IOSR Journal of Computer Engineering (IOSRJCE)
ISSN: 2278-0661, ISBN: 2278-8727 Volume 7, Issue 5 (Nov-Dec. 2012), PP 30-34
www.iosrjournals.org

Evaluation of Token Based Mutual Exclusion Algorithms In Distributed Systems

Ami Patel[1], Sanjay Patel[2]
[1](PG Student of Computer Engineering, Merchant Engineering College, Gujarat, India)
[2] (Assistant Professor, Department of Computer Engineering, LDRP, Gujarat, India)

Abstract: *This paper presents a framework for token based mutual exclusion algorithms in distributed systems. Their exists some traditional token based mutual exclusion algorithm. Some new algorithms are proposed in order to increase fault tolerance, minimize message complexity and decrease synchronization delay. In this paper, some new approaches are used, like Token ring algorithm with centralized approach, which is a betterment of the already existing token ring algorithm and overcome all the problems in the existing algorithm. A new token passing approach , which incurs 3 messages at high load, irrespective of no of node N and N message at low loads. Fairness algorithm for priority process, which has low message complexity and fairness in token algorithm. Several Token approach which allow simultaneous existence of several tokens. Hope the proposed framework provides a suitable context for technical and clear evaluation of existing and future methods.*

Keywords: *Critical Section, Inverted Tree Structure, Logical Ring, Mutual Exclusion, Token*

I. Introduction

The mutual exclusion problem involves the allocation of a single, non shareable resource among n processes, by means that just one process can execute in its critical section at a given time. Mutual exclusion problem was first introduced in centralized systems. In these systems mutual exclusion ensure with preserving semaphores and monitors, and one of the nodes function as a central coordinator which is fully responsible for having all the information of the system and processes ask only the coordinator for permission to enter their critical sections. But in distributed systems the decision making is distributed across the entire system and the solution to the mutual exclusion problem is far more complicated because of the lake of common shared memory and physical clock. So obtain a complete knowledge of the total system is difficult.

Lots of algorithms are proposed in distributed systems. They classified into two groups. One of them is token-based, in this group there is a unique token in the system which ensure mutual exclusion. So the requesting node must have it to enter the critical section. Another one is permission-based group, that requesting node has to ask all other nodes for their permissions to enter the critical section.

This paper is organized as follows: in section 2, presents general model of distributed system and formally describes the mutual exclusion problem. In section 3, introduces the proposed token based algorithm. Finally in last section concludes our work.

II. Model And Problem Definition

2.1. System Model

In general, most of mutual exclusion algorithms use a common model. In this model, a distributed system is a set of independent and autonomous computers. These computers are called as node or site and connected via a communication network. Each node has abstract view of whole system that communicates with message passing [1]. The most important purposes of distributed system are assigned as providing an appropriate and efficient environment for sharing resource, having an acceptable speed and high reliability and availability.

2.2. The Mutual Exclusion Problem

Mutual exclusion problem in distributed systems has received great consideration in recent 3 decades. This problem ensures that concurrent processes access common resource and data sequentially. In addition each process that executes in its critical section for a finite time, must do without interfering with other processes. Also, when no process is in a critical section, any process that request entry to its critical section must be permitted to enter without delay [2]. Eventually mutual exclusion must be without deadlock and starvation.

III. Framework For Token Based Mutual Exclusion Algorithms

By reason of the mutual exclusion importance in distributed system for keeping system consistency and increasing concurrently, various algorithms are proposed. In order to evaluate performance of these algorithms

various criteria are defined as synchronization delay and message complexity. Synchronization delay is the average time delay in granting critical section. Message complexity is the number of messages exchanged by a process per critical section entry. Also evaluates message complexity in different heavy and light load of system state.

In addition, two new measures are proposed as decision theory and nodes configuration. In the first one, if each node need not keep information about the concurrent state of the system, the algorithm will be called static. On the other hand the algorithm is called dynamic. Also, in nodes configuration if nodes are assumed to be arranged in logical configuration, the algorithm is called structural, otherwise is called non-structural.

