Dealing with Mobile Ad-Hoc Networks with Optimized Permission Based Mutual Exclusion Algorithm

Sanjida Singhani
Department of CSE CU Gharuan, India.

Vikas Wasson
Asst Prof CSE CU Gharuan, India.

Abstract –The paper presents a proposal for the optimized total number of messages exchanged in permission based algorithm. In order to achieve it, a new message known as *Hold* message has been used. Additionally, a *timestamp priority* has been utilized in order to reduce the number of sites to which hold message is sent. With this, the parameter, *t*n,* will get optimized reducing the total number of messages exchanged in the proposed algorithm. The paper also aims at discussing the correctness proofs for static analysis of the algorithm. Umpteen DME protocols have been designed and implemented in order to achieve the optimization of various DME parameters, *viz.*, liveness, fairness, message complexity and safety. Most of the previous approaches lacked message complexity to which our proposal suggests to achieve optimized total number of exchanged messages.

Index Terms – Distributed Mutual Exclusion, Mobile ad-hoc network, Critical section, Message complexity

1. INTRODUCTION

A Mobile Ad hoc network (MANET) is characterized by self configuring infrastructureless networks of mobile devices linked via wireless channel [1]. Based on this feature, it is also termed as an autonomous system of mobile nodes and associated hosts which collectively form an arbitrary and dynamic topology [2]. Basically, processes must share common hardware or software resources that assist each other to work independently at large scale, in distributed mobile ad-hoc networks [3]. Further, the access to a shared resource must be synchronized on account to make certain that, at any given time, only one process employs the live and available resources. Each process has a code segment, critical section (CS), meant for accessing the shared resource [1]. Hence; managing and coordinating the execution of critical section is a big issue, needs concern, in ad-hoc networks. By utilizing a finite-time *mutually exclusive* access, by the CS, this issue can be resolved. In addition, each node must request permission to enter its CS and release the same after exiting.

The existing literature describes, two main approaches that have been proposed for solving the DME problem, namely, centralized and distributed [2, 3]. In the former approach, one node is selected to act as a central coordinator. Further, this node is made fully responsible for storing the complete information of incoming requests coupled with the information of available resources in order to make the best use of the shared resource.

On the other hand, in later approach, the decision-making is spanned over the entire system. To accomplish the task of achieving DME using distributed approach, the two principles as follows:

- Token- based algorithms named so because of the presence of token in the system
- Permission-based algorithm attributed so because of the collection of permission from nodes in the system.

1.1Token-bsased Approach

In token-based approach, there are two methods of using token for entering into CS. The first method states that only one process can enter CS by using a special object called token, which is unique to the whole system. Here, token acts as a privilege to that process for entering the CS [2].A process, the current owner of the token, selects the next token owner by making use of priority as the base for selection criteria. In a particular scenario where no process wants to enter the CS, then the token is held by the current process itself.

On the other side; the second method that has been followed suggests that the processes are logically arranged and organized in a specific ring topology where the token is circulated from one process to another, providing them access to enter into the CS [2]. After it exits from its CS, the token is released for further circulation. However, in a case where the process is not interested to enter CS then it passes the token to the next node in the logical ring. Also, Starvation Freedom is guaranteed if the ring is unidirectional.

1.2 Permission-based approach

In the permission-based approach, the process can be allowed to enter CS only by explicitly acquiring permission from a set of nodes or from all nodes, in the system. There is no requirement of token, therefore attributed as non token based-approach. A priority in the form of logical clocks or timestamps can be established for incoming requests [3]. When a node completes its execution and exits from CS, it informs all other nodes from which it had received permission. Permission-based approach is further divided into two types based on, *i.e.*, *voting* and *coterie* [7]. In voting-based approach, each node is assigned with a vote, in the system itself. Therefore, a node that wants to execute CS, asks for permission from those nodes that constitute the majority of votes. On the flip side, algorithms that follow coterie-based approach possess a coterie, collection of quorums (*i.e.*, set of nodes), is attached to the system. Therefore; a node must obtain permission from each and every node of quorum present in the coterie in order to access the CS [7]. Figure 1 (given below) describes the flow of mutual exclusion algorithms.

