International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 4 Issue 09, September-2015

Literature Survey on Distributed Mutual Exclusion Algorithm for Mobile Ad-Hoc Network

Prabhjot Singh
Department of ECE,
CGC-COE, LANDRAN,
Punjab

Namita Naag
Assistant Professor,
Department of ECE,
CGC-COE, LANDRAN,
Punjab

***Abstract*- In a distributed system several devices are connected to each other to share resources like software's or hardware's, which makes Mutual Exclusion essential for distributed system during the sharing process. The region where all the sharing takes place is the Critical region. Mutual Exclusion on Distributed system allows critical resource to be shared between different mobile nodes in a Mobile Ad-Hoc Network. In this the region is divided and in a logical way sharing is done. Critical section can be defined as a state when a node or device is actually sharing or using the resources which are required in the process of communicating with each other. A request is involved for making decision regarding that which node or device would enter critical section. This request can be made to neighboring nodes or by imposing any algorithm which can make it possible the sharing process to be fair for each node i.e. no node should wait for an infinite time for its turn or no node should get infinite access to resources. In this paper, several mutual exclusion algorithms are discussed by comparing their advantages and disadvantages. Section 1 includes the introduction of the Distributed Mutual Exclusion problem and its categories. In Section 2 the basic requirements for a system to attain mutual exclusion. In section 3 discussions of all the algorithms is covered.**

Keywords: Distributed Mutual Exclusion (DME), Critical Section, Mobile Ad-Hoc Networks (MANET).

I. INTRODUCTION

The problem with mutual exclusion which occurs is when some two or more processes or devices try to work at same instant of time simultaneously [1]. When they compete for the critical section at the same time, no one of the devices get chance to share resources, and none of them can't use it fruitfully [2, 3, 4].So, to prevent from this issue, a distributed algorithms is designed to manage the critical region [5].
Critical section (CS) is a code in which sharing of resources is done or accessed. Now practically sharing common resources simultaneously is not possible and cannot be synchronized for sharing resources. So, if two nodes try to access the critical section can lead to crisis. Mutual exclusion is to ensure that at a time only one of the concurrent processes are allowed to access the common or shared resources at an instant of time. In case of distributed systems, where multiple sites are considered it is named as distributed mutual exclusion (DME).

MANETs has no restrictions when it comes to topology and the sites are free to move within a region i.e. the critical region. This free moving of the sites in the region can generate link failure which is a main issue for us discussing Mutual Exclusion. Also the nodes use the battery power for the processing which makes them totally relay on those batteries. This is the major problem which is observed in MANETs than static networks. Another factor which has to be considered is to satisfy the Combinatorial Stability. It is a state when a node is waiting for making a decision of whether to enter critical section or not, at the same time the underlying topology of the network can change. So, the processing time of the algorithm should be fast enough that a change in the network layout does not affect the objective of the algorithm. It is a concern when it comes to token-based DME algorithms or permission-based algorithms [8].

Distributed Mutual Exclusion Algorithm can be classified into two major categories: Token-based algorithm and Non token-based/Permission-based algorithm. Token-based algorithm depends on the site entering to critical section by accessing a token. Further it can be classified as circulating and requesting method. In the circulating method a token is passed among all the participating sites and the site which possess the token gets the chance to enter the critical section and after that it releases the token back in the circulation. The other method is requesting one in which it a site requests the other participants for entering into the critical section. Suzuki-kasami's algorithm is an example of token-based mutual exclusion algorithm for distributed systems. In the case of Permission-based a site desiring to enter to critical section must first get permission from all the sites before it enters in the critical section or from some nodes, it varies according to the algorithm. Lamport, Ricart-Agrawala,Roucairol and Carvalho's algorithm are some examples of Non token-based mutual exclusion algorithms for distributed systems.

Distributed Mutual Exclusion Algorithm can also be classified by three approaches named: Token-based approach, Non Token-based approach and Quorum based approach [9]. Below are listed the main requirements for mutual exclusion algorithm.

