A HYBRID GROUP DISTRIBUTED MUTUAL EXCLUSION ALGORITHM BASED ON PRIORITY MECHANISMS

DR. PAWAN K THAKUR[1] AND VIVEK CHAUDHARY[2]

1. **Associate Professor, Department of Computer science and Engineering, Govt. College Dharamshala, H.P. (India) pawansarkaghat@gmail.com**
2. **Research Scholar , Career Point University, Kota , Rajasthan; viveksalil@gmail.com**

ABSTRACT

A distributed system is a software system in which components located on networked computers communicate and coordinate their actions by using different methods such as tokens, messages or quorum. The main reason with mutual exclusion problem is that when concurrent access to share resources to shared resources by different sites are made. Only one process is allowed to execute the critical section at any given time. The group mutual exclusion problem is generalization of mutual exclusion problem where processes in the same group can enter the critical section simultaneously.

In this paper , we propose a Hybrid Group Mutual Exclusion Algorithm based on priority. When the processes of different types wants to enter the critical section, their priorities will be checked and then only they will be allowed to enter the critical section .Our algorithm is hybrid type which uses the concept of token as well as message passing.

Keywords: Distributed systems, Distributed group mutual exclusion, critical section.

1. INTRODUCTION

Mutual exclusion is one of the important concepts of distributed system. The main reason for mutual exclusion problem is that when concurrent access to share resources by different sites are made [15]. Mutual exclusion is the fundamental issue in the design of distributed systems. Only one process is allowed to execute the critical section at any given time. The shared variables cannot be used to implement mutual exclusion in distributed system. The design of distributed mutual exclusion algorithms

have to deal with unpredictable message delays and incomplete knowledge of system state[1]. The three basic approaches for implementing distributed mutual exclusion are used. These are: Token based approach[2][10], Non token based approach[6][5][14][8] and Quorum based approach[4][12]

In token based approach, a logical token representing the access right to the shared resource is passed in a regulated manner among the sites. The site who is having the token is allowed to enter the CRITICAL SECTION. Mutual exclusion is ensured because the token is unique. The algorithms based on this approach have to search the token .These types of algorithms provide better message complexity and easy to extend but the loss of token is the bottleneck .

In non token based approach , each site freely and equally competes for the right to use the shared resource. The message are used among the sites to determine which site will enter the CRITICAL SECTION. A site enters the CRITICAL SECTION when an assertion, defined on its local variables become true. The assertion becomes true only at one site at a given time and it ensures the mutual exclusion. These type of Algorithms are fault tolerant but at the cost of increased message complexity.

In quorum based approach, each site request permission to execute the critical section from a subset of sites. This set of sites is called quorum. Any two quorums contains a common site. This common site is responsible to make sure that only one request executes the CRITICAL SECTION at any time. These type of Algorithm have lesser message complexity because they have to take permission from the subset and not from all the processes in the system but the problem of creating and initialization of quorum is there.

In this paper, we consider the problem of distributed group mutual exclusion[18] which is generalization of distributed mutual exclusion problem. In distributed mutual exclusion , only one process can enter the critical section whereas in group mutual exclusion , different processes of same group can enter the critical section. The multiple reader/single writer problem is a special case of group mutual exclusion problem. For reading one group is used and for writing different group is used. If different requests are coming from different processes for the purpose of reading(same group), then those processes can concurrently

read the data. For example , consider the application, where different data is stored in the different pen drives attached to the system. The users who need to access the same data from the currently used pen drive , can access the data concurrently. The users who need to access the data from different pen drive will have to wait until the currently used pen drive finished all its requests or there is some conflicting request.

2.0 Related work:

The problem of GME was firstly given by Yuh-Jeer Joung[18]. There are different GME algorithms which are based on the following categories:

(i) If a process wants to enter the critical section , it send requests to some processes and after getting the reply, it enters the critical section.

(ii) The second GME category is token based . In this case the process which is having the token can enter the critical section.

(iii) The third GME category uses both the above methods. The process who wants to enter the critical section , obtains permissions from some processes(quorum) and then processes of same type can use the concept of token.

