Quorum Based Algorithms using Group Choice

JaeHyrk Park[1], Kwangjo Kim[1] and Yoshifumi Manabe[2]

[1]International Research Center for Information Security (IRIS)

Information Communication Univ, Korea

[2]Nippon Telegraph and Telephone Corporation, Japan

Abstract

This paper discusses the quorum based algorithm for group mutual exclusion defined by Yuh-Jzer Joung. Group mutual exclusion[4,5,6] is a generalization of mutual exclusion that allows a resource to be shared by processes of the same group, but requires processes of different groups to use the resource in a mutually exclusive style. Joung proposed a quorum system, which he referred to as the surficial quorum system for group mutual exclusion and two modifications of Maekawa's algorithm[6]. He mentioned that when a process may belong to more than one group, the process must identify one of the groups it belongs when it wishes to enter CS(Critical Section). However, his solution didn't provide mechanism of identifying a group which maximizes the possibility to enter CS. In this paper, we provide a mechanism for identifying that each process belongs to which group.

I. Introduction

Quorum-based mutual exclusion algorithms are resilient to node and communication line failures. In order to enter its critical section, a node must obtain permission from a quorum of nodes. The design issues for group mutual exclusion using quorum system have been modeled by Joung[6]. Before mentioning group mutual exclusion, we overview distributed mutual exclusion algorithm area. A distributed system can be viewed as a set of processes that share many types of resources, such as processors, memory cells, and printers, many of which must be accessed in a mutually exclusive manner; that is to say, they can be accessed by at most one process at a time. Therefore the mutual exclusion problem - that of guaranteeing mutually exclusive access to a resource - is considered to be a basic problem in distributed systems[1,2,3,4,5,6]. The rest of this paper is organized as follows: Section 2 presents the distributed 1-mutual exclusion. Section 3 presents the group mutual exclusion which is related to our research topic. Section 4 discusses new algorithm using group information. Section 5 summarizes our conclusions.

II.Mutual Exclusion

1.Distributed mutual exclusion

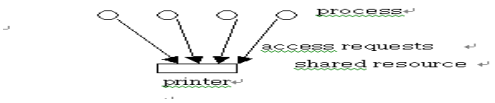

Figure 1 : Distributed mutual exclusion

The first distributed mutual exclusion algorithm is proposed by Lamport. To guarantee Mutual exclusion, no deadlock, and no starvation, he proposed a logical clock. The pair of a logical time and a process identifier is

called a timestamp. By using timestamps, his algorithm avoids starvations and deadlocks. To decrease the number of message per mutual exclusion invocation and to increase the availability, the concept of coterie is proposed by Garcia-Monlina and Barbara[2].

We explain concept of 1-mutual exclusion. When processes in a distributed system share a resource which must be accessed exclusively, the access to the resource must be controlled as in the case of concurrent processes in stand-alone operating systems. To enter a critical section, a process must assure that there is no process which is being in a critical section in the distributed system. Many algorithms have been proposed to solve the distributed mutual exclusion problem. Also, such algorithms follow one of these three types.

(1)Permission-based principle

A process P wishing to enter a critical section requests some other processes to permit it to enter a critical section. If a permission is given from each process P is asking, P can enter the critical section.

(2)Token-based principle

A process can enter a critical section while it is holding the token. The mutual exclusion is guaranteed because there is only one token in the system and there are no two processes having a token at the same time.

(3)Quorum based principle

A process can enter a critical section when it receives permission from a set of processes. The set of processes is called a quorum. The quorum system C =(Q_1 , Q_2, ... , Qm), where $Q_i \subseteq P$ satisfying the following conditions.

Intersection Property

$$\forall Qi, Qj \in C, Qi \cap Qj \neq \emptyset$$

Minimality Property

$$\forall Qi, Qj \in C, Qi \subseteq Qj$$

III.Group Mutual Exclusion

1. Concept

Group mutual exclusion is formulated by Joung in 1998, based on the classical mutual exclusion problem[5]. Group mutual exclusion is a generalization of mutual exclusion that allows a resource to be shared by processes of the same group, but requires processes of different groups to use the resource in a mutually exclusive style. In group mutual exclusion a process requests a session that is resource area which user want to use before entering its critical section; processes are allowed to be in the critical section simultaneously provided they have requested the same session. An example of group mutual exclusion, described by Joung, is a CD juke box shared by multiple processes: Any number of processes can simultaneously access the currently loaded CD, but processes wishing to access a CD other than the currently loaded one must wait. In this case, the sessions are the CDs in the juke box. Group means as follows: Let P = 1,...,N be a set of nodes. Let G = (g_1,...,g_m) be the set of groups. Each request v is associated with the set of groups it belongs.

2 Requirement of Group mutual exclusion

Group mutual exclusion is necessary to satisfy below condition.

Mutual exclusion

At any given time, no two processes of different groups are in CS simultaneously.

Lockout freedom

A process wishing to enter CS will eventually succeed.

Concurrent entering

if some processes request a session and no process requests a different session, then the processes can concurrently enter the CS.

3. Quorum Based Algorithm for Group Mutual Exclusion

Let's show briefly definition of m-group quorum system by Joung. Let P={1,...,N} be a set of nodes, which belong to m groups[6]. Each Node does not belongs to groups, but each request belongs to groups.