II. REQUIREMENTS

Main requirement for a mutual exclusion algorithm is that only one site at a time can execute the critical section and execution of critical section by two sites simultaneously is not possible. Besides this there are some other requirements essential for distributed mutual exclusion algorithm to have [9]:

1) Safety: No two sites must be allowed entering in critical section at same time.
2) Freedom from deadlocks: Mutual exclusion should be free from deadlocks. Site entering into critical section should release it in a finite time so, that a fair chance is given to all other sites.
3) Freedom from starvation: If a site request for entering into critical section, it should not be forced to wait for an infinite time for critical section i.e. every site should get a chance to execute the critical section.
4) Fairness: A site requesting for entering into critical section must get a chance to enter critical section regardless of any seniority list or first-come-first-service (FCFS). This can be ensured by using queue in the algorithm.
5) Fault-correctness and Fault-Tolerance: If during a process a state of failure i.e. a node is dead which can be by losing its battery power then there should be measure for controlling or correcting so that the process keeps on working [8, 9].

III. DISTRIBUTED MUTUAL EXCLUSION ALGORITHM

Lamport [10] gave the idea of distributed algorithm for mutual exclusion systems in 1978 [7]. His idea was to make use of timestamp. Timestamp is the time at which a request for entering in critical section is made by a node. Based on the idea, each site which is going to critical section has to send a message involving a timestamp and it's Id (A unique name given to each node in the critical region). So, when a site enters into critical section by sending a request message to all the sites and waits for reply message from all the sites and when it releases the critical section it has to send a message to inform all the other sites that they can access critical section. Also a site has to maintain a queue of requests as when a site receives more requests it adds the later request in to request queue or the one which has less priority than the other one and sees to it when the earlier request is processed. But if a situation occurs when two sites in a critical region ask for critical section then it makes decision on considering the timestamp in the message generated by the two sites. In this the site which has the lower timestamp gets access to critical section and those having higher timestamp have to wait for a finite time for their turn [6]. In this election is distributed between all sites, thus the damage point is absent, also the mutual exclusion is totally work out, but due to extra messages it makes traffic when a number of requests are flowing. This algorithm creates $3(N-1)$ messages per request, $(N-1)$ total number of requests, $(N-1)$ total number of replies, $(N-1)$ total number of releases.

Ricart-Agrawal [10] proposed an optimized algorithm of Lamport's algorithm and it was a permission-based algorithm for MANETs. Ricart-Agrawala used two messages *REQUEST* and *REPLY* message from site to attain and to release the critical section that dispenses the release message with replay message. In this a node can enter in critical section only after notifying all the other nodes. A node makes an attempt by sending a Time stamped *REQUEST* message to all the other nodes upon which it gets an *REPLY* message immediately or it has to wait for a message to enter in critical section. If any of the node has smaller timestamp in comparison to the requesting node then it will process it own request and after that the requesting node would be given possession to critical section. This algorithm creates about 2 (N - 1) messages per request, where (N - 1) are the REQUEST messages and (N – 1) are the REPLY messages i.e. permission messages. Here N is the number sites [8, 10].

Roucairol and Carvalho [17] proposed an improvement to the Ricart-Agrawala algorithm. In which they stated that if a site has received a REPLY message from another site, then the site can use the critical section till it sends the REPLY message to any other site i.e. the site can use the critical section for as many time it want and the possible condition for other to use it is when they send REQUEST message to the site for having possession over critical section or the site itself releases it. Now this could last for a single round to a number of times if no other site is requesting for critical section. With this change in the algorithm of Ricart-Agrawala a site requesting for critical section by their algorithm the number of REQUEST messages required to access the critical section are can be 0 or 2 (N - 1). By this if a site is possesses the critical section it need not request again to any other site for using it again but on the other hand it increases the condition of starvation for other sites which could be requesting for critical section but have to wait for a finite period or more time for their turn. This algorithm also not gives fairness to all sites as it violates the basic requirement of a distributed system. A site can attain the critical section for an infinite time as it gets permission for using the critical section from all of the other sites. But, it cannot be implemented as it removes starvation for one site but adds for other sites in the critical region.