Based on above three categories , different algorithms were proposed. We will discuss some of the algorithms used in the GME. Joung proposed two different algorithm for GME. These are Joung's broadcast based algorithm[16] and Joung's quorum based algorithm[17]. Joung's broadcast based algorithm was an extension of Ricart and Agarwala distributed mutual exclusion algorithm[15]. Joung proposed two algorithms RA1 and RA2. In RA1, the process which wants to enter the critical section , sends a request message to all the processes and upon receiving reply message from all the processes, it enters the critical section. There are some concurrency related issues in RA1, which was later solved by using RA2. In Joung's quorum based algorithm , the concept of quorum is used. A process has to obtain permission from all the processes in the quorum to enter critical section. For concurrency, Joung proposed two algorithms Maekawa_M, which sends message in parallel. A serial version called Maekawa_S, which obtains sequential permission from each process in quorum. These two algorithms avoids deadlock .

A concept of surrogate quorum[11] was given by Atraya and Mittal algorithm. In this algorithm , when a process obtains a permission in a quorum to enter the critical section, it becomes the leader and all other processes in quorum have to take permission from the leader. This algorithm has low message complexity but concurrency is the issue because when leader is in critical section, it cannot grant requests. Mittal and Mohan[9] proposed an algorithm called TokenGME. It is an extension of Suzuki and Kasami[7] algorithm. It has two types of token primary and secondary. When a process wants to enter the critical section , first of all it has to acquire primary token of particular type. On receiving requests of the same type, the primary token issues the secondary token to all the requesting processes and all the processes of same type enters the critical section. In this way concurrency is achieved.

Different proposed algorithms do not consider the waiting time and execution time of the process.In our algorithm , we have calculated the value of the process which is in the request queue and only that process will be selected which is having the maximum value. The different factors are considered such as waiting time, execution time , priority , age and group size. This value is dynamic and keeps on changing in the real environment.

3.0 System Model:

The distributed system consists of set of n processes and a set of communication channels. The distributed system is asynchronous and does not have global clock. Information is exchanged between different processes by passing message asynchronously. We have assumed that message delay is finite and processes are non faulty and channels are reliable.

3.1 Group mutual exclusion problem:

The problem of GME was firstly given by Yuh-Jeer Joung[18]. In GME problem the processes which are competing for the critical section , must be placed in the request queue. From the request queue , main token is assigned to the process after considering the different factors such as waiting time, execution time , priority , age and group size. If a process has been granted the main token of a particular type, then this process will grant the sub tokens to different processes of the same type. If there are n processes of the same type, then n processes can enter the critical section.

Group mutual exclusion satisfies the following properties:

(i) Safety: This property states that there will be only one main token in the system. At any time , the number of main token should not exceed more than one which further states that if the two processes are of different group, then they can not enter the critical section simultaneously.

(ii) Liveliness: This property ensures that every process gets a chance to enter the critical section and it avoids unnecessary blocking and starvation.

(iii) Concurrent entry: If the processes belongs to the same type, then they can enter the critical section concurrently.

3.2 Performance metrices:

(i) Message Complexity: It is the number of messages required to enter the critical section by any process.

(ii) Message size complexity: It is the data which is Piggybacked from the message.

(iii) Concurrency: it is the number of processes which are in the critical section at a given time.

(iv) Synchronization delay: It is the time when process from the current session exits from the critical session and next process from the different session enters the critical section.

4.0 A new hybrid group mutual exclusion algorithm based on priority:

In this section we present a new hybrid group mutual exclusion algorithm considering the different factors such as waiting time , priority , execution time , age and size of the group. Our algorithm solve the group mutual exclusion problem and also increases the concurrency.

4.1 Outline:

It is a hybrid algorithm. It uses the concept of message passing and tokens. In this algorithm , two type of tokens are used, one is main token and other is sub token. Initially , the concept of message passing is used and then algorithm uses tokens to enter the critical section. In the beginning , process P_i is having the authority to enter the critical section i.e. it is having the Main Token. Later when there are number of processes in the system, the message passing concept is used to select the Main node

which will have the authority to enter the critical section. This main node will be assigned the main token. When some other process makes the request, then the type of that requesting process is checked , if it is same as that of process holding the main token, then sub token is issued to that process and the process enters the critical section. It is repeated again and again and the type of process which is requesting the critical section is checked, and the process enters the critical section if type is same. If a request from process say P_k comes to enter the critical section and its type is different then its request can not be fulfilled immediately and this type of request is entered into the request queue of the Main token. A process which is having the main token can execute the critical section as long as some conflicting request arrives. If the conflicting request comes from the process which is having different type , then the process of release token begins. On knowing about the conflicting requests , all the processes which are having the subtokens will start the process to release the tokens. Thereafter the main token will be passed to another process selected from the pending request queue. Here we have presented a mechanisms to select the next process. The next process will be selected from the request queue based on the factors described above.

