2012 International Conference on Computer Technology and Science (ICCTS 2012)
IPCSIT vol. 47 (2012) © (2012) IACSIT Press, Singapore
DOI: 10.7763/IPCSIT.2012.V47.30

Designing an Efficient Algorithm in Distributed System for Mutual Exclusion

Manvendra Singh [1], Deepali Mishra [2] and Dheeraj Pandey [2]

[1] National Institute of Technology, Durgapur, India
[2] Shri Ramswaroop Memorial College of engineering and management, Lucknow, India

Abstract. This paper presents an extension of Raymond Tree based algorithm in mutual exclusion. We have introduced the concept of time constraint, which was absent in Raymond's tree based algorithm. We have applied the time constraint to hold the token, It's provides good fairness for executing the critical section, as we have introduced the priority on basis of both FCFS and execution time. Priority is decided on the basis of execution time of each node and arrival time of each request. This gave fare chance to each node in order to access the critical section. We also maintained a global table for there priority assignment by which we can show that our proposed algorithm is efficient in terms of fair allocation of token and fair utilization of resources.

Keywords: Mutual Exclusion, FCFS, Raymond tree based algorithm, Priority.

1. Introduction

Mutual exclusion, in computer science, refers to the problem of ensuring that no two processes or thread can be in their critical section at the same time, it is a fundamental concept in the distributed system. Many different algorithms have been introduced to solve the problem of mutual exclusion and serializing concurrent accesses to a shared resource. These algorithms can be essentially divided into two groups: Non token-based or permission based like Lamport, [1] Ricart-Agrawala, Maekawa and token-based like Suzuki-Kasami , Raymond , Naimi-Trehel . Non token based algorithm are based on the principle that a node may enter critical section only after having received permission from all the other nodes (or a majority of them) present in a system. The major drawback of these algorithms is the high communication overhead like Lamport having a message complexity of 3(N-1) where N is the number of nodes present in a system. In the Token-based algorithms, a system-wide unique token is shared among all nodes, and the presence of token to a node gives it the exclusive right to enter into the critical section, thus ensuring the safety property.

In distributed system it is required that common resources must be used in mutual exclusive way. This problem is similar to mutual exclusion in a shared-memory environment but in shared memory we have atomic instructions like semaphores (e.g., test-and-set) that can be used to provide mutual exclusion [3], but such concept do not exist in a distributed environment.

In distributed environments, mutual exclusion is provided through a series of messages passed between nodes that are interested in a certain resource. Several algorithms to solve mutual exclusion for distributed systems have been developed and they can be distinguished by their approaches as token-based and non-token-based. The first group is based on broadcast protocols or they may use logical structures and data like sequence number etc. with point-to-point communication. On the other hand second group that is token-based algorithms, such as Raymond, Naimi-Trehel etc are organized in a logical tree and that a node always sends a token request to its father in the tree. Tree-based algorithms have an average lower message cost as compare to non token-based, and many of them result in a logarithmic message complexity O(logN) with regard to the number of nodes. The advantage of these algorithms is the simplicity of their local data

structures. However it suffer from one problem that it is very sensitive to node failure since it cannot tolerate even a single failure of one of the nodes in a token request path.

Our work is based on token-based algorithm [4], we have selected the Raymond tree based algorithm and we have introduced the time constraint which will be fair for all nodes present in system. Each node sends their request to the privilege node (node containing the token) with their execution time. Privilege node process the request and then make a final priority request queue.

2. Related work

Before starting our research work we had gone through the complete study of Raymond Tree based algorithm. This algorithm is a Token-based algorithm. In Raymond algorithm each node has its own holder and local queue, this node point to their immediate parents, when the token moves some nodes change their direction accordingly. All nodes point to the privilege node which holds a token, this node also acts as a root node.

2.1. Nodal Properties

I. Each node has only one parent to whom received requests are forwarded

II. Each node maintains a FFIO queue of requests

III. Each node forwards only a single request for each time that it sees the token.

Raymond's algorithm is guaranteed to be O(log n) per critical section entry if the processors are organized into a K-array tree(e.g. binary tree). Additionally, each processor needs to store at most O(log n) bits because it must track O(1) neighbors at a time.

The main problem with the Raymond's algorithm is that ,that it does not give fair chance to each node , concept of execution time is absent in the Raymond's algorithm[2] ,which reduces it's efficiency , it may cause deadlock if single node enter to the critical many numbers of time.

We had also studied another token-based algorithm Suzuki-Kasami. In this there is a completely connected network of processes. There is one token in the network and the holder of the token has the permission to enter critical section. Each process maintains an array of request where req[j] denotes the sequence number of the latest request from process j.

This algorithm is also not that much fair, it also not having the concept of time constraint. After studying all these algorithms we decided to introduce the concept of time constraint in the existing algorithm. This will give equal fair chance to each node in order to access the critical section.