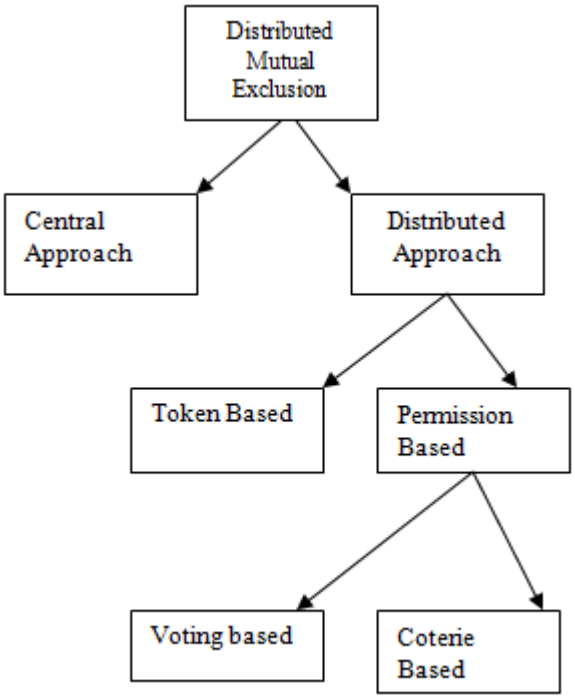


Figure 1: Flow diagram of DME approaches

The main objective of our proposed algorithm is the reduction in total number of exchanged messages along with the new message called Hold message. In addition, it ensures deadlock freedom. The propose technique specifically works on the reduction of number of sites to which a node has to send "*Hold*" message by applying timestamp priority which will further optimize the reduced flow of messages in the system

2. RELATED WORK

The first proposed solution for distributed permission-based mutual exclusion problem by Lamport in 1978, popularly known as Lamport's algorithm, used three categories of messages: *request*, *reply* and *release*. In order to serve the request messages, it uses the concept of logical clocks and assigns sequence numbers to the incoming request *i.e.*, *timestamp* [4]. Thereafter, every node maintains a queue of pending requests for entering into the CS. A node, n_i , when wants to execute CS, a message is broadcasted to all other nodes and its corresponding request is stored in a local queue. Further, upon receiving the request message from n_i and after storing the message in its own queue, node n_j , sends back a timestamped reply message. However, n_i can access CS only when two conditions are met: First, the received reply messages from all other processes must have timestamps greater than its own timestamped request. Second, the nodes' own request must be kept at the front of its queue. A release message broadcast is followed by the exit of n_i from the CS. Therefore; the message complexity of this algorithm is 3(N-1). Ricart and Agrawala (RA) improved Lamport's solution by reducing message complexity from 3(N-1) to 2(N-1). This algorithm utilises two messages, request and reply, thereby, avoiding the release messages [5]. In case, each node, either in CS or requesting CS, has a higher priority request, would not send the reply messages. A node enters CS only after receiving permission from all nodes and upon exiting the CS, it sends all reply messages that have been deferred. Maekawa's algorithm introduced the concept of coterie by associating each node with a set of nodes. In its design, there is always a node in the intersection of two subsets [7]. A node n_i must obtain permission from all other nodes in its home set, S_i, before it can enter its CS. After exiting CS, it replies to node at the top of requesting queue instead of sending message to all nodes in the queue. The number of messages required to handle a request is 3 times the size of the request set [7]. For a system with N nodes, the size of each request set is roughly square root of N, therefore, total message complexity is $3\sqrt{N}$.The authors, Mukesh Singhal and D. Manivannan Singhal in 1997, conceptualized a "look-ahead" technique for especially infrastructured networks in mobile environment so as to handle DME. The technique, rather than enforcing mutual exclusion among all the nodes of a mobile system, enforced it only among those nodes concurrently competing for CS [9]. Reduction in message overhead was the ramification. Further, "look-ahead" mutual exclusion algorithms eliminates unnecessary communication among sites, hence are more efficient. This technique resulted in message complexity proportional to average number of active sites at any time instead of the total number of sites in the system.The paper by Weigang Wu, Jiannong Cao, Jin Yang in

