Fault Tolerant Resource Management with Mutual Exclusion Algorithm for Mobile Adhoc Networks

Sri B.Mallikarjuna Smt V.Saritha Smt Ch.Puspalatha

Abstract- In this paper we propose permission based Resource Management with Mutual Exclusion Algorithm for mobile ad-hoc networks (MANET). To reduce the message cost, the algorithm uses the DAG (Directed Acyclic Graph) technique, which enforces MUTEX only among the hosts currently competing for the critical section. We propose mechanisms to handle doze and disconnections of mobile hosts. The algorithm can also tolerate link or host failure we would like to propose an algorithm which is completely fault tolerant (covers temporary and permanent faults).It also has the mutual exclusion property for critical resource management to solve permission based algorithm. At the end we have proved the algorithm mobility is high and load level is low.

Index Terms- MANET, fault tolerant, DAG, safety, liveness, critical section, MUTEX

I. Introduction

In a mobile environment especially a MANET (mobile ad hoc networks), link failure (eg: single shielded) and host failure (eg: battery exhausted) occur frequently. Link failure may lead to message loss while host failure may result in accidental disconnections. Such failures are to be handled by mutual exclusion algorithm .Mutual exclusion provides access to shared critical resources (resource which may be accessible by a single process at a time). I would like to propose permission based solutions, with token based solutions, there exist a unique token in the system and only the node holding the token may access the critical section. To access the critical section using permission based solution[1] a process P_i is require to receive permission from a set of nodes $S=\{S_1,S_2,..,S_n\}$[2]. Every node may communicate directly with its neighbors by exchanging messages and keep information about its neighbors. The communication delay is assumed to be finite but not bounded. In mobile adhoc network the topology of the network is arbitrary the topology may change with respect to the time. Any mutual exclusion algorithm have to ensure two properties: Safety[2] and Lifeness[2].The ***safety*** property ensures that at most one process is executed in critical section at any time, while the ***lifeness*** property ensures that requesting node will succeed to enter its critical section in finite time. Performance of mutual exclusion algorithm is to be evaluated by the number of messages generated per critical section entry, synchronization delay, and size of information control.

A solution to the MUTEX problem must satisfy the following three correctness properties

Mutual exclusion (safety): At most one host is allowed to enter the CS at any movement
Deadlock free (liveness): If any host is waiting for the CS, then in a finite time some host enters the CS
Starvation free (Fairness): If a host is waiting for the CS, then in a finite time the host enters the CS

II. Background

The hosts are intermittently requesting to enter the critical section(CS) to gain exclusive access to shared resources. So many MUTEX algorithms for MANETs have been proposed in literature. logical ring [3] and considers the delay and distance between the nodes, to place them on different clusters It reduces the intra-cluster message and gives higher priority to local nodes in a cluster message and gives a higher priority to local nodes in a cluster, for entering the critical section [2]

This algorithm applies three extensions to Naimi-Trehel's algorithm[5], based on idea of limiting the propagation of requests between nodes of different clusters. Two different types of permissions have been used in existing permission-based algorithms [10]. The Ricart–Agrawala type permission, which is adopted in our proposed algorithm, is proposed by Ricart and Agrawala [21]. A host that wants to enter the CS, sends request messages to all other hosts. Requests for CS are assigned globally unique priorities, e.g.Lamport-like timestamps [10]. If the receiver of a request is not requesting the CS or its priority is lower, it grants permission to the requester immediately by sending a reply. Otherwise, it grants the permission after its own execution of the CS. The semantics of such permission is “as far as I am concerned, it is OK for you to enter the CS” [21]. A variation of the Ricart–Agrawala algorithm is proposed in [7], by remembering the recent history of the CS execution so as to reduce message cost. Singhal proposed a dynamic Rciart–Agrawala type algorithm [20], by dynamically changing the set of the hosts to which a requesting host needs to send request messages.

