores this message; otherwise it sends the *Yield* message to the site that sent the *Inquire* message. The quorum construction[17] is made by extracting a tree from the initial network. When a site wants to construct a quorum, it calls to the following recursive function called Build_ Quorum with its identifier as a parameter:

```
Function Build_ Quorum (node: site): quorum_ set;
            Begin  Return ({node}
             (Build_ quorum_ father (node. F, node)
             (Build_ quorum_ child (node. C)
              );
End
Function Build_ Quorum_ Father (node, son, site): quorum_ set;
Begin
             If (node = Nil) then return ({ })
             Else If (State (node ≠ fail)
             Return ({ node}(Build_ Quorum_ Father (node. F, node));
                   Else
begin
                         Select a Child C such that
                          Node. C ≠ son;
                          State (node.C) ≠ fail;
                          Return ({node. C}
                             (Build_ Quorum_ Father (node. F, node)
                             (Build_ Quorum_ Child (node. C);
                                   );
                          End;
 End;
Function Build_ Quorum_ Child (node: site): quorum_ set;
Var quorumchild: array [1.. maxchildren] of quorum_ set;
 Begin
 If (node = Nil) then return ({ })
 Else If (state(node) ≠ fail then
           Begin
                 Select a Child C such that
                       State (node. C) ≠ fail;
                 Return ({node} (Build_ Quorum_ Child (node. C));
            End
 Else
            Begin
                  (I ([1.. node. Numberchild];
 quorumchild [I]:= Build_ Quorum_ Child (node. Childi);
            return ((quorumchild [I]);
            End;
End;
```

Arborescent logic: The *N* nodes of a distributed system are arranged logically in a tree with any degree. A path in the tree is a sequence of nodes $a_1, a_2, \ldots, a_i, a_{i+1}, \ldots, a_j$, *as* a_{i+1} is the node son of a_i.

A quorum is constructed while grouping all nodes of a path leading from the root node to the pending node. If a node of the path is not functional then all paths which begin from its sons and end at pendings can replace it[3,18]. For example, let's consider the binary tree of the Fig. 1 which is extracted from an organized complete network with nine (09) sites all initially supposed operational.

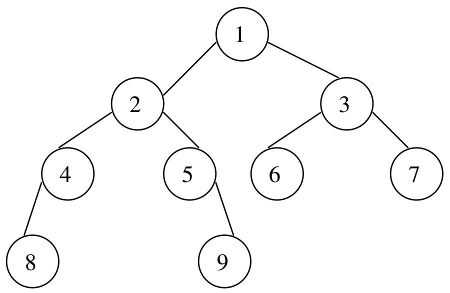

Fig. 1: Logical structure of the network

At a given instant t_i, let's imagine the following script:

t0: the site 6 wants to enter in critical section, it constructs its quorum which is {1,3,6}, it sends its request to these sites.

t1: the site 6 enters in critical section after receiving the permission of its quorum. Therefore, the waiting line of sites 1, 3 and 6 contains the identity of the site 6.

t2: the site 4 wants to enter in critical section, it constructs its quorum which is {1,2,4,8} or {1,2,4,9} (there is a choice), then it sends its request to sites of its selected quorum.

t3: the site 4 receives the permission of the sites 2 and 8 and possibly from itself. There remains for it the permission of the site 1 that keeps up the respect of the mutual exclusion.

t4: the site 6 leaves its critical section; it sends the *Relinq* message to sites of its quorum so that they can remove its identity of their waiting lines.

t5: now, the site 1 can send its permission to the site 4, that enters in critical section.

t6: sites 1, 2, 3 and 8 fall in breakdown.

t7: the site 6 wants to enter in critical section, it constructs its quorum, that will be in this case {4,5,6,7,9}.

It can be noted that the sites 5,7 and 9 give their permission to the site 6 (because their waiting lines are empty), it is the site 4 that adjusts in this case the mutual exclusion.

From this example, it can be noted that even though the root of the logical structure falls in breakdown, the proposed algorithm permits to rebuild a new logical structure. There resides the innovative idea of this work.