2005presented the first permission-based solution exclusively for DME problem in MANETs. However, it uses "look-ahead" technique, presented by M. Singhal [10] (for infrastructured mobile networks). The proposed protocol has reduced message complexity in MANET environment. Also, the authors presented timeout mechanism to deal with MANET susceptibility to link and host failures. It provides better performance under high load situations, *i.e.*, when more mobile hosts are active. Furthermore, the paper describes a conventional method for fault tolerance in MANETS [9]. The paper by Moharram Challenger, Peyman Bayat and M.R. Meybodi in 2006 provided proposal for an asynchronous message passing algorithm for distributed system. In their work, notable improvements on the number of messages exchanged are made. To exemplify, a process P_i on finishing CS sends a special message, FLUSH message was sent to both, concurrently requesting process and the next highest priority request, whose requests were not answered earlier [11]. After examining these requests, P_i can determine the sequence of execution of these processes to execute CS. Following this, the optimization is achieved. For an instance, P_k is considered to be the highest priority among all request messages. Then, P_i, other than replying to m nodes, can send reply only to P_k apprising it of all the information that P_i has collected. This leads to reduced message complexity [11], hence, enhancing the performance of the system. Its message complexity, per critical section access, fluctuates between (N-1) and 2(N-1).The approach introduced by Murali Parameswaran and Chittaranjan Hota in 2010, used a new message called *"Hold"* on account toensure that the requesting nodes are alert with information of the currently executing CS node. It used an adaptable timeout mechanism to tackle variant execution times with critical sections [13]. The paper discusses about an algorithm that can deal situations where the CS executing node can fail, with the help of the *"Hold"* message along with the adaptive timeout method. It also informs about the expected time a node remains in CS. Thus, it also resolves the issue that if a node has crashed or executing a lengthy process. The major drawback is the increased message complexity of the algorithm with the introduction of new message *"Hold"*. The improvement could be the reduction in the number of the sites to which *"Hold"* has to be sent.

3. DRABACKS OF TOKEN AND PERMISSION BASED APPROACHH

From the review of existing literature, the following inferences have been drawn:

Token-based approach has the following drawbacks:

- This approach is highly susceptible to the loss of the token. Consequently, a deadlock situation arises.
- Existence of duplicate tokens causes problem.
- For uniqueness of token, complex token regeneration must be executed.

Permission-based approach has the following drawbacks:

- Lamport's algorithm suffered high message overhead. Moreover, the algorithm does not handle failures to make the system fault-tolerant.
- Ricart Agrawala proposed an improved version of Lamport's algorithm. However, it suffered from single point of failure as well as incurs high message complexity.
- In Maekawa algorithm, there is no defined order for messages that are sent to the subset of nodes, which in case of communication delay, leads to deadlock situations.
- Communication delays are typical in a MANET environment. To handle this, new algorithms were proposed with new message like *FLUSH* and *Hold*. Although, the protocols resolves deadlock problem, however, they incurs increased message complexity.

4. MOTIVATION

The prime motivation of our proposed algorithm is to ensure that there is less message traffic with the existence of new message and at the same time guarantees deadlock freedom. The proposed approach works on the reduction of number of sites to which a node has to send "*Hold*" message by applying timestamp priority. This will optimize the reduced flow of messages in the system.

5. ARCHITECTURE MODEL AND ASSUMPTIONS

We assume a MANET comprising of N nodes (N_0-$N_{(n-1)}$), each having unique identification number, *Idno* and a particular timestamp value, T_cs_i (timestamp value of i^{th} node to retain the critical section). Further, the mobile nodes forming dynamic topology communicate with each other as well as access the shared resource in a wireless channel through asynchronous message exchanges. Moreover, only one process accesses the available shared resource.

Therefore, the requesting nodes are notified about the exit time of the current node, to access the critical section. This will also ensure that the current node in the CS has not arbitrarily failed or crashed. It has been presumed that Link and Node failures are certain, however, information can be recovered, either by resetting the values, or by using older set of values. The system model imposes a finite time on the access of CS by a particular node, thereby, maintain liveness into the system.

6. OVERVIEW OF ALGORITHM

*6.1*Data Structures Used

- **Idno:** A unique identification number of each node.
- **REQ_que:** A queue which is maintained by the node in the CS to keep track of the Request messages to access CS.
- **HOLD_que:** A queue which is maintained by the node, currently in the CS, to keep the track of number of nodes to send the "Hold" messages.
- **T_req:** A vector to keep track of the timeout values of REQ messages.
- **T_cs_i:** A vector to keep track of time upto which a node retains the CS.
- **Tcs_exit:** A vector to maintain the amount of time left for current node to exit CS.
- **Inft_set:** An array maintained by each node to keep track of nodes to send REQ message and seek permission before entering CS.