3.1. Description of token-based approach

In this approach the right to enter a critical section is produced by a unique message, named token. The concurrent owner of the token chooses the next token owner and sends it the token. So granting the privilege to enter the critical section performs by the owner of the token. In 1985, Suzuki and Kasami [3], presented an algorithm that token by means of privilege message transmitted to requesting process based on sequence number. Some of the algorithms use a logical structure like in Raymond [4]. In this algorithm, nodes (each process performs in a node) arranged in rootless tree structure and every node is related only to its neighbours and is aware of their information. In Naimi and Trehel algorithm [5], each node sends its request only to another one that knows as a current root and waits for its permission. In addition this algorithm uses two data structures, one of them is a queue for keeping requests and the other is a logical rooted dynamic tree for assigning token. But this algorithm is so sensitive to node failure and recovery. In consequence in [6] presented a dynamic algorithm which is able to failure detection, regenerates lost token and robust against failures.

3.2. Evaluation of token-based approach

The most advantage of Token-based approach is simplicity. In this approach, for example if the logical structure of algorithm is a ring then the token transmits from a node to another continuously. Table1 represent the comparison and evaluation of well-known algorithms which mentioned in previous section according to proposed measures.

Table 1. Comparison and evaluation of token-based approach

Algorithm Name	Evaluation measures					Description
	Message complexity		Synchronization Delay	Configuration	Decision Theory	
	Heavy Load	Light Load				
Suzuki-Kasami	N	0	-	Non structural	dynamic	Token as a privilege message
Raymond with tree structure	O(log N)	O(log N)	((log N)/2)T	structural	static	Nodes in rootless tree
Naimi-Trehel	O(log N)	O(log N)	-	structural	dynamic	Use two structures :queue & logical tree
Dynamic tree	O(log N)	O(log N)	-	structural	dynamic	With failure detection & recovery

3.3. Overview of some new approach

This section describe a some new approaches for token based mutual exclusion which are basically related to Existing algorithm, with some additional features.

3.3.1Token ring algorithm to achieve mutual exclusion in distributed system – a centralized approach[7]

In this algorithm, a process can enter only one critical section when it receives the token. Consider a set of processes that are logically organized in a ring structure. Between several processes, one process acts as a coordinator. It is the coordinator's task to generate a token and circulate the token around the ring, as needed. The token can move in any direction as per the necessity. In the ring structure, every process maintains the current ring configuration of the system. If a process is removed or added into the system, then the updating must be reflected to all the processes' ring configuration table.

Suppose the processes are p0,p1,p2,...pn. Process p0 is a coordinator and process p1 wants to enter a critical section, so it send a request message. A request message contains two parameters - (1) PID and (2) Time stamp of request generation. At that time, if no other process is executing in its critical section, and the

coordinator retains the token, then the coordinator will send the token to the requesting process (p1). After acquiring the token, the process keeps the token and enters in its critical section. After exiting from the critical section the process returns the token to the coordinator.But, if some other process p_i (i != 1) is executing in its critical section, then the coordinator sends a WAIT signal, to the requesting process (p1) and the request from process p1 is stored in the REQUEST QUEUE, maintained by the coordinator only. In between, if any other processes want to enter into critical section and send request to the coordinator, then those requests will also be stored in the REQUEST QUEUE. When the coordinator gets back the token, then it sends the token to one of the waiting process in the REQUEST_QUEUE, which has smallest TIMESTAMP_OF_REQUEST_GENERATION.

3.3.2 New token passing distributed mutual exclusion algorithm [8]

In this algorithm, No assumptions are made with respect to the network topology and the communication medium. The token is passed from node to node, and only the token-holder is permitted to enter its critical section.

In this algorithm, the token contains an ordered list or queue (referred to as the Q-list) of all the nodes that have been scheduled to execute their critical section. The token is passed from node to node in the order specified by the Q-list. The node which executes the critical section is the current node at the head of the Q-list. The Q-list is created by a node currently designated as the arbiter of the system. Every node in the system keeps track of the arbiter by setting a parameter called ARBITER. An arbiter node executes the following two phases: Request Collection Phase and Request Forwarding Phase.During the request collection phase, the arbiter node collects all the requests from the nodes that are seeking access to their critical sections, and creates the ordered Q-list. At the end of the request collection phase, when the arbiter gets the token in its possession, the token is updated with the newly constructed Q-list and transmitted to the node that is at the head of Q-list, along with the contents of Q-list. Further, the arbiter node declares the node of the last request in the Q-list as the new arbiter, by sending a broadcast message to all the nodes in the system. A node, on receipt of the message electing it as the new arbiter, enters the request collection phase. The token, which is passed from node to node according to the Q-list, finally reaches the new arbiter node. It is possible however, that a node sends its request to the previous arbiter before a message informing it of the current arbiter arrives. In this case, the request message needs to be forwarded to the current arbiter. Thus, an arbiter enters the request forwarding phase after the request collection phase in order to forward requests that did not arrive during the request collection phase, but were transmitted before the identity of the current arbiter was received at the requesting nodes. Any requests arriving after the end of the forwarding phase, are dropped.