Algorithm presentation: The Build_ Quorum (node) is the local function of every site which is executed in an indivisible manner. Build_ Quorum (node) constructs the quorum of the claimant node.

Consider the primitives of communication and synchronization: SEND (MSG) to *j*: *j* designates the receptor of message MSG. This primitive is not binding, its execution is indivisible.

WAIT (Condition): the site remains waiting until the condition becomes true.

When a site *j* detects the failing of one of its neighbours, for example site *i*, it sends a message called FAIL(i) to all sites of the network (broadcasting). At the reception of the message FAIL(i) by a site *k*, two situations can occur:

* If the line of the site *k* contains a request of the site *i*, then the site *k* deletes it.
* If the site *k* wants to enter in critical section and if the site *i* belongs to its quorum, then it ignores this quorum and construct another.

Insurance of the process: Thanks to the property of quorum minimality the mutual exclusion is guaranteed.

Even in the case of sites breakdown of a quorum (except the one that is in critical section) the mutual exclusion remains insured, because every site in critical section contains its identity in its line.

A Deadlock may be defined as the presence of a circuit in the graph of awaiting sites.

Thanks to the strong tolerance to breakdowns of the algorithm, the deadlock is avoided. In case of quorum failing, two situations are distinguished:

* The site in critical section is not faltering, then it is present in all subsequently constructed quorums and it keeps the mutual exclusion.
* The site in critical section is faltering and then the critical resource is free and attainable by all sites.

The equity is defined positively by Raynal[6]: $\forall p_i$: the critical section is attainable by p_i.Thanks to the symmetry of the proposed algorithm and previously hypothesis, the equity is relatively guaranteed.

Complexity study: The arborescence structure has returned the easy implementation contrary to the other structures such as the Petri networks. The used variables are of limited size.

* Either distributed systems composed of *N* sites; or the complete tree of degree *d and* with depth *h* that represents the system distributed in the algorithm, (Fig. 2). Let's consider the algorithm:
* The cardinality Card (quorum_set)=$h+1$, (complete tree).
* A site waiting to enter in critical section sends the Request message to its quorum sites (except itself). Consequently it sends [card(quorum_set)-1] messages, that are equal to *h* messages.

* The claimant site waits to receipt *h* Reply messages.

Therefore, the accesses at the critical section, in a network represented by a complete tree of depth *h*, require precisely (*2*h*) messages.

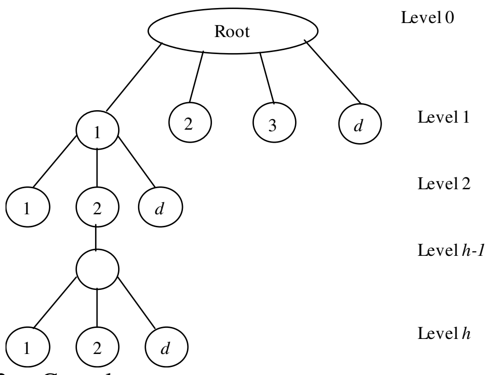

Fig. 2: Complete tree

Considering the Fig. 2 where it is clear that the depth *h* is a function of *N* nodes, *h = f(N),* then:

$$N = 1 + d + d^2 + d^3 + ... + d^{h-1} + d^h \Leftrightarrow N = \sum_{i=0}^{h} d^i$$

$$\Rightarrow N = \frac{(d^{h+1} - 1)}{(d-1)} \Rightarrow d^{h+1} = N*(d-1)+1$$

$$\Rightarrow h+1 = \log(N*(d-1)+1)/\log d$$

$$\Leftrightarrow h+1 = \log[N*(d-1)*((1+N*(d-1))/(N*(d-1)))]/\log d$$

$$\Leftrightarrow h+1 = \log[N*(d-1)+\log(1+1/(N*(d-1))]/\log d$$

As $1/(N*(d-1)) \rightarrow 0$ then $h+1 = \log(N*(d-1))/\log d$

$$\Leftrightarrow h+1 = [\log N + \log(d-1)]/\log d$$

$$\Leftrightarrow h+1 = [\log N/\log d] + [\log(d-1)/\log d]$$

however $[\log(d-1)/\log d] \approx 1$, then:

$$h+1 = [\log N/\log d] + 1 \Leftrightarrow h = \log N/\log d$$

Therefore the highest value of h is $h \leq (\log N/\log d)$. Consequently, the exchanged messages number in a complete tree with degree d and N nodes, reaches the critical section in (*2*h*).

