International Journal on Emerging Technologies **4**(2): 124-127(2013)

ISSN No. (Print) : 0975-8364
ISSN No. (Online) : 2249-3255

A Survey of Mutual Exclusion Algorithms in Distributed Computing

Rahul Singh*, Jagrati Malviya** and Kanchan Jha**
**Department of Information Technology, BUIT BU Bhopal, (MP)*
***Department of Computer Science and Engineering, BUIT BU Bhopal, (MP)*

(Received 15 October, 2013 Accepted 26 December, 2013)

ABSTRACT: Over the last 10 years distributed computing systems have attracted a great deal of attention. The problem of mutual exclusion in distributed systems has attracted considerable attention over the last decades. The mutual exclusion problem requires that, at a time, only one of the deserving processes be allowed to enter its critical section (CS). Number of solutions has been proposed to the mutual exclusion problem in distributed systems. Different algorithms have used different techniques to achieve mutual exclusion and have different performances. In this paper, we present a survey of different approaches such as token based approach, permission based approach and hybrid approach.

Keywords: Distributed System, Mutual Exclusion, Critical section.

I. INTRODUCTION

A distributed system is a collection of autonomous computers connected via a communication network. There is no common memory and processes communicate through message passing. One of the most important purposes of the distributed systems is to provide an efficient and convenient environment for sharing of resources. They also provide computational speedup and better reliability [1]. A distributed system consists of a collection of geographically dis- persed autonomous sites connected by a communication network. The sites have no shared memory and communicate with one an- other by passing messages [2].

Many operations in a distributed system require mutual exclusion to guarantee correctness.

Distributed systems usually have some shared resources to which concurrent access is not permitted. Shared resources are usually accessed through procedures called Critical Sections (CS). At any time, only one site is petted to execute the CS. A Mutual Exclusion Algorithm defames the rules to coordinate entry into the CS and mediate conflicts when two or more sites desire to execute a CS at the same time. In the absence of shared memory, the algorithms depend on message ex- changes between sites to coordinate entry into the CS [3].

II. MUTUAL EXCLUSION IN DISTRIBUTED COMPUTING SYSTEMS

A distributed computing system is a collection of autonomous computing sites that do not share a global or common memory and communicate solely by exchanging messages over a communication facility.

In a distributed computing system any given site (also referred to as "node") has only a partial or incomplete view of the total system and a system-wide common clock does not exist. Processes must share common hardware or software resources, cooperating in such a way that they can work in parallel and independently of each other. The access to a shared resource must be synchronized to ensure that only one process is making use of the resource at a given time.

Each process has a code segment, called a critical section, in which the process can access the shared resource. The problem of coordinating the execution of critical sections by each process is solved by providing mutually exclusive access in time to the critical section. A process is said to execute repeatedly a sequence of non-critical section code and critical section code segments, each of finite execution time. Each process must request permission to enter its critical section and must release it after it has completed its execution.

A mutual exclusion algorithm must satisfy the following requirements:

1. At most one process can execute its critical section at a given time.
2. If no process is in its critical section, any process requesting to enter its critical section must be allowed to do so in finite time.
3. When competing processes concurrently request to enter their respective critical sections, the selection cannot be postponed indefinitely.
4. A requesting process can not be prevented by another one to enter its critical section within a finite delay.

To simplify, an algorithm must provide mutually exclusive access to a resource, ensure deadlock freedom, ensure starvation freedom, and must provide some fairness in the order that requests are granted.

Two approaches can be used to implement a mutual exclusion mechanism in a distributed computing system. In a centralized approach, one of the nodes functions as a central coordinator. Processes ask only the coordinator for permission to enter their critical section. Only when a requesting process receives permission from the coordinator can it proceed to enter its critical section.

The central coordinator is fully responsible for having all the information of the system and for granting permission to make use of a shared resource.

In a distributed approach, the decision making is distributed across the entire system and the solution to the mutual exclusion problem is far more complicated because of the difficulty to obtain a complete knowledge of the total system. This is due to the lack of a common shared memory, a common physical clock and because of unpredictable message delay.