3.3.3 FAPP: A new fairness algorithm for priority process mutual exclusion in distributed systems [9]

The fairness algorithm for priority processes (FAPP) assumes an inverted tree structure as the logical topology where competing processes form vertices of the tree. Priority may be associated with each process, and a process is allowed to enter into its CS only when no higher priority process is pending. It allocates the token to the requesting process with the highest priority. Equal-priority processes are served following the FCFS order to ensure fairness of allocation. The algorithm uses Dynamic Process Priority, so the property of liveness is not violated. The process P_{hold} that is holding the token at any instance is in the root of the tree. The root changes each time the token is transferred to some other process. The new node that gets the token is designated as P_{hold} and forms the new root of the tree. This algorithm is based on two rules:

Rule 1: When a node A finds a token request from another process B with priority r then for all such process C in PL_A (priority list of A) whose priority t is less than r, the process priority of C is increased by 1 to t +1.Rule 2: After a process gets the token it enters the critical section. As it comes out of the critical section, the priority of a process is reset to its original value.

Working Process:

- When a process M with priority {r} wants to enter the critical section (CS) ,then insert <M, r> in Request Queue PL_M with respect to priority assigned with process M and token request <M, r> to its parent process.
- When a process $S \neq P_{hold}$ receives a token request <K, r>and priority {r} from another process K, and Priority of K, i.e., r, is higher than the priority t for some process C whose token request is sent through S. Priority value t is increased by 1 i.e., t = t+1 and then, insert <K, r > in PL_S in the sorted order of descending priority and send token request(S, r).
- When a process J = P_{hold} receives a token request <K, r>and priority {r} from another K, there is a pending request from P_k, replace old priority of K with r. I the priority r is greater than other pending process's priority(t), then Priority value t is increased by 1. then, insert <K, r > in priority queue in the sorted order of descending priority;

- On completing the execution of a CS, Ej=P_{hold} remove the first process tuple <M, m> from PL_E ,send token(M, m) and M would be the new P_{hold} - the token is passed to M from E. If priority queue of process E is not null, then send dummy token request(E, x) means E places a token request to M along with the highest priority which E receives.
- The newly designated root process M = P_{hold} that receives token (G, r) ,remove the first tuple <G, r > from PL_M , after executing CS, process works with its original initial priority and if $PL_M \neq$ null send dummy token request <M, x>.

The process may stop at any intermediate node Pk, if there is already a pending request for Pj whose priority is higher than that of Pi.

3.3.4 Several tokens distributed mutual exclusion algorithm in a logical ring network[10]

This algorithm is based on the token ring and allows simultaneous existence of multiples tokens in the logical ring of the network. Each of the competing nodes generates a token for the permission to enter the CS. The token traverses the logical ring structure. The process can only enter the critical section if it gets back its own token. Each node maintains separate queues for storing different CS entry requests and thereby allows the existence and evolutions of multiple CSs at the same time Every node maintains separate queues for each requested resource type in its local memory. The token generated by a node moves forward either clockwise and anticlockwise. Each token is uniquely identified by its request TokenID and ResourceID, where TokenID also works as a priority identifier and is defined as TokenID= (SeqNum, PID). The sequence number, SeqNum, is a locally assigned unique sequence number to the request token and PID is the process identifier. Each node maintains the highest sequence number seen so far in a local variable *HSeqNumSeen.* When a node makes a request for the CS, it uses a sequence number which is one more than the value of *HSeqNumSeen.* When a token is received, *HSeqNumSeen* is updated us follows:

HSeqNumSeen=max (HSeqNumSeen, SN), where, *SN* is the sequence number in the received token.

- When a node Ni wants to enter the critical section, CS(r), and generates a token, makes a copy of the token in its node and passes it to the next node.
- Any node Nj receives a token for CS(r), and reacts to it in the following ways:
- If Nj has no intention to enter the CS(r), it replaces the sequence number by HSeqNumSeen and simply passes the token to the next node.
- if Nj is now in the CS(r), it puts the incoming token in queue and when Nj exits CS, it releases the tokens to the next node sequentially (if any) from queue.
- if Nj has already generated a token but not yet received that back, it compares the incoming token's priority with it's generated token's priority. if priority of received token is higher then, it passes the token to the next node; otherwise it puts the token in queue.
- If the node Ni receives the token, that is all other nodes allowed Ni to enter CS(r), and then it enters the CS. On exiting from the CS(r) it sends the queued tokens to the next node sequentially (if any) and deletes the associated copy and original token.