4.2 Sorting the Request queue:

In various algorithms , queues are sorted by using FIFO policy. But FIFO policy does not consider different factors such as waiting time, priority and execution time. These parameters are important for the applications. We will use the formula which will allow us to insert a new process in request queue.

Vi,c(t)= t/(sessioni –waiting time)* Pri *1/(Execution time) +subset age + subset size

Here t is the current time of requesting process, Pri is the priority of the requesting process. Subset age is the sum of all the ages of the subset and subset size is the number of processes in the subset. To increase the priority of requesting processes which are having low execution time , we will use 1/execution time in our formula.

4.3 Safety criteria:

Once the token_transfer function is initiated , the next process which becomes the main process i.e. selected from the request queue, can not use the token immediately. First of all , it has to wait for the release of subtokens acquired by the previous session. If all the

subtokens are released associated with the previously acquired main process , then only the next process from the another session will enter the critical session.

4.4 Description of the algorithm:

In this algorithm , we have used a number of factors for selecting the next session. When the next session is selected , a number of factors such as waiting time , execution time, priority, age , size must be considered. On selection of next session, first the size of the session is calculated and it should be maximum in the request queue. However it will lead to the problem of starvation. To avoid starvation, the concept of age is used.

For selecting the next session, the type of that session is considered. Here requests in the request queue is divided into subsets based on type. The value of that request is calculated which has been described in the section " Sorting the request queue". While selecting the next session , this value is considered. Now that process in the next session is selected where the process has maximum value in the request queue. This process will hold the main token and all the processes which are in the subset will hold the sub tokens.

A formal description of the proposed algorithm is shown in figure.

In figure1 we have described the different variables which have been used in our algorithm. Figure2 shows the initialization part. Figure3 shows the sequence of events when process P_i generate the request of type x. here process P_i is the initial token holder and it will use the token and enter the critical section. Otherwise the request vector is incremented and request message is broadcast to all the processes. It is shown in figure4 when the request message has been received from the process P_j. If it is a new request , it update the request vector else it is checked whether the process is main token holder and its type is same, then it is granted the subtoken and process enters into the critical section. If the session of the token is less than zero, then its request is entered into the request queue and call to subtoken is initiated. Here one more condition is tested for different type of token. If the token is of different type , then release message is broadcasted to all the processes.

In figure5, it is shown that P_i is the main token holder and it issues subtoken requests to processes of same type. In figure6, the procedure to leave the critical section is explained. Add the request of the new process in the request queue if it is not added

otherwise the procedure to call send token is invoked. If the process is main token holder and conflicting request arrives , then the process to release all the secondary token begins.

Figure7 explains what happens when release message is received from process P_j . Here the next session is checked and if it is safe then it enters the critical section. In figure8 , the procedure to send token is explained. Here the request queue is checked . The value of process P_i is calculated as explained in "Sorting the queue section". This value is based on waiting time, priority, groupsize , execution time and age. That process is selected whose value is maximum and the process to transfer main token and secondary token begins.

Variables:
Requesti:Vector[1…n]of tuple<no,type>
(sessioni) :sequence number of latest session that Pi knows through release message.
No_release:Number of release message that Pi receives from session
M_Token: Main token with the following attributes:

(i) current_gp
(ii) gp_size
(iii) idle
(iv) type
(v) priority
(vi) age
(vii) session
(viii) noofsubtokens_p: No of subtokens issued for the previous session
(ix) request_queue
(x) granted:vector[1…n] of number of granted requests for each process.

Figure1

Initialisation:
Session=0
No_release=0
M_tokeni.granted=0
M_tokeni.idle=true
M_tokeni.session=0
M_tokeni.noofsubtokens_p=0
M_tokeni.request_queue=0
M_tokeni_noofsubtokens_c=0

Figure2

```
Procedure Request(When process Pi generate request for using critical section of type x)
If( M_token is not null) and (M_tokeni=0) and (M_tokeni.type=x) then
        Set M_tokeni.session=1
        Check M_tokeni is safe
        Enter the critical section
Else
        Increment requesti.number
        Broadcast request message to all processes.
```

Figure3

```
Procedure Receive(number,x)(when request message is received from some process Pj)
If request(number) of process Pj less than number
        Update update requesti vector if new request
If ( process Pi is main token holder)  and (Pj request is new request)
        If (M_tokeni.session>0 ) and (request of same type)
                Send sub_token to process Pj
                Increment M_tokeni.granted[j]
        Else add Pj request to request_queuei
        If M_tokeni.idle then call send_token()
Else if(M_tokeni.type is different) and (M_tokeni.idle )
        Broadcast release message to all processes.
```