An m-group quorum system : A=(C(1),··,C(m)) over P consists of m sets, where each $C(i) \subseteq 2^P$ is a set of subsets of P satisfying the following conditions[6]:

Non-emptiness

$$\forall 1 \le i \le m,\ Q_i \neq \emptyset$$

Intersection Property

$$\forall 1 \le i, j \le m, i \neq j,\ \forall\ Q_1 \in C_i, \forall\ Q_2 \in C_j \Rightarrow Q_1 \bigcap Q_2 \neq \emptyset$$

Minimality

$$\forall 1 \le i \le m, \forall\ Q_1,\ Q_2 \in C_i\ \ Q_1 \neq Q_2 \Rightarrow Q_1 \sqsubseteq Q_2$$

Suppose a quorum member gives permission to only one process at a time. Then, by the intersection property, no two processes of different groups can be in CS simultaneously.

IV. New algorithm using group information

1. Problem of Joung's algorithm

Joung presented two modifications of Maekawa's algorithm[6] so that the number of processes that can access a resource at a time is not limited to the structure of the underlying quorum system, but to the number that the problem definition allows. His algorithm use special entry policy to increase concurrence performance of same group. A reference process

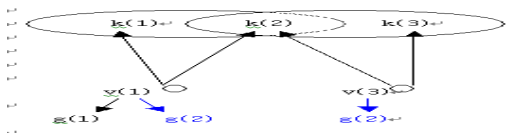

Figure 2 : Problem of Joung's algorithm

is used such that subsequent requests from the same group are also granted so long as the reference remains in CS. Maximal resource utilization can be achieved by allowing more processes of the same group to share CS, regardless of whether some other group of processes are waiting for CS or not[6]. However, in Joung's algorithm, new processes of the same group cannot enter CS when there are requests with different groups.

He mentioned that when a process may belong to more than one group, the process must identify a unique group to which it belongs when it wishes to enter CS. However, His algorithm didn't provide mechanism of identifying a unique group. Let's show Figure2. Assume process v(3) is in CS as group g(2). When process v(1) belongs to g(1) and g(2) , v(1) needs to identify a group (g(1) or g(2)) before making request. If v(1) selects g(1), v(1) can not enter the CS. That means even though v(1) belongs to g(2), v(1) can not enter the CS at the same time. We provide mechanism for identifying the group which maximizes the possibility to enter CS.

2. New algorithm using Group Choice mechanism

As mentioned before, Joung's algorithm didn't consider group information. We got the idea from allocation algorithm using local coterie. As distributed resource allocation algorithm, the processes maintain a distributed database, which

keeps pairs each consisting of a process and a resource it is currently accessing [7]. In order to keep pairs, algorithm use D(v) which that memorizes whether or not r is allocated to a process for each r∈R(u) which is a set of resources.

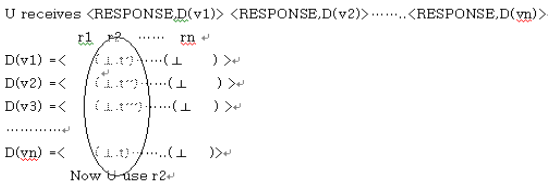

Figure 3 : Mechanism of allocation algorithm

Joung didn't consider identifying each process belongs to which group. That is to say, even though each process that wants to enter critical section belongs to same group, if one process choose another group, it cannot critical section. To avoid this problem, we use A(v). Each process stores a pair of each node and group allocation information in A(v), after response of each node. Each process that sends request to each node of one quorum(a set of processes) has their own A(v) variable. When process that acts as a group member receives all response from each node, process checks content of A(v) and chooses group. This mechanism is safe because A(v) just consist of ⊥ of one group. For example, the case of A(v)={(V(1),⊥),(V(2),⊥),(V(3),⊥)} or A(v)={(V(3),g(1)),(V(4),⊥)} can happen. not A(v)={(V(1),g(2)),(V(2),g(3))}. Out of processes that want to enter critical section, just processes of same group can enter the critical section, which means all processes in the critical section belong to same group at that time. Through such mechanism, processes of same group can enter CS safely. We can not only avoid the problem of Joung's algorithm but also increase concurrency and decrease message complexity.

V. Conclusion

In this paper, we modified Joung's Quorum-based algorithm for group mutual exclusion and proposed new group mutual exclusion algorithm. This research about group mutual exclusion algorithm can be related to cryptographic protocols such as secure broadcast Byzantine protocols which are a fundamental building block for implementing replication in fault-tolerant distributed systems using threshold signatures. Also, quorum system concept in distributed algorithm may be used for secure multi party protocols. The efficient and secure message passing way in distributed algorithm can be used for secure communication between multi party based on cryptographic techniques. As a future work, we will be focusing on cryptographic protocols using distributed algorithm in distributed system.

Reference

[1] S.Ichiro and K.Tadao,"A Distributed Mutual Exclusion Algorithm", ACM, Vol.3,No.4, pp344-349 , Nov.,1985.

[2] H.Garcia-Molina and D.Barbara, "How to assign votes in a distributed systems", JACM, Vol.32,No.4, pp841-860, Oct.,1985.

[3] Y. Manabe and S.Aoyagi, "A distributed k-mutual exclusion algorithm using k-coterie", COMP 93-43 , 1993.

[4] V.Hadzilacos, "A note on group mutual exclusion", In Proc. 20th PODC, 2001.

[5] Y.-J.Joung, "Asynchronous group mutual exclusion. (extended abstract)". In Proc. 17th PODC, pp51-60, 1998. Full paper in Distributed Computing, Vol.13,No.4, pp189-206, 2000.

[6] Y.-J.Joung, "Quorum-Based Algorithms for Group Mutual Exclusion", LNCS 2180, pp16-32, 2001.

[7] H.Kakugawa and M.Yamashita, "Local Coteries and a Distributed Resource Allocation Algorithm",Information Processing Society of Japan. Vol.37,No.8, Aug.,1996.