The comparative evaluation of different parameters of algorithms described in section 3.3 are shown in Table 2.

Table 2. Comparative evaluation of different parameters of algorithms

Algorithm Name	Network Topology	No of Messages	Data Structure	Waiting Time	Fairness and Prioritized Access	Features
Token Ring with centralised approach	Logical Ring	3(Request-grant-Release) 4(Request-Wait-Grant-Release)	Request Queue at central node	0 to n-1	Fairness is achieved and based on FCFS	Does not allow the circulation of the token along the ring when there is no need
New Token Passing Algorithm	--	3 Messages Privilege, Request and New-arbiter message	Queue list is created by arbiter node	--	Fairness and partial prioritized access	Two parameters: Request collection phase and request forwarding phase
FAPP	Inverted tree structure	Request Control Message and Token Transfer Control message	Request queue at each node	O(log N) N is a no of nodes in a balanced binary tree	Fairness and Prioritized access	Use more control messages
Several Token Algorithm	Logical Ring	n-1	Priority queue for resources at each node	n	Fairness is achieved	Simultaneous existence of several tokens

IV. Conclusion

After evaluation of various algorithms, we find a new approach, which we are going to implement. Our new approach will be based on logical ring topology. In this approach, One process is selected as a coordinator among logical ring. Initially every process has assigned a process-id and priority which is equal to process id.

Coordinator maintains a resource queue for every resources and multiple tokens which are equal to no of resources in the system. When any process want to execute a critical section(CS), then it sends a request message with a parameter(Resource name, Process id, Priority) to the coordinator. If any other process is not executing that critical section and coordinator process has token, then coordinator process sends the token to the requesting node. otherwise, coordinator process store the request message in a resource queue with two field: Process id and Priority. At that time, another process wants to execute a same CS, coordinator process also save the request message in a queue. Coordinator process checks the priority of the both process. If the priority of earliest request message is low than the next message, than increment the priority by one and then sort the queue.Such a way, this type of approach contains multiple existence of token at central coordinator and also contain dynamic priority for a requested process. At the time of implementation, we consider no of message per CS entry, waiting time and system throughput as a parameters.

References

[1] W. Stallings, "Operating Systems Internals and Design Principles", Prentice Hall, pp.205-261(2009).
[2] M.G.Velaquez, "A Survey Of Distributed Mutual Exclusion Algorithms", Technical Report CS. Colarido State University, September 1993.
[3] Ichiro Suzuki and Tadao Kasami, "A Distributed Mutual Exclusion Algorithm", ACM Transactions on Computer Systems, Vol. 3, No. 4, pp. 344-349, November 1985.
[4] Kerry Raymond, "A Tree-Based Algorithm for Distributed Mutual Exclusion", ACM Transactions on Computer Systems, Vol. 7, No. 1, pp. 61-77, February 1989.
[5] Mohamed NAIMI, Michel TREHEL, André ARNOLD, "A LOG (N) Distributed Mutual Exclusion Based On The Path Reversal", Journal Of Parallel And Distributed Computing, 34(1), pp. 1-13, April 1996.
[6] J. Sopena, L. Arantes, M. Bertier and Pierre Sens, "A Fault-tolerant Token based Mutual Exclusion Algorithm Using A Dynamic Tree", 2005
[7] Sandipan Basu, "Token Ring Algorithm To Achieve Mutual Exclusion In Distributed System – A Centralized Approach", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 1, January 2011.
[8] Sujata Banerjee and Panos K. Chrysanthis, "A New Token Passing Distributed Mutual Exclusion Algorithm", In Proceedings of the Intl. Conf. on Distributed Computing Systems (ICDCS), 1996.
[9] Sukendu Kanrar and Nabendu Chaki, "FAPP: A New Fairness Algorithm for Priority Process Mutual Exclusion in Distributed Systems", Journal Of Networks, VOL. 5, NO. 1, January 2010.
[10] Ousmane THIARE, "Several Several Tokens Distributed Mutual Exclusion Algorithm in a Logical Ring Network", 2009 international Conference on Machine Learning and Computing IPCSIT vol. 3, 2011.