6.2 Types of Messages Used

- *Request for Critical section, REQ*: A mobile node when wants to access critical section, it will send request, REQ, to all nodes in its*Inft_set*. However, the nodes which are not demanding CS, will respond to the requesting node by sending immediate Reply or "*Hold*" message. Also, *T_req* is set, which gives the estimate of round trip time between nodes.
- *Reply message*: When the nodes in the *Inft_set* gets REQ, which contains identification number and timestamp value. Nodes themselves check if they are requesting for CS or not, then they send immediate Reply message (if not requesting). After getting Reply from all nodes in its *Inft_set*, it enters CS.
- *Hold message*: While the node is in the critical section, if it gets REQ then it will send "*Hold*" message which encloses *Tcs_exit*, which specifies the amount of time left for it to exit the critical section.

7. PROPOSED MODEL

In MANETs, suppose, there are many nodes that are requesting for CS simultaneously, sending "*Hold*" message to all by current node in CS becomes overhead. Therefore we use the concept of low timestamp value here, the nodes with low timestamp value in their REQ will be send "*Hold*" message to notify the amount of time left by current node to exit CS.

7.1Working of algorithm

In the mobile ad hoc environment, the working of proposed algorithm is divided into two scenarios. In both the scenarios, the commonalities are:

- We assume that there are four mobile nodes, N_0, N1,N2and N_3 forming MANET.
- Each mobile node has its own identification number, Idno.
- When a node wants to enter CS, it sends request to other nodes and waits for their Reply, thereby using them as permission (either "Hold" or Reply) to enter into the CS.

7.2 Scenario1

Initially, we assume that there is no node in the CS and also, Request queue, REQ_que is empty. Suppose, at some interval, mobile node N_0 wants to enter CS. It sends Request, REQ that possess its identification number and timestamp value, to its own Inft_set. All nodes in the Inft_set, if not interested in accessing CS, will reply to the node N_0 by sending Reply message as permission to the node. After obtaining all Replies, N_0 enters into CS. The algorithm for entering into CS is discussed below in the form of pseudo code given in table 1:

```
//mobile node N0 wants to enter CS
proc Send_REQ
Begin
{
SetN0 Idno;
SetN0 T_csi;        //time to retain CS
for (N1, N2, ... Nn ε info_set0)
{
Send REQ (Idno + T_csi);
Set T_req for every REQ;
If (N1, N2 ... Nn are not demanding CS)
{
Send "Reply" to N0;          }
N0 enters CS.          }
}
End
```

Table1. Algorithm for requesting CS

7.3 Scenario2

In the second scenario, we have proposed an algorithm where N_0 is already in CS. Further, N_2 and N_3 want to access the CS simultaneously. Nodes N_2 and N_3 will send REQ embedded with the timestamps, to their corresponding *Inft_sets.* As N_0 is present in the *Inft_set* of both the nodes, timestamp priority is used to break the symmetry of concurrent request messages. Among the requesting nodes, the one with low timestamp, *T_cs* ,will receive "Hold" message from N_0 , shown in Fig2

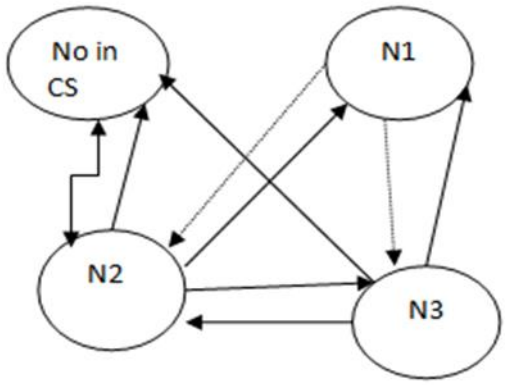


Figure 2: Concurrent request from N_2 and N_3 while N_0 in CS

Request message ⟶

Reply message ⋯⋯▸

Hold message ⟵⟶

Following table presents the pseudo code of the second scenario:

```
// N0 in CS, N2 and N3 demands for CS simultaneously
Begin
   Set N2 Idno + T_csi;
   Set N3 Idno + T_csi
   N2 send REQ ( N0, N1, N3 ε info_set2 );
N3 send REQ ( N0, N1, N2 ε info_set3);
 If ( N1 doesn't demand CS )
 {
 Send "Reply message" ;
 }
else N2 and N3 waits;
 for ( N0 in CS )
 {
 Add N2 and N3 REQs to REQ_que of N0 ;
 Compare T_csi of all REQ (REQ_que);
 Send "Hold" message to low T_csi, N2 ;
 Set Tcs_exit;  // for every "Hold" message//
Add N3 to "HOLD"_que;
}
N0 N0 exit CS;
                N2 enters CS;
}
End
```