Suzuki and Kasami's reduced the number of minimum messages which are required to attain the critical section. Suzuki and Kasami's algorithm [16] reduced it to N (number of sites) number of messages in comparison to the Ricart-Agrawala Algorithm. It is a token-based algorithm in which mutual exclusion is achieved by maintaining a token among all nodes for entering the critical section. In this if a node sends a request to another node it sends its identification and sequence number along with its request. So, if a node has the token then only it can enter the critical section and the status to other nodes is busy. In this every node maintains a list of the sequence number and the request order by all the other nodes. The other node reply only if the first node has the priority more than it has or it holds the request until it fulfils all the other requests or its own use. They came with an idea of a single PRIVILEGE message and it is always the first node has the privilege and a node having the privilege can enter the critical section repeatedly till it passes it to another node and if there is no request for critical section by any site, then the one which possess the privilege last retains the privilege till a request is generated. Also the allocation of the critical section is done by first-come-first-served (FCFS) manner which makes it necessary to maintain a queue which holds the requests for the critical section. Unlike Ricart-Agrawala's algorithm in which a request has to complete a round-trip to all node in communicating with them. They stated their algorithm to be deadlock free and starvation free. In addition to that a list is maintained which has the number of requests that are being processed i.e. fulfilled for each node. So, we can say that message complexity is of exactly N for N Processes. Or it is zero in case when the node already has the privilege message with it.

The algorithms which came after Ricart-Agrawala algorithm were not able to remove mutual exclusion effectively but Meaekawa [11] proposed one which uses the 'quorum' for making the decision regarding taking permission for using the critical section than taking permission from each site, it was called as quorum-based distributed mutual exclusion algorithm. In this it was proposed that the nodes which are participation in the process are in the quorum. The use of voting technique by Thomas [12] is based on a majority attained by a node and requires that a node requesting mutual exclusion obtain a permission vote from only a majority of the nodes regardless all the nodes in the critical region. Thus, in the best case, we can say the number of permission messages required to obtain mutual exclusion is reduced to a half i.e. N/2. Thus if a node want to invoke to mutual exclusion a REQUEST has to be send and get permission from only the members of the quorum. By this the steps involved in communicating with all the nodes is removed. So, when REQUEST is sent it is verified by the quorum and if no other participant is interested in using the resources then it is allotted critical section. But when it comes to MANETs it adds up the steps required for selection of the quorum, when compared to its benefits. The message complexity is of 3√N messages per mutual exclusion, √N messages to convey a request, √N messages to obtain permissions, and √N messages to release mutual exclusion

The algorithm proposed by Ricar-Agrawala [10] it requires 2 (N - 1) messages exchange for getting entry to critical section, while in Suzuki and Kasami [16] they reduced it to just N messages. Then Maekawa [11] improved the algorithm and reduced the minimum messages to O$\sqrt{N}$ (O - quorum). After that Raymond and Kerry [19] came with algorithm which enhanced the performance by using a tree topology and reduced the message required to O (log N). Raymond's algorithm is a lock-based algorithm on a distributed system for mutual exclusion. It requires a logical structure of a K-ary tree on distributed resources. As defined, each node has only a single parent, to which all requests to attain the critical section are made. In this algorithm each node has to communicate only with its neighbor node in the tree structure and only holds the information of its neighbor node only. In this each node has a parent node to which it sends its entire request and each node maintains a FIFO queue for recording the list of the request sequence. So if a node has to forward the privilege to another node it will first check the queue and if it is on the top then it will forward the privilege to that node and delete the first entry in the queue and in other state it will add it to the queue. If a node itself wants the critical section and it has requests pending in its queue it would place itself in its own queue. They guaranteed that to enter critical section *O (log N)* messages are required if we organized the nodes into a *K-ary* tree. In addition, each site needs to store at most *O (log N)* bits because it must track *O* (1) neighbors.

