An Optimized Algorithm for Mutual Exclusion in Distributed System

Ankit Kumar Rao, Rama Shankar Sharma
Department of Computer Engineering, RTU Kota, India
ankanics2010@gmail.com, sharma_rs@yahoo.com

Abstract: In distributed systems, nodes or sites want to access a shared resource. If more than one node accesses the shared resource at the same time, it may be the problem of concurrency and replication of resource. To avoid this problem several Distributed Mutual Exclusion (DME) algorithms are used. Algorithms should take lesser message for achieving mutual exclusion. As fewer messages are required in a system, then it consumes less amount of energy. So it is important for energy conservation. This paper introduced an optimization of existing Ricart-Agrawala mutual exclusion algorithm in distributed system.

Keywords: Mutual exclusion, synchronization delay, message complexity, critical section

I. INTRODUCTION

Distributed Mutual Exclusion algorithm is used when one of the many processors or node wants to access a shared resource. Shared resource may be a shared memory or some codes or a linked list etc [3]. At a time only one node can access the resource [1]. The protocols to fulfill this criterion are called Mutual Exclusion Algorithm in Distributed system, first described in [5]. Several nodes or sites may try to access the shared resource, but at a time only one site can access the resource. This requirement is essential because if more than one site access the resource than the state of any resource will change time to time. The resources are shared in distributed system. If a site changes the structure of a resource, it should be synchronized. To follow the synchronization these mutual exclusion algorithms are used. Any node access the shared resource, this phenomenon is called entering in Critical Section (CS).

The performance of DME algorithm is measured in two ways [8]. First is that how many messages are required for a node to enter in critical section. Less the messages are required then it is a fair algorithm. Second measurement is synchronization delay i.e. how many messages are required between two critical section executions. The existing algorithm requires very high message complexity to enter in critical section. High message complexity leads to create more complexity in program, more energy consumption [7]. This paper is devoted to introduce a new algorithm that can reduce the message complexity of an existing algorithm. This Improved algorithm optimizes Ricart and Agrawala algorithm which is described in [2]. Ricart-Agrawala algorithm takes 2(n-1) messages for one Critical Section execution where n is total number of nodes in the system. The proposed algorithm uses only (n-1) message. It reduces need of n-1 messages.

II. RELATED WORK

Various researches had been made in mutual exclusion problem in distributed system. Lamport introduced first mutual exclusion algorithm in [1], which is called Lamport’s Distributed Mutual Exclusion algorithm. Also Lamport’s logical clock is introduced. All the nodes set their local time using the timestamp of receiving message. This algorithm takes total of 3(n-1) messages (n is total number of nodes) to one Critical Section execution. Synchronization delay is 1 message for this algorithm. In [2] Glenn Ricart and Ashok Agrawala optimize Lamport’s algorithm. Message complexity is reduced to 2(n-1). Synchronization delay is 1 for this algorithm. In [4] Raymond described a tree based structure for nodes. This algorithm takes O (log n) message for Critical Section execution. Suzuki-Kasami introduced a new approach for mutual exclusion in [6] where messages sequence number is used instead of timestamp.

III. STRATEGIC APPROACH

Distributed Mutual Exclusion model is defined in [7]. All the sites or nodes must be complete connected to each other. There is no fault or crash of node in the system. A failure detector is can be used if there is possibility of a crash of a site, but this is out of the scope. All the nodes are assigned unique ids from 0 to n-1. There is only one requesting process on each site or node. At any time any node can request to enter in critical section. Messages should not be lost in the system.

The approach is a modified version of Ricart and Agrawala DME algorithm. In Ricart-Agrawala algorithm it takes total 2(n-1) messages for a node to enter in critical section if total n nodes in the system. A new strategy is presented which reduced the required message to n-1. In Ricart and Agrawala algorithm a node i wants to enter in CS, sends requests to all the other nodes. On receiving a request message, a node j sends a reply immediately if it is not interested in CS. If the node j is interested in CS, it compares the priority of the incoming request with its own priority (priority is based with timestamp of message or node id), and the node with higher priority is permitted to enter in CS. Node i will enter in CS only if it gets all n-1 replies from other remaining nodes. Total 2(n-1) messages required for one CS execution.