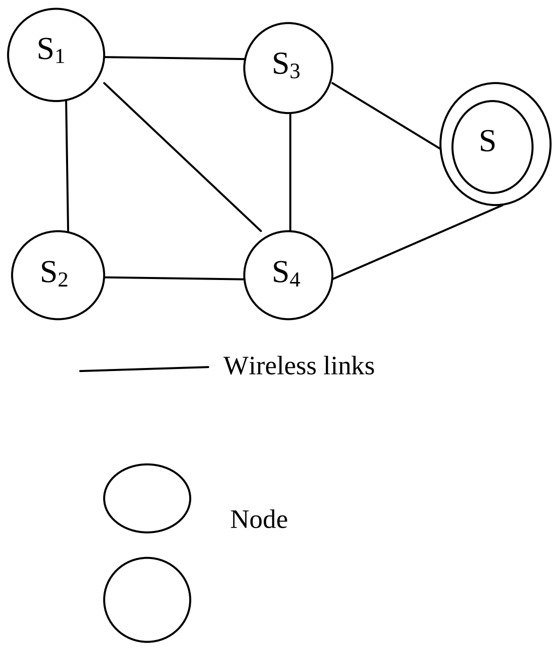
The algorithm proposed in, based on which our new algorithm is designed, is also Reicart–Agrawala type algorithm. It made a modification to the Ricart–Agrawala algorithm [10] so that, instead of involving all the hosts in the system, MUTEX is enforced only among the hosts which are currently competing for CS. On each host S_i , there are two sets. The Info_set$_i$ includes the IDs of those hosts which S_i needs to inform when it requests to enter CS, and the Status_set$_i$ includes the IDs of the hosts which would inform S_i when they request to enter CS. If a host wants to enter CS, it just sends request to the hosts in its Info set. When a host wants to disconnect from the network, it offloads the current values of its data structures to its serving MSS which would then act on behalf of the host in the execution of the algorithm.

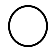
III. PROPOSED ALGORITHM

We define the algorithm as a structure mapped on real topology of network which is represented by a Direct Acyclic Graph (DAG) of permission based pointers , maintaining the multiple paths leading to the node holding the permission . The algorithm is well suited to the resource management mutual exclusion because it requires nodes to keep information only about their immediate neighbors. Each node keeps information which is used to update the Direct Asyclic Graph strucure in case of link failure in order to have always the permession holder node always the root of this tree. It is assumed that the permission cannot be lost and communicational links are bi directional. The nodes move with a limited speed so it cannot disconnect from the network during activation of the algorithm and during message transmission.

The algorithm starts by constructing a permission-based connected by the Directed Acyclic Graph [2], maintained by the permissions distributed over the nodes and directed to the permission holder. The permissions are REQUEST, REPLY and data structures. These permission are defined by the relative elevation of a node in relation to its neighboring nodes. The elevation of node i has the form (α,β,i)[5] and increase according to its distance from permission holder. The neighbors of each node are divided into two sets: a set of nodes connected to incoming links and a set of nodes is connected to outgoing links in order to maintain many routes to the nodes holding permission. When it receives the first request from one of its neighbors it maintains the request queue to store and order requests and also for backsword path to reach the requesting node. Upon receiving permissions, the nodes which detect its own id is in the top of requesting queue becomes a “sink” by modifying its elevation to be lower than its neighbors and finally enters the CS. The request holder will always be the lowest in the DAG..So, the partial rearrangement of DAG is necessary.