Then Singhal, el al. [13] came with an idea that if a site is not competing for critical section then is it necessary to take permission from that site? So, they observed that a site need not consult other sites that are not currently in a need of critical section. They introduced an idea of 'look-ahead technique' in which before sending REQUEST message a site identifies that which or how many sites are concurrently competing for critical section and then enforcing mutual exclusion on those sites which are competing rather than to all the sites. The benefits of this algorithm were it saved the resources as well as to low the message overhead which makes it more suitable for mobile systems. It also omitted the site requesting or replying to all sites if it is not even participating in the resource sharing practically. The motivation behind this was the observations i.e. a site need not consult sites which are not contenting for the critical section. A site is only required to consult only those sites which are currently competing for the Critical Section. Even the traffic depends upon the number of active sites in a process in a certain critical region and non participating sites have no request overhead as requesting sites are not sending any messages to non participating sites. They also introduced the concept of dynamic distributed algorithm [14]. It used the dynamic Information sets, Info_set and Status_set, to keep track of sites that are currently involved in critical section or waiting for critical section. The message complexity for a request is 2 (Φ-1) where Φ is the sites which are competing for accessing Critical Section.
As singhal el al. introduced a 'Look Ahead Technique' which minimized the message overhead by interaction between only those sites which are competing for critical section. Along with these Wu et al. [15] introduced three new messages DOZE, DISCONNECT and RECONNECT. They also introduced FIFO (First-In-First-Out) service which was not feasible in case of MANETs as the sites in region have no fixed locations or topology to follow. The important problems which are to be considered in case of tolerance is link failure and host failure which is the very frequent in MANETs. If we use timeout and retransmission of *REQUEST* message then this link failure and host failure can be removed or minimized. By using Info_set and Status_set we can handle the doze and disconnection mode.

When a site wants to enter “DOZE” mode, it broadcast a *DOZE* message to all the sites in its Status_set and Info_set, and all the other sites moves the site to no REQUEST zone in which no message is send to that site. When the site wakes up it can resume the algorithm without any special algorithm. The benefit of DOZE mode is that the site can save its battery power when it does not want to use the network resources for sending or receiving. When a site wants to disconnect from the critical region, it can simply generate a *DISCONNECT* message, rather than *DOZE* message. When the site want to reconnect it can inform all the sites in the region by sending a *RECONNECT* message to inform all the sites. Last is the measure which is used for handling the timeout of the REQUEST message, in this fault tolerance is achieved. When a site generates a request message it adds a timeout with the message i.e. TO_{REQ} which expires when the reply for the request is not replied or the reply not reaches the site. By this an infinite waiting by a node for reply from other sites can be resolved i.e. starvation.

IV. CONCLUSIONS

This paper concludes, all the algorithms being used for Distributed Mutual Exclusion. However there are many difficulties for attaining mutual exclusion among all the nodes, so still research is being carried out. A summarized comparing of these algorithms in which we will refer to the advantages, disadvantages and the number of message by each site or message complexity for gaining the critical section is required in these algorithms.

Algorithm	Advantage	Disadvantage	No. of Messages
Lamport	1.Absolved the Mutual Exclusion 2.There is no starvation	1. Increase in number of messages 2. Traffic increased	3 (N-1)
Ricart-Agrawala	1 Absolved the Mutual Exclusion 2. Reduced the number of messages.	1. Increase in number of messages 2. if a node fails starvation can occur	2 (N-1)
Roucairol and Carvalho	1 Absolved the Mutual Exclusion 2. Reduced the number of messages	1. Increase in starvation 2. No fairness as a node can use CS for infinite time	2 (N-1) or 0
Suzuki-kasami	1 Absolved the Mutual Exclusion 2. Reduced the number of messages	1. Increase in starvation 2. If a node has Privilege it can use the CS infinite times and privilege is forwarded in a fixed way than Dynamically	N or 0
Maekawa	1 Absolved the Mutual Exclusion 2. Only the nodes in quorum participates and no request is send to non quorum nodes	1. deadlocks possible 2. Additional steps required for making the quorum. 3. Complex method in case of MANETs	$3\sqrt{N}$
Raymonds	1. Follows a K-ary type of structure. 2. Reduced the number of messages	1. Nodes need to maintain a queue of requests. 2. Waiting time increases.	O (log N)
Singhal et al.	1 Absolved the Mutual Exclusion 2. Sites who require CS should only be considered	1. Reduced the number of requests 2. But increase in complexity when MANETs are considered	2 (Φ-1) Φ are the sites competing for CS

REFERENCE

[1] Ye-In Chang, A hybrid distributed mutual exclusion algorithm, Microprocessing and Microprogramming, Volume 41, Issue 10, June 1996, Pages 715-731.