Figure 4

```
Procedure receive(isM_tokeni)(when process Pi is main token holder)
Check if the M_tokeni is main token holder and M_tokeni is safe
For(k=0;k<=n;k++)
If (process Pk request is same type)
        Send sub_token to Pk
        Increment M_tokeni.granted[k]
        Increment noofsubtokens_c
If(M_tokeni is safe)
        Enter the critical section
```

Figure 5

```
Procedure leave(when process Pi leave the critical section)
If (M_tokeni is idle) and (token is M_tokeni)
        For(k=0;k<=n;k++)
        If(request of Pk>granted[k]
                Add Pk request to request_queuei
        Call send_token()
Elseif (Pk request if of different type)
        Broadcast release message to all processes.
```

Figure6

```
Procedure Release( When the release message is received from process Pj)
If( session<session)
        Set session=session
Else(session=session)
        Increment no_release
If(M_tokeni is not null) and(M_tokeni is safe)
        Enter the critical section
```

Figure7

```
Procedure send_token()
If request_queue is non empty(calculate the value of process Pi in pending requests)
        Vi(t)= Vi,c(t)= t/(sessioni –waiting time)* Pri *1/(Execution time) +subset age +
    subset size
If(value Vi(t) of process Pj is maximum say x)
        Assign session to process Pj
        Send sub_tokens to all the processes of current session of type x
```

Figure8

5.0 Theoretical analysis of the algorithm:

This algorithm satisfies the safety, liveliness ,starvation free and concurrent properties. In this algorithm it is clearly mentioned that there is only one main token in the system i.e.

There is only one main token in the system at any time (1)

Suppose there are two tokens T1 and T2 . Now if session(T1) is equal to the session (T2) then it means that type of T1 and T2 are same (2)

If P_i is holding T1 and it is executing critical section then it satisfies the safety property for T1. (3)

Proof of safety:

Assertion: If two processes P_i and P_j are executing the critical section concurrently, then the session must belong to same type.

Proof: Suppose P_i is having token T1 and P_j is holding token T2 and P_i and P_j are using the critical section simultaneously.(Using 2). It implies that token T1 and T2 holds the safety property. (Using 3), which further implies that session(T1)=session(T2) and Type(T1)= Type(T2)(Using 1).

It proves the safety property.

Liveliness:

Assertion: Every process gets a chance to enter the critical section and it avoids unnecessary blocking.

Proof: In our algorithm the main process is selected according to different factors such as waiting time, priority, execution time , size and type . The process which has the maximum value will have the main token. After that this process will issue subtokens to the different processes having the same type. Whenever a new session is initiated , the priority of processes waiting in the rquest queue increases by some factor. It means that low priority processes of different types can hold the main token and enter the critical section. This proves our liveliness property.

Starvation freedom:

Assertion: Starvation occurs when one process must wait indefinitely to enter the critical section even when other processes are entering and exiting critical section. Starvation is impossible when every request in the critical section is fulfilled.

Proof: Consider the processes P_i and P_j . If the two processes are of same type, then they can enter the critical section concurrently(By using 2 and 3). If P_i and P_j are of different type, then either P_i should leave the current session or vice-versa. Also we have calculated the priority based on as waiting time, priority, execution time , size and type. When a new session is initiated , the priority of low priority processes increases by some amount. It means that after waiting for some time , low priority processes can get the chance to execute the critical section.

Concurrent entry:

Assertion: If two different processes P_i and P_j belongs to the same type, then they can enter the critical section concurrently.

Proof: From (2) , it is established that if two tokens T1 and T2 are of same session belonging to processes P_i and P_j , then their type must be same and both follows the safety property. It means that P_i and P_j can enter the critical section concurrently.'

Performance analysis:

Our algorithm uses three different messages, request ,release and token. Since this algorithm is broadcasting (n-1) request message to all the processes and (n-1) release message and atmost one token message.

Message complexity: 2(n-1)+1

Message size complexity: O(1)

Synchronization delay: The main aim here is to determine the number of message between exiting of critical section and entering of other process into the critical section. At some time, a process must leave the critical section. When the main token holder of the current session chooses a new main token, the new session starts as soon as new main token holder receives the main token from the current session. So the synchronization delay is one message hop.