Table 2: Algorithm for Hold Message

8. PROOF OF CORRCTNESS

The section discusses the proof of three properties Liveness, Fairness and Safety, to ensure the correct working of the proposed algorithm.

Theorem 1: With the help of "Reply" and "Hold" message, the algorithm ensures fairness as well as determines the waiting time.

Argument: Assume that a mobile node N_j wants to access critical section while another node N_i is already executing CS. Any site that belongs to the information set as well as requesting CS simultaneously, receives either a Reply message or a "Hold" message. The Reply messages are sent immediately by the nodes which are not demanding CS access. On the other side, the "Hold" message is sent by the node N_i, currently in CS. This informs about the waiting time to the requesting node. Further, if more than one process request for CS at the same time, the decision of sending "Hold" message is made on the basis of their timestamp values. Also, it proves that only one process per node executes the CS

Theorem 2: The algorithm ensures liveness.

Proof: Presuming a situation, when more than one node requests for the CS access, simultaneously. Since, each request in the proposed algorithm is timestamped, which is already received by every node in the Inft_set. Therefore, based on the timestamp priority, requesting node with lower timestamp value will be sent a "Hold" message. Also, the node is notified about its waiting time. This ensures CS availability to all nodes and therefore, guarantees liveness of the system.

Theorem 3: The algorithm ensures Safety.

Proof: Without the loss of generality, frequent node/link failures occur in dynamic MANET environment. This results in the loss of messages. If the failed or crashed link/node is not in Inft_set and is not waiting for Reply, then there will be no effect on the execution. Whenever link/node failure occurs, it will recover after retrying time period or resetting to the older values. Thereafter, resuming to its normal functions, it can participate in the network execution, thereby, ensuring safety.

9. RESULTS

Result I: Comparison of Message Complexity

The Fig 3 shown below represents the comparison of the number of Hold messages using the proposed technique with the existing technique.

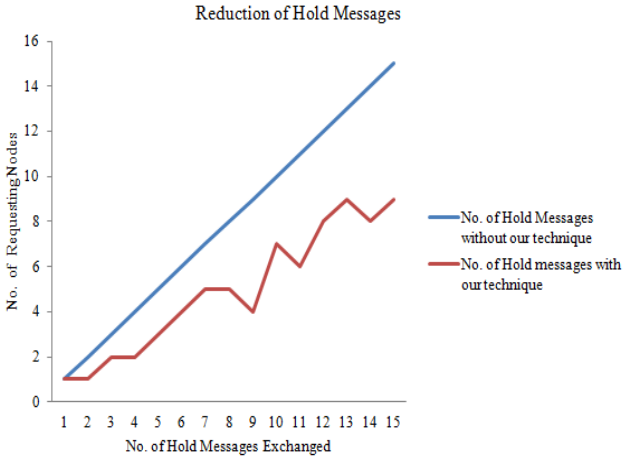


Figure 3 Hold messages exchanged with or without our technique

The graph shows the number of Hold messages exchanged among the requesting or contending nodes to execute CS. In the existing protocol, the Hold message is send to all the contending nodes by the node currently executing the CS. Therefore, with the increased number of contending nodes, the number of Hold message also increases linearly (blue line). On the other side, using the proposed technique, the number of Hold messages reduces by applying timestamp priority. However, at times, there can be some nodes requesting CS at the same time. In such cases, the Hold message will be send to the node having lowest timestamp value and rest of the requesting nodes will form the queue. This, in turn, will substantially reduce the overhead of sending Hold messages to all nodes (red line). The variation reflected in the graph appears, in case, more than one node, having same timestamp values, requests for the CS access. In this situation, the Hold message is send to all nodes having same timestamp values.

Result II: Bandwidth In Terms of Total Messages Exchanges per CS entry

The Fig 16 shows the comparison between the Performance of existing and proposed Technique. With Performance, we refer to the total Bandwidth, i.e., Per CS entry and exit operation, total number of messages exchanged.