A. Improvement of Ricart and Agrawala Algorithm:

Some assumptions are introduced in this improved
algorithm. Every node n (0,1,2,3…n-1) have their own queue and an integer variable x which value is equal to their respective node id. i.e. x= id. (x=0 for node 0, x=1 for node 1 etc). The queue is used for storing requesting messages receiving from other nodes. There are two conditions. x=node id or x! = node id. First condition occurs initially when x=node id and no CS is executed in the system. Second condition implies when at least one critical section execution is completed, this condition will be described later.

Suppose k number of nodes wants to enter in CS (n>=k). Initially all k nodes have x value equal to their respective node id’s. So all the k nodes will send a requesting message to other nodes and wait for a t time (t is maximum communication delay so that any node can send and reply the message to other node), also they will queues the receiving requests (sending by other nodes) in their queues. The message contains receiver node id, timestamp of the message, and destination message id. Also other n-k nodes when receives this requesting message, queues the request in their respective queues and wait for t time. After t time every node sort the queue according to node id or timestamp of the messages (all the nodes have same sorting elements in their queues) and top most node in queue (node i) will enter in CS. Other nodes set their x value equal to last node (node p) in queue and clear the queue. Last node p will definitely enter in CS after sometimes, so any new request will be asked to node p. Later when node i come out of CS, it informs the second node j in queue. Node j enters in CS and not need to send information message to other nodes as other nodes already set their x = last node p’s id. This step saves n-1 message of the system. If there is a new request by some node c, then node c will check its x value which is equal to node p (this is a second condition), node c request for node p. If node p is in CS currently, then it queues the request. After comes out (or it is already out of CS) it sends OK message to node c and clears its queue. Node c enters in CS and broadcast the information message to all n-1 nodes that it is entering in CS. All other nodes change their x value equal to node c. so later if any other node wants for CS, then it will have to take permission to node c. This process uses only n-1 messages for a single node to enter in CS and saves n-1 messages than Ricart-Agrawala algorithm. Synchronization delay for one CS execution is one and two messages for condition first and second respectively.

IV. SIMULATION RESULTS

Various DME algorithms have different message complexity for CS execution. Lamport’s algorithm takes 3(n-1) messages, Ricart-Agrawala algorithms takes 2(n-1) messages, but the proposed algorithm takes only (n-1) messages which saves significant amount of energy of the system. Table shown below is the result of the simulation. Total number of messages required for 3 different algorithms are calculated.

Table 1. Messages required for different algorithms to enter in critical section

<table>
<thead>
<tr>
<th>Number of nodes in the system</th>
<th>Lamport’s algorithm 3(n-1)</th>
<th>Ricart-Agrawala 2(n-1)</th>
<th>Improved algorithm 2(n-1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2500</td>
<td>7497</td>
<td>4998</td>
<td>2499</td>
</tr>
<tr>
<td>5000</td>
<td>14997</td>
<td>9998</td>
<td>4999</td>
</tr>
<tr>
<td>7500</td>
<td>22497</td>
<td>14998</td>
<td>7499</td>
</tr>
<tr>
<td>10000</td>
<td>29997</td>
<td>19998</td>
<td>9999</td>
</tr>
</tbody>
</table>

Figure 1 explains the chart comparison of different algorithms. It clearly indicates that message complexity is reduced by significant amount when more number of nodes are used. Thus it conserves more energy of the system than the other two existed algorithms.

V. CONCLUSION AND FUTURE WORK

In this paper various DME algorithm are introduced. The comparison of different algorithms is based on the basis of message complexity and synchronization delay of algorithm. A major task how to reduce the number of messages exchanged for an entry to the CS. To acquire this a new strategy is described which is optimization of Ricart-Agrawala algorithm. In the future work the node and link failure can be taken to the consideration. Moreover find out the upper bound in message complexity, reduce synchronization delay to its minimum level, and find out upper and lower bound on
communication delay in message transfer from one node to another.

VI. References