During the execution of algorithm, some links may fail and/or may be created. In figure 1, part (a) depicts a simple mobile ad hoc network





a) Example of mobile ad hoc network

Incom_Lset$_i$: The array of the IDs of the hosts to which Si needs to send request messages when it wants to enter CS.
Outgoing_Lset$_j$: The array of the IDs of the hosts which, upon requesting to access CS, would send the request messages to Si .
To ensure the correctness of the algorithm, the following conditions must be satisfied:

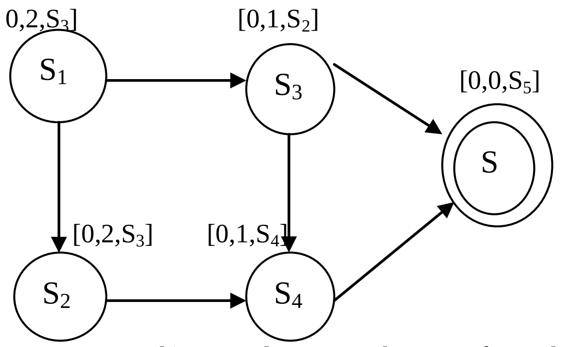
(1) $\forall S_i$ :: Incom_Lset$_i$ $\cup$ Outgoing_Lset$_j$ = S;
$\forall S_i$ :: Incom_Lset$_i$ $\cap$ Outgoing_Lset$_j$ = $\emptyset$;

(2) $\forall S_i$ $\forall$ S_j :: S_i $\in$ Incom_Lset$_i$ => S_J $\in$ Outgoing_Lset$_j$;

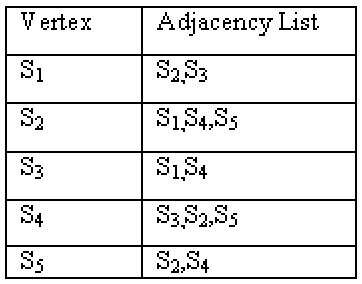
Obviously, condition
(1) The host is guarantees that host S_i knows the request status of all the other hosts and there is no redundancy information. Condition
(2) The host is guarantees the consistency among the sets of all MHs

REQUEST: the message sent from a host requesting CS to other hosts for getting their Permissions. The message contains the priority of the request (e.g. unique timestamp).
REPLY: the message sent by a receiver of a REQUEST to grant the permission of accessing CS

Initialization of the algorithm is



b) Initialization and request for node S_5



Vertex	Adjacency List
S_1	S_2,S_3
S_2	S_1,S_4,S_5
S_3	S_1,S_4
S_4	S_3,S_2,S_5
S_5	S_2,S_4

Observe that each host Graph G is represented an an adjacency structure Figure (b) shows the state of the network with logical links after initialization of the algorithm. Node S_5 is holding the request(has no out going links) and all other nodes point to host

S_5..Along the paths to request holder node, the elevation decrease. Node S_1 is issue a request for the over $path_A$ which is enquired on S_4 request queue, and forwards the request to node S_2 which enquires the S_4 request. And finally S_3 S_3 enquires S_2 request

Linked representation of graph G, which maintains G in memory by using its adjacency lists, will normally contain two files (or set of records), one called the host file and other called Edge File, as follows

a) Host File: Host file will contain the list of vertices of the graph G usually maintained by an array or by a linked list. Each record of the host file will have the form

HOST	NEXT-HOST	PTR	

Here host will be the name of the host , NEXT-HOST points to the next host in the list of host in the host file when the host are maintained by a linked list. and PTR will point to the first element in the adjacency list of the host appearing in the Edge file. The shaded area indicates that there may be other information in the record corresponding to the host

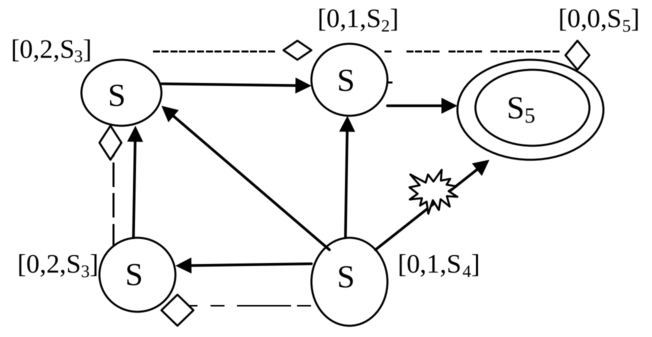
b) Edge file: The edge file contains the edges of the graph G. Specifically the edge file will contain all the adjacency lists of G where each list is maintained in memory by a linked list .Each record of the host file will correspond to a host in an adjacency list and hence, indirectly to an edge of G. The record is usually of the form