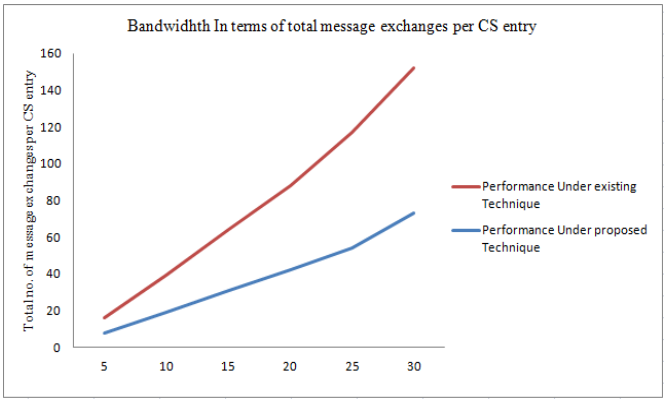


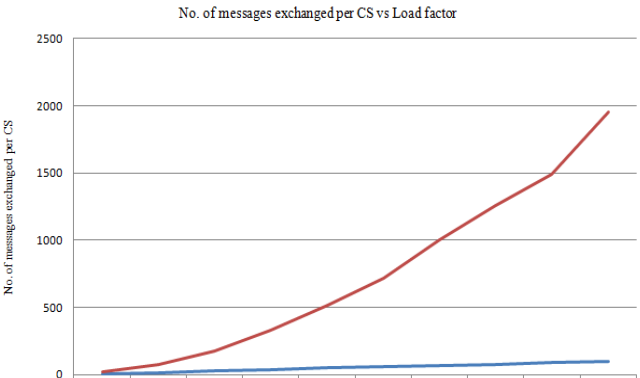
Figure 4 Bandwidth of the System

We have analyzed our proposed algorithm for N = 5, 10, 15, 20, 25, 30. Here, N represents the number of sites competing for mutual exclusion to enter CS. With increase in the number of competing nodes, the number of messages exchanged with each entry and exit is also observed to increase. In general, the performance of a distributed mutual exclusion algorithm is determined by the load of number of message exchanges under heavy/ light load . The light load refers to a situation where there is only one node in the critical section and no pending/new request is there. On the other side, when there is high demand for CS access, leading to piling up of the requests, the system results in heavily loaded situation. The obtained graph represents that there is less flow of messages using proposed algorithm in comparison with the existing. In addition, less message overhead means improved performance.

Result III: Performance Under Light and Heavy Load

The Fig 5 shows the performance of our proposed technique under light and heavy load i.e, lesser and higher number of nodes requesting the critical section execution. The graph represents varying nodes N and number of messages exchanged per CS entry under load level 1 and load level 50%.

The load level 1 is the ideal case where there is one node at a time that requests for CS. Under load factor 50% means half of the nodes in the info set requests for CS access at the same time. This makes the resquest queue heavily loaded. The graph shows that the increase in load level will increase the number of messages exchanged per critical section.



	5	10	15	20	25	30	35	40	45	50
L=1	9	17	29	39	49	59	69	79	89	99
L=50%	22	78	177	330	518	720	1006	1260	1493	1950

Figure 5 No. of messages exchanged per CS vs. Load Factor

10. PERFROMANCE

In proposed algorithm, the message complexity will exceed 2(φ-1) because of using additional "Hold" message. However, the proposed algorithm reduces total number of "Hold" messages when compared to [13]. The message complexity will be [2(φ-1) +t*n] where t is the timeout period and n is number of nodes in HOLD_que. In the algorithm, we have optimized the t*n parameter by reducing n factor and applying timestamp priority, thereby, leading to controlled flow of messages in the system.

11. CONCLUSION

Various solutions have been framed and can be found in the literature for achieving DME using Token-based and Permission-based approach. In Permission-based solutions,a process that requests to access CS must receive permission from all nodes in its information set by message exchanges. However, the number of messages exchanges is large in the existing literature. The proposed work focus on the reducing the number of message exchanged including new message "Hold", thereby, optimizing t*n parameter. Further, this reduces the overall latency, thus, increasing the performance of the system.