3. Our Proposed work

We introduce a new algorithm to provide mutual exclusion in a distributed environment that supports priority queuing. The algorithm uses a token-passing approach that is background of Raymond's Algorithm. In our algorithm we make a FIFO queue so it will deal the things in FCFS manner, this queue is maintain at node which is containing a token, in that queue we store the least execution time required by the each node in order to execute the critical section. This queue will show that when any node request for the critical section. After this we maintain another queue that we will store the sum of execution time required by the each node and their time of arrival(exact time is not know due to absence of global clock so we take the index value of each node in FIFO queue) . For example if two node A and B sent request with their execution timeT_1 and T_2 to the privilege node that having a critical section and the arrival time A & B is t_1 and t_2. Then there new value will be T_1'=T_1+t_1 and T_2'=T_2+t_2. After this we give priority according to the value of T_1' and T_2'.

3.1. Algorithm

1: NECESSARY REQUIREMENT FOR SENDING A PARTICULAR MESSAGE IS-

HOLDER = SELF ^ ¬ USING ^ REQUEST_Q ≠

EMPTY ^ HEAD(REQUEST_Q) ≠ SELF

2: ASSIGN_PRIVELEGE:-

Step1: if HOLDER = SELF ^ ¬ USING ^
REQUEST_Q ≠ EMPTY
THEN

Step2: arrange the twodifferent queue where
$P_{New}=A_T$(arrival time) + time slot

Step3: REQUEST_Q←P_{New}

Step4: HOLDER = DEQUEUE(REQUEST_Q)

Step5:ASKED=FALSE

Step6:IF HOLDER = SELF

3: MAKE REQUEST

IF HOLDER ≠ SELF ^ REQUEST_Q ≠ EMPTY ^ ¬ASKED

THEN

SEND REQUEST TO HOLDER WITH IT'S EXECUTION TIME

ASKED = TRUE

4. Analysis

We are going to explain to explain our concept with the help of example. We consider the following tree.

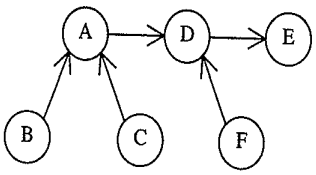

Fig. 1: Example tree.

In above example E is a privilege node. All other nodes are pointing towards the E. Each node send request to the E with their execution time. E processes the requests and forms the priority queue.
E will have following request queue:-

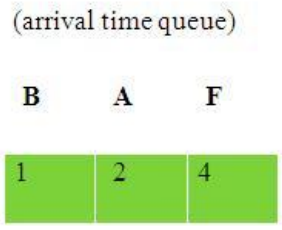

Fig. 2(a): Arrival time queue.

Fig. 2(b): Execution time queue.

Now we will make a priority queue according to the sum of arrival time queue and execution time queue. This is the final queue which is use to sending the token to the required queue.

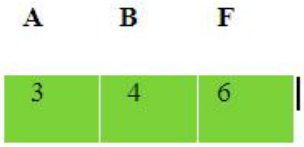

Fig. 2(c): Final for sending the token queue.

According to the above queue it is clear that now it's A chance to hold the token.

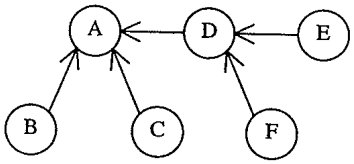

Fig. 3: privileged node.

In the above diagram node A holds the token and become the privilege node.

5. Global Table

Now it's time to maintain a global table, global table basically show the execution time of each node with their respective priority queue value and their total waiting time, so this table will be display globally as well as locally so that each node can calculate their total weighting time for the token.

Table 1: Global Table for priority assignment

Node	Time slot	P_{new}	Total waiting time
A	1	3	0
B	3	4	1
F	2	6	4

So with the help of global table each node came to know its total waiting time. This thing help the node to perform other work rather wasting its time in waiting for the token.

6. Conclusion

We can able to achieve fairness in utilizing the distributed shared resources. Concept of time constraint is efficiently applied in the existing algorithm. Our algorithm is more efficient than the existing algorithm. Concept of global table is also used which will be very useful in the distributed system. We have defined the fairness in terms of satisfying two priority criteria one is based FCFS and second is based on time slot. The proposed algorithm in our paper is better in term of efficient utilization of resource with fair access. Node can not deviate from their assigned priority because all node having knowledge about the priority of each other. So our proposed algorithm is transparent for all nodes in distributed environment.

7. References

[1] L. Lamport. "Time, clocks and ordering of events in distributed systems". Comm. ACM, 21(7):558–565, June 1978

[2] K. Raymond. "A tree-based algorithm for distributed mutual exclusion". ACM Trans. of Computer Systems, 7(1):61–77, Feb. 1989.

[3] Y. Chang. "Design of mutual exclusion algorithms for real-time distributed systems". *Journal of Information Science and Engineering*, 10:527–548, 1994.

[4] Frank Mueller **"**Prioritized Token-Based Mutual Exclusion for Distributed Systems" Humboldt-Universit¨at zu Berlin, Institut f¨ur Informatik, Berlin (Germany)