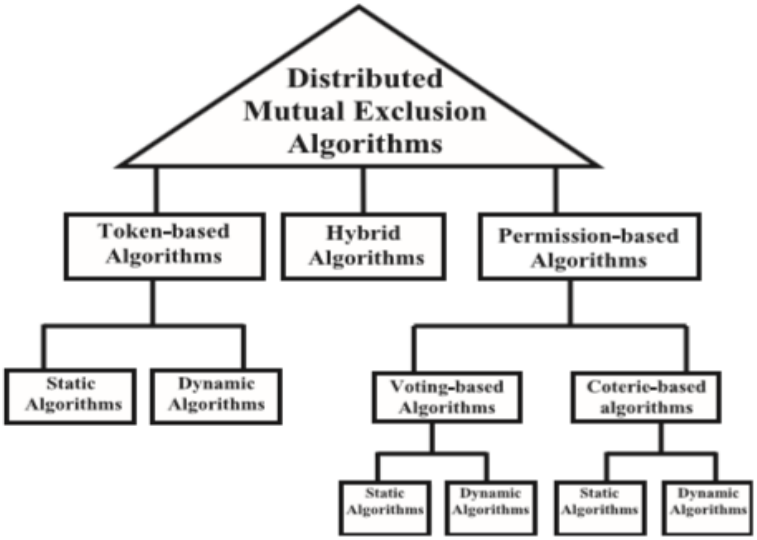

Fig 1: Classification Tree for DME Algorithm.

III. BASIC APPROACHES FOR THE DESIGN OF DISTRIBUTED MUTUAL EXCLUSION ALGORITHMS

Distributed mutual exclusion algorithms can be classified into two groups by a basic principle in their design. These two groups are token-based algorithms and permission-based algorithms. The basic principle for the design of a distributed mutual exclusion algorithm is the way in which the right to enter the critical section is formalized in the system.

A. The token-based approach

In the token-based group the right to enter a critical section is materialized by a special object, namely a token. The token is unique in the whole system. Processes requesting to enter their critical section are allowed to do so when they possess the token. The token gives to a process the privilege of entering the critical section. A token is a special type of message. The singular existence of the token implies the enforcement of mutual exclusion. Only one process, the one holding the token, is allowed to enter to its critical section.

At any given time the token must be possessed by one process at most. Granting the privilege to enter the critical section is performed by a single process, which is the current owner of the token. This process chooses the next token owner and sends it the token. A distinction has to be made between the mechanisms used to move the token among the processes in the system. If processes are logically organized in a direct ring structure, the token can travel around the ring from process to process to give them the right to enter the critical section. If a process receives the token and it is interested in the critical section (CS), it can proceed to its execution. After the process exits its CS the token is released to circulate again. On the other hand, if the process is not interested in its CS it just passes the token to the next node in the logical ring. If the ring is unidirectional, starvation freedom is ensured. Under light load this method has a high cost since the token message circulates even if no process wants to enter the CS, but it is very effective under high load.

Another method to move the token in the system is by asking for it when a process wants to enter its CS. A requesting process sends a request message to the token holder and waits for the token arrival. After completing the execution of its CS, the process holding the token chooses a requesting process and sends it the token. If no process wants to use the token, the token holder does not need to send the token away. Using this method a major concern is how to locate the token holder in order to minimize message exchanges originated by a requesting process.

The token-based approach is highly susceptible to the loss of the token, since this can induce a deadlock situation. Also, problems can occur with the existence of duplicated tokens. Complex token regeneration must be executed to ensure the uniqueness of the token.

B. The permission-based approach

In the permission-based group the right to enter a critical section is formalized by receiving permission from a set of nodes in the system. A process wishing to enter its critical section asks the others to give it their permission to proceed; and then it waits until these permissions have arrived. A process enters its CS only after receiving permission from all nodes in a set.

Non-requesting processes send their permission to requesting ones. Each process may grant its permission to only one process at a time. A priority or an order of events has to be established between competing requesting processes so only one of them receives permission from all other nodes in the set.

Only one process, the one that has received permission from all members of a given set of nodes, is allowed to enter the critical section. This enforces the requirement for mutual exclusion.

Granting the privilege to enter the critical section is performed by the set of nodes that send their permission to requesting processes. Conflicts are solved by a priority or an order of events mechanism.

The problem of finding a minimal number of nodes from which a process has to obtain permission to enter its CS has to be considered. This can be translated as to how many rights a process has to collect in order to proceed to the execution of the critical section. Many protocols have been developed to find a majority or quorum of processes from which rights have to be collected. The solution to this problem has a direct impact in the cost of messages exchanged per mutual exclusion invocation [4]..

C. Hybrid Approach

A hybrid approach to mutual exclusion is proposed to minimize both message traffic and time delay at the same time. A hybrid mutual exclusion algorithm using the release local sites first mode, the requesting group semantics, and the requesting sequence is found to be an efficient way to control the interaction.