REFERENCES

[1] B.D. Kshemkalyani and M. Singhal, Distributed mutual exclusion algorithms in Distributed Computing Principles, Algorithms and Systems, 1st ed. Cambridge University Press, May 2008.
[2] G. Coulouris, J.Dollimore, Tim Kindberg, Distributed System concept and Design Addison-Wesley, Pearson Education, 2001.
[3] Tanenbaum, A.S., and Steen M.V, Distributed Systems Principles and Paradigms, Prentice-Hall International, Inc, 2002.
[4] Lamport, L. "Time, clocks and the ordering of events in a distributed system.," Comm. A CM 21, pp 558-565, 7 July 1978.
[5] G.Ricart and A. K. Agrawala, "An Optimal Algorithm for Mutual Exclusion in Computer Networks," Communications of the ACM, pp 9-11, 1981.
[6] Maekawa, M., Oldehoeft, A.E., and Oldehoeft, R.R, Operating Systems Advanced Concepts, Menlo Park, CA: Benjamin/Cumings, pp:200-208, 1978.
[7] M Maekawa, "A √N algorithm for mutual exclusion in decentralized systems," ACM Trans on Computer Systems, Vol. 3, No 2, pp. 145-159, May 1985.
[8] M. Singhal, "A Taxonomy of Distributed Mutual Exclusion", Journal of Parallel and Distributed Computing 18(1), pp.94-101, 1993.
[9] M. Singhal, and D. Manivannan, "A Distributed Mutual Exclusion for Mobile Environments", Proc. IASTED Intl. Conf. on Intelligent Systems, pp 557-561, 1997.
[10] Weigang Wu, Jiannong Cao, Jin Yang, "A Scalable Mutual Exclusion Algorithm for Mobile Ad Hoc Networks," Proc. of the 14th International Conference on Computer Communications and Networks (ICCCN2005), San Diego, USA, Oct. 17-19, 2005.
[11] Moharram Challenger, Peyman Bayat and M.R. Meybodi, "A Reliable Optimization on Distributed Mutual Exclusion Algorithm" TRIDENTCOM, 2006
[12] Bharath Kumar A.R. and Pradhan Bagur Umesh , "An Improved Algorithm for Distributed Mutual Exclusion by Restricted Message Exchange in Voting Districts" in11th International Conference on Information Technology, 2008.
[13] Parameswaran, Murali; Hota, Chittaranjan, "A novel permission-based reliable distributed mutual exclusion algorithm for MANETs," Wireless And Optical Communications Networks (WOCN), 2010 Seventh International Conference On, vol., no., pp.1-6, 6-8 Sept. 2010.
[14] Parameswaran, M.; Hota, C., "Arbitration-based Reliable Distributed Mutual Exclusion for Mobile Ad-hoc Networks", Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, 2013 11th International Symposium and Workshops on, vol., no., pp.380-387, May 13-17, 2013.
[15] K. Erciyes, "Distributed mutual exclusion algorithms on a ring of clusters" ,Proc. of International Conference on Computational Science and Its Applications ICCSA 2004, vol. 3045/2004, LNCS, SpringerVerlag, May 2004, pp 518–527. doi: 10.1007/b98053.
[16] I. Suzuki and T. Kazami, "A distributed mutual exclusion algorithm," ACM Trans on Computer Systems, Vol.3, No.4, pp 344-349, Nov 1985.
[17] S. M. Masum, M. M. Akbar, A. A. Ali and M. A. Rahman, "A consensus-based ℓ-Exclusion algorithm for mobile ad hoc networks, " Ad Hoc Networks, Vol. 8, N o.1, pp. 30-45, 2009.
[18] J. E .Walter, J. L. Welch and N. H. Vaidya "A Mutual Exclusion Algorithm for Ad Hoc Mobile Networks," Wireless Networks, Vol. 7(6), p. 585-600, 2001.
[19] K. Raymond, "A tree-based algorithm for distributed mutual exclusion," ACM Trans on Computer Systems,pp. 61-7, Feb. 1989
[20] B.Sharma, R. B. (2014). DMX in MANETs: majorresearch trends since 2004," Proceedings of the InternationalConference on Advances in Computing and Artificial Intelligence. *ACM* , pp.50-55.
[21] Shruti, P. S. (2015). Quorum-based Mutual Exclusion Algorithm forMobile Ad-hoc Network (MANET). *ICCCA* .