EDGE	ADJ	NEXT	

Here:
1. EDGE will be the name of the edge host (if it has one)
2. ADJ points location of the host in the host file
3.NEXT points to he location of the next host in the adjacency List

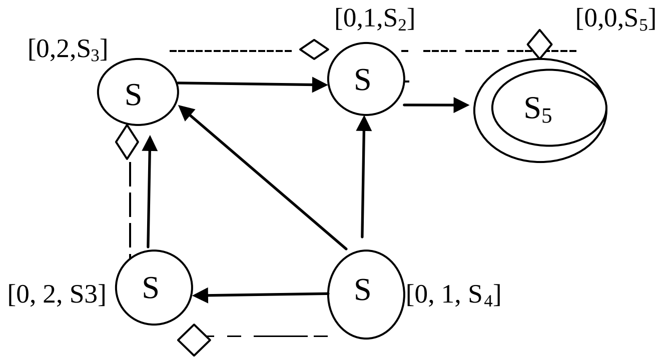
Figure c shows how the graph G in fig(c) may appear in memory. Here the hosts of G are maintained in memory by a linked using the variable START to point to the first host. (Alternatively, one could a linear array for list of hosts, and then Next-Host would not required.) Note that the field EDGE is not needed here since the edge have no name. Figure(c) also shows the adjacency list (S_5, S4, S1) of the host S_2.
The algorithm mutual exclusion is
Step 1: the all hosts are randomly allocated to the host file
Step 2: any one of the host will become the start pointer
Step 3: the next host will be allocated through the host file
Step5: Based on the pointer the adjacency will be executed from

the left to right
$\longrightarrow$ Logical pointer (corresponds to wireless links)
------------◇changing path message transmission



d) Handling of link failure by the algorithm

When a node detects a failure of its last existing out going link, this results that there is no path to the request holder. Then, it invokes a partial rearrangement of the DAG using the method described in [3] which avoids formation of a cycle. Generally, the failure may happen at any time, the algorithm provides the mechanisms to ensure its normal functioning. In the example given in figure (c), the link S_4 –S_5 has failed due to an increase distance between hosts S_4 and S_5. The failed link is the last out going S_4 which does not hold the request .So , S_4 rearranges its links, changes its elevation that causes host S_2 to have no out_going links. Also, after this, S3 has any out going links. Consequently, the tree is partially rearranged.



c) Handling the messages

Figure (d) shows the result of permission grant from host S_5 to S_1 causing the elevation changes of the hosts S_2 ,S_1 and S_3 . These changes ensure that all logical links point from hosts with higher elevation towards hosts with lower elevation

When a new link is detected , the two adjacent nodes of this new link exchange messages to achieve the necessary modifications of outgoing and incoming links.The algorithm guarantee the safety and liveness property[6].

When a host wants to enter the CS, it first sets TS_{req} to the current time and sends the REQUST message to all the hosts its Incoming_Lset. To tolerate link and host failures, a timeout is set in TO_{req} for each request message sent. The host then waits for a REPLY message a corresponding to each REQUEST message sent out. If the $Incoming\_Lset_i$ is empty. It enters CS immediately

When a host S_i receives a REQUEST message from another host S_j, it moves S_j to $Incoming\_Lset_i$ and records the request in $Q_{req..If}$ S_i itself is not requesting for CS or its priority is lower, it sends a REPLY message to S_j and removes the record for Sj in Qreq. If Sj is in $Outgoing\_Lset_j$ before Si receives the REQUEST form Sj and Si is requesting for CS with a lower priority, Si sends a REQUEST to Sj.