[2] Hoda Taheri, Peyman Neamatollahi, Mahmoud Naghibzadeh, A hybrid token-based distributed mutual exclusion algorithm using wraparound two-dimensional array logical topology, Information Processing Letters, Volume 111, Issue 17, 15 September 2011, Pages 841-847.

[3] Wim H. Hesselink, Alex A. Aravind, Queue based mutual exclusion with linearly bounded overtaking, Science of Computer Programming, Volume 76, Issue 7, 1 July 2011, Pages 542-554.

[4] Wang Zheng, Liu Xin song, Li Meian, Ad hoc distributed mutual exclusion algorithm based on token-asking, Journal of Systems Engineering and Electronics, Volume 18, Issue 2, 2007, Pages 398-406.

[5] Weigang Wu, Jiannong Cao, Jin Yang, A fault tolerant mutual exclusion algorithm for mobile ad hoc networks, Pervasive and Mobile Computing, Volume 4, Issue 1, February 2008, Pages 139-160.

[6] Yixin Chen, Ruoyun Huang, Zhao Xing, Weixiong Zhang, Long-distance mutual exclusion For planning, Artificial Intelligence, Volume 173, Issue 2, February 2009, Pages 365-391.

[7] Hossein Nick Khah et.al, SURVEY OF MUTUAL EXCLUSION ALGORITHMS, International Journal of Computer and Electronics Research [Volume 2, Issue 6, December 2013]

[8] Parameswaran, Murali, and Chittaranjan Hota. "A novel permission-based reliable distributed mutual exclusion algorithm for manets." Wireless And Optical Communications Networks (WOCN), 2010 Seventh International Conference On. IEEE, 2010.

[9] Singhal, Mukesh, and Niranjan G. Shivaratri. *Advanced concepts in operating systems*. McGraw-Hill, Inc., 1994.

[10] Ricart, Glenn, and Ashok K. Agrawala. "An optimal algorithm for mutual exclusion in computer networks." *Communications of the ACM* 24.1 (1981): 9-17.

[11] MAEKAWA, MAMORU. "A dA/Algorithm for Mutual Exclusion in Decentralized Systems." *ACM Transactions on Computer Systems* 3.2 (1985): 145-159.

[12] Thomas, Robert H. "A majority consensus approach to concurrency control for multiple copy databases." *ACM Transactions on Database Systems (TODS)*4.2 (1979): 180-209.

[13] Singhal, Mukesh, and D. Manivannan. "A distributed mutual exclusion algorithm for mobile computing environments." *Intelligent Information Systems, 1997. IIS'97. Proceedings*. IEEE, 1997.

[14] Singhal, Mukesh. "A dynamic information-structure mutual exclusion algorithm for distributed systems." *Distributed Computing Systems, 1989., 9th International Conference on*. IEEE, 1989.

[15] Wu, Weigang, Jiannong Cao, and Jin Yang. "A fault tolerant mutual exclusion algorithm for mobile ad hoc networks." *Pervasive and Mobile Computing* 4.1 (2008): 139-160.

[16] Suzuki, Ichiro, and Tadao Kasami. "A distributed mutual exclusion algorithm."*ACM Transactions on Computer Systems (TOCS)* 3.4 (1985): 344-349.

[17] Carvalho O.S.F. and G. Roucairol, "On Mutual Exclusion in Computer Science, Technical Correspondance," Journal of ACM, 1985.

[18] Chen, Yu, and Jennifer L. Welch. "Self-stabilizing dynamic mutual exclusion for mobile ad hoc networks." *Journal of Parallel and Distributed Computing* 65.9 (2005): 1072-1089.

[19] Raymond, Kerry. "A tree-based algorithm for distributed mutual exclusion."*ACM Transactions on Computer Systems (TOCS)* 7.1 (1989): 61-77.