Concurrency:

The maximum concurrency of this algorithm is n.

If n processes are requesting the critical section are of same type, then all the processes will receive the sub token after receiving the main token by one process. All these processes can now enter the critical section . So the maximum concurrency of this algorithm is n.

6.0 Conclusion and future work:

We have presented a hybrid group mutual algorithm based on priority. In this algorithm we have calculated the priority of processes by including different factors which are essential for the applications in group mutual exclusion. This priority is based on waiting time, execution time, age and size of the processes. In this algorithm , that process is selected from the request queue which has the maximum value calculated on the above factors. For selecting the process from request queue, the message passing concept is used. Later this algorithm uses tokens to select the different sub processes. First the main process is selected and the main token is handed over to this process. Later this process is responsible for generating the sub tokens from the different processes if the type of the processes are same. All these processes can enter the critical section. Our algorithm has achieved maximum concurrency and also have considerable message complexity.

In this case we have not considered the case of fault tolerance. Also simulation can be conducted on this algorithm. These can be considered as future work.

REFERENCES:

1. D. Agrawal and A. E. Abbadi, An efficient and fault-tolerant solution for distributed mutual exclusion, ACM Transactions on Computer Systems, 9(1), 1991, 1–20.
2. D.agarwal,A.El Abbadi,"A token baesd fault tolerant Distributed mutual exclusion algorithm,journal of parallel and distributed computing,24,pp.164-176,1995.
3. Dijkstra, E.W. Solution of a problem in concurrent programming control. Commun. ACM 1965, 8, 569.
4. G. Cao and M. Singhal, "A Delay-Optimal Quorum-Based Mutual Exclusion Algorithm for Distributed Systems", IEEE Transactions on Parallel and Distributed Systems, Vol 12, No. 12, pp. 1256-1268, Dec. 2001.
5. G. Ricart, A.K. Agrawala, An optimal algorithm for mutual exclusion in computer networks, Comm. ACM (CACM) 24 (1) (1981) 9–17.
6. Goscinski, A.M. (1990). Two Algorithms for Mutual Exclusion in Real-Time Distributed Computer Systems. J. Parallel Distrib. Comput., 9, 77-82.
7. Ichiro Suzuki and Tadao Kasami," A distributed mutual exclusion algorithm", ACM transactions on Computer Systems Vol.3,no4,pp.344-349, nov.1985.

8. M. Singhal, A taxonomy of distributed mutual exclusion, Journal of Parallel and Distributed Computing, 18(1), 1993, 94–101.
9. Neeraj Mittal and Mohan, " A priority based distributed group mutual exclusion algorithm when group access is non uniform", Journal and parallel Distributed computing Vol. 67,pp. 797-815,2007.
10. Peyman Neamatollahi, Hoda Taheri, Mahmoud Naghibzadeh, "A Distributed Token-based Scheme to Allocate Critical Resources", IEEE 2011
11. Ranganath Atreya and Neeraj Mittal, " A dynamic group mutual algorithm using surrogate quorum", in the procedding of 25th IEEE International conference on distributed computing system,2005 pp. 251-260
12. Ranganath Atreya, Neeraj Mittal, Member, IEEE Computer Society, and Sathya Peri, "A Quorum-Based Group Mutual Exclusion Algorithm for a Distributed System with Dynamic Group Set", IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 18, NO. 10, OCTOBER 2007
13. Ricart, G., Agrawala, A.: An Optimal Algorithm for Mutual Exclusion in Computer Networks. CACM, Vol. 24(1). (1981) 9–17
14. Sandeep Lodha and Ajay Kshemkalyani, A Fair Distributed Mutual Exclusion Algorithm, IEEE Transactions on Parallel and Distributed Systems, Volume 11 , Issue 6 (June 2000), Pages: 537 - 549.
15. Singhal, M.: A Dynamic Information Structure Mutual Exclusion Algorithm for Distributed System. IEEE Trans. Parallel and Distributed Systems, Vol. 3(1). (1993) 94–101
16. Y.-J. Joung, ,"The congenial talking philosophers problem in computer networks", Distributed computing Vol.15,pp 155-175,2002.
17. Y.-J. Joung, " Quorum based algorithm for group mutual exclusion", IEEE transactions on parallel and distributed system, vol 14, no. 5 pp.2003,may2003
18. Y.-J. Joung, Asynchronous group mutual exclusion, Distributed Comput. (DC) 13 (4) (2000) 189–206.