Upon receiving of a REPLY message from host Sj , Si removes the timeout (in TOreq) associated with Sj . If Si finds no request from Sj in its Qreq, Sj is moved to Outgoing_Lset$_i$.

When the timeout for a REQUEST message expires, the requesting host sends a REQUEST again. When all the replies for REQUEST messages have been received, the requesting host enters CS. On exiting CS, a host sends REPLY messages to all hosts in its Incoming_Lset$_{i.}$.

It is worth notice that when two hosts compete for the CS simultaneously, if we do not recorder the REQUEST separately in Qreq, it is possible that the host with the lower priority never gets a REPLY from the other host. This is caused by the non-FIFO property of communication channels.

Here any host failure the resource management algorithm is to first created the adjacency list the host file it allocates the all hosts randomly and then pointer will be generated any host having the malicious, it change the path the pointer points the new path again it construct the new DAG technique we fallow The packet transmission path will be reconstructed fig(d) may appear in memory. Here the hosts of G are maintained in memory by a linked using the variable START to point to the first host. (Alternatively, one could a linear array for list of hosts, and then Next-Host would not required.) Note that the field Adjacency is not needed here since the edge have no name. Figure (d) also shows the adjacency list ($S_{5,}$ S4, S1) of the host S_2 after the fault tolerance

IV. PERFORMANCE ANALYSIS

Number of messages per CS entry (MPCS):

The algorithm says that the average number of messages to enter into the CS is incoming link set and out gong link set are equal. So the number of messages exchanged among the hosts for each execution of the CS.

Therefore the MPCS under low load condition is ;

$MPCS_{low} = (n + n) / 2= n$

Under the high load condition n/2 hosts issues current request messages .Each request message the reply message must be send so half of these hosts are in outgoing link list is n/4. Therefore the MPCS under the high load condition is:

$MPCS_{high}= (n/2 + n/4) = 3 * n/4$.