Remark: In the case where the tree is not complete, this number in included in the interval: 2*(1...h).

CONCLUSION

This paper apprehended the mutual exclusion problem by introducing a new idea based on the concept of quorum which permitted to reach several objectives:

* Reduction of the messages number. This number is less than (log*N*/log*d*).

* Access to the shortest path to route the demand.

* Distribution of the demand volume of the waiting lines of each site.

* Tolerance to sites failures.

Let's note, however, that our results brought about the need of some important developments. One of them could be raised by the tool of synchronization and the stamping since it represents an implementation problem because of its no-limited aspect. Another one could be raised by the overcharge on the root.

REFERENCES

1. Amir, Y., C. Nita-Rotaro, J. Stanton and G. Tsudik, 2003. Scaling secure group communication: Beyong peer-to-peer. Intl. Proc. DISCEX3, Washington DC, April 22-24.
2. Raynal, M., 1985. Distributed Algorithm and Protocols. (In French). Eyrolles Edition, Paris.
3. Senouci, M., M. Meftah and H.A. Beghdadi, 2005. Introduction to Operating System Theory (In French) Dar El Gharb Edn., Algeria.
4. Trehel, M. and A. Housni, 1999. Introduction of the priority in distributed mutual exclusion algorithms. ISAS'99, 5th Int. Conf. on Information Systems Analysis and Synthesis, Orlando, (USA) pour 31 Jul.-4 Aug.
5. Housni, A., M. Trehel and A. Arnol, 2001. A log*N* distributed mutual exclusion algorithm for two groups. Proc. of ACM Symp. on Appl. Computing, pp: 531-538, Las Vegas.
6. Raynal, M., 1984. Algorithmic of the Parallelism: The Problem of the Mutual Exclusion (In French). Dunod Edition, Paris.
7. Le Lann, G., 1977. Distributed Systems towards a Formal Approach. IFIP Congress, Toronto Canada, pp: 155-160.
8. Lamport, L., 1978. Time, clocks and ordering of staleness in a distributed system. ACM., 21: 558-565.
9. Ricart, G. and A.K. Agrawala, 1981. An optimal algorithm for mutual exclusion, in computer networks. ACM., 24: 9-17.
10. Carvalho, O.S.F. and G. Roucairol, 1983. On mutual exclusion in computer network. Proc. ACM, 26: 3-5.
11. Maekawa, M., 1985. A $\sqrt{N}$ algorithm for mutual exclusion in decentralized systems. ACM Trans. Comp. Syst., 3: 145-159.
12. Naimi, M., 1987. An arborescent structure for a class of algorithms distributed of mutual exclusion. Ph. D. Thesis, University of Besançon- France.
13. Cao, G. and M. Singhal, 2001. A delay-optimal quorum-based mutual exclusion algorithm for distributed systems. IEEE Trans. Parallel and Distributed Systems, 12: 1256-1268.
14. Marin, B., A. Luciana and S. Pierre, 2004. Hierarchical token based mutual exclusion algorithms. Research Report, INRIA, REGAL Team, Rocquencourt, pp: 20.
15. Senouci, M., Y. Slimani and K. Hadouda, 1994. To fault tolerating algorithm for distributed mutual exclusion. Proc. of the 7th Intl. Computer Science Days, pp: 121-130, Tunis.
16. Naïmi, M., M. Trehel and A. Arnold, 1996. A log*N* distributed mutual exclusion algorithm based on the path reversal (In French). J. des Systèmes Parallèles et Distribués, 34 : 1-13.
17. Amir, Y. and A. Wool, 1998. Optimal availability quorum systems. Inf. Process. Lett., 65: 223-228.
18. Agrawal, D. and A. El-abbadi, 1991. An efficient and fault-tolerating solution for distributed mutual exclusion. ACM Trans. Computer Systems, 9: 1-20.