IV. DESCRIPTION OF TOKEN-BASED ALGORITHMS

In this algorithm, a completely different approach to achieving mutual exclusion in a distributed system is introduced. Here we have a specific network topology with no inherent ordering of the processes. In software, a logical ring is constructed in which each process is assigned a position in the ring. The ring position may allocate in numerical order of network addresses or some other means. It dose not matter what the ordering is. All that matter is that each process who is next in line after itself. When the ring is initialized, process 0 is given a token. The token circulate around the ring. It is passed from process k to process k+1, in point-to-point messages. When a process acquires the token from its neighbor, it checks to see if it is attempting to enter a critical region. If so, the process enters the region, dose all the work it needs to, and leaves the region. After it has exited, it passes the token along the ring. It is not permitted to enter a second critical region using the same token.

If a process is handed the token by its neighbor and is not interested in entering in critical region, it just passes it along. As a consequence, when no processes want to enter any critical regions, the token just circulates at high speed around the ring. As usual, this algorithm has problems too. If the token is ever lost, it must be regenerated. In fact, detecting that it is lost is difficult, since the amount of time between successive appearances of the token on the network is unbounded. The fact that the token has not been spotted for an hour dose not mean that it has been lost; somebody may still be using it. The algorithm also runs in trouble if a process crashes, but recovery is easier than in the other cases [1]. Deadlock occurs if no node is in its critical section and there are two or more processes wishing to enter the CS, but they are not able to do so. This could occur essentially if the token is lost or does not eventually reach a node which has requested it. The loss of the token cannot be easily distinguished from system connectivity loss. The existence of more than one token would violate the mutual exclusion requirement, thus the detection of a token loss is not a trivial task [4].

Token based algorithm are broadcast based or logical structure based,static or dynamic. Some examples of token based algorithm are Ricart-Agrawala algorithm, Suzuki-Kazami algorithm, Mizuno-Neilsen-Rao, Neilsen-Mizuno algorithm, Helary-Plouzeau-Raynal algorithm, Raymond's algorithm, Singhal's algorithm, Naimi-Trehel algorithm, Mishra-Srimani. Mishra and Srimani algorithm,Nishio-Li-Manning algorithm.

V. DESCRIPTION OF PERMISSION-BASED ALGORITHMS

Permission-based algorithms require rounds of message exchange among the nodes to obtain the permission to execute CS. The basic idea on which permission-based algorithms are based is as follows: When a process wants to enter its CS, it asks other nodes for their permission. A process on receiving a request grants its permission if it is not interested in CS. If it is interested in CS, the priority of the incoming request is established against its own request. Generally, priority decisions are made using timestamps [5]. Some examples of token based algorithm are Lamport's algorithm, Ricart-Agrawala algorithm, Carvalho-Roucairol algorithm, Ricart and Agrawala algorithm, Raynal's algorithm, Maekawa's algorithm, Sanders' algorithm, Agrawal-El Abbadi algorithm, Singhal's algorithm.

VI. CONCLUSION

In this study, we have presented many distributed mutual exclusion algorithms. And principles and characteristics have been described, and their cost in the number of messages exchanged for an entry to a critical section (CS) to take effect has been shown.

We find that different algorithms give different performance capabilities with respect to different metrics. These algorithms can be evaluated on the basis of performance measures like message complexity, delay of synchronization, availability, communication delay, etc. An algorithm may be optimal with respect to another of these parameters but may show poor performance as regards another. Which algorithm to choose for any particular application is an important decision for system designer? This is totally depending on a designer, for choosing an algorithm.

RFERENCES

[1]. Mohammad Rastegari and Amir Masoud Rahmani. Solving Critical Section problem in Distributed system by Entangled Quantum bits Department of Computer Engineering, University of Science and Research, Tehran February, 2008.

[2]. Ye-In Chang, Mukesh Singhal, and Ming T. Liu. A Hybrid Approach to Mutual Exclusion for Distributed Systems, Dept. of Computer and Information Science The Ohio State University Col UmbuS, OH 43210-1277.

[3]. Supriya Madhuram and Anup Kumar, A Hybrid Approach for Mutual Exclusion in Distributed Computing Systems Engineering Mathematics and Computer Science University of Louisville, Louisville, KY 40292.

[4]. A Survey of Distributed Mutual Exclusion Algorithms Martin G. Velazquez Technical Report CS-93-116 September 6, 1993.

[5]. P.C. Saxenaa and J. Raib. A survey of permission-based distributed mutual exclusion algorithms, A School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India bDepartment of Mathematics, P.G.D.A.V. College (Eve), University of Delhi, New Delhi, India Received 24 May 2002; received in revised form 8 October 2002; accepted 11 October 2002.