Table1: The table shows #MPCS the high and low load

No. Of Hosts	#MPCS-LOW	#MPCS-HIGH
0	0	0
20	20	7.5
40	40	15
60	60	22.5
80	80	37.5

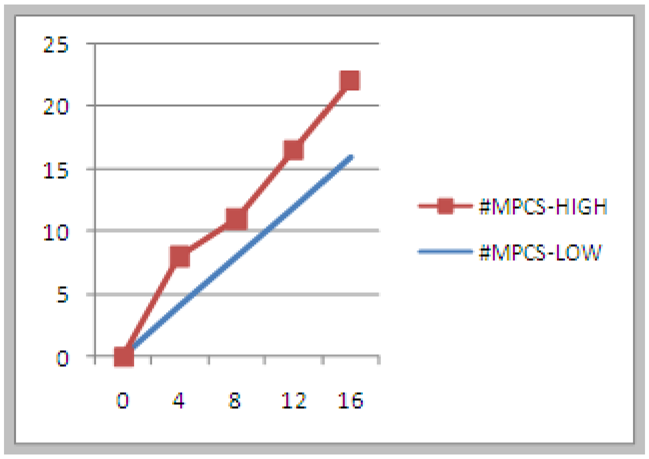


Fig. 1: The graph shows the #MPCS at high and low load

Table 2: The table shows #MPCS the high and low load

No. Of Hosts	#MPCS-LOW	#MPCS-HIGH
0	0	0
4	4	4
8	8	3
12	12	3.5
16	16	6

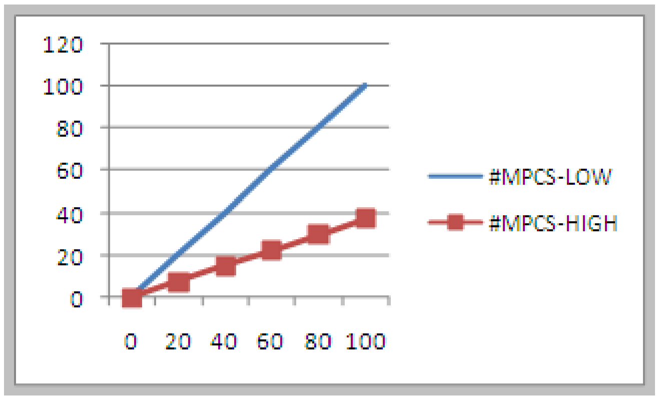


Fig. 2: The graph shows the #MPCS at high and low load

Synchronization delay (SD): The number of sequential messages exchanged after a host leaves the CS and before the next host enters the CS .Under low load conditions the synchronization delay is meaningless, because it measures the interval between the arrival of two requests. Under the high load conditions, when host S_i exits the CS, it will send REPLY messages to all the hosts in its Incoming_Lset, i.e. that is the host that have pending requests at the earliest time will enter the critical section immediately after it receives the REPLY from S_i. Therefore, the Synchronization delay under the high load condition is

SD_{high}=1, i, e.one message transferring time.

Response time: The time interval that a host waits to enter the CS after its request for CS arrives. Under low-load conditions,

most of the time, no more than one host competes for the CS. When a host wants to enter CS, it sends REQUEST messages to the hosts in its Incoming_Lset and then all these hosts send REPLY immediately after they receive the REQUEST. Therefore the response time under the low-load level is:
$RT_{low} = 2$, i.e. twice of the time of transferring one message.
Under high-load conditions, there is always a pending request at each host. The hosts are in the waiting chain with respect to the timestamps of their requests, i.e. the time when they issue requests for CS. A host in the chain can enter CS after its predecessor exits, so each host needs to wait for the hosts whose requests are earlier. On average, each host has to wait for n/2 such hosts. Assuming the average time of an execution of CS is A, the response time under high-load conditions is
$RT_{high}=(A+SD_{high})*n/2=(A+1)*n/2$

Table 3: shows RT_{high} at different average time and different no of hosts

Average time A	No. of host	RT_{high}
2	5	7.5
3	10	20
4	15	37.5
5	20	60

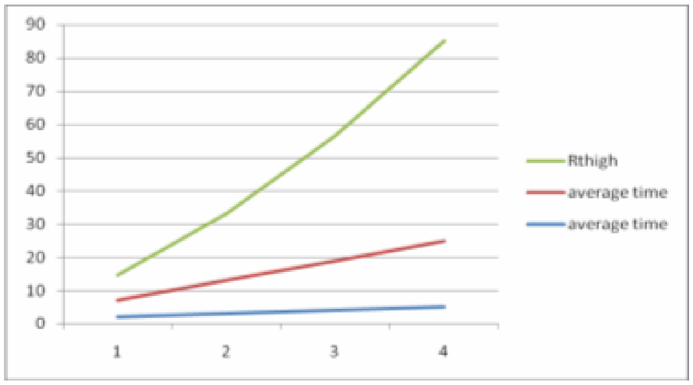


Fig. 4: The graph shows the RT_{high} at different average time and different no of hosts

V. CONCLUSION

In mobile ad hoc network we follow with explanation of fault tolerant happening in Mutual Exclusion algorithm with specific logical topology is DAG[9] for reduce the number of message are exchanged .I designed a mechanism using timeout to tolerate intermittent and recoverable link/host failure which ofter occur in MANETs. The algorithm can also handle dozes and disconnections of hosts .The performance analysis show that the algorithm performs better low load level and high mobility level. The algorithm can save much more communication cost**.** The DAG technique will follow the fault tolerant technique.

REFERENCES

[1] E.W.Dijkstra, solution of a problem in concurrent programming control Comm, Acm (CACM) 8 (9) (1965) 8 (9) 569.
[2] Neeraj Mittala, Prajwal K. Mohan "Apriority-based distributed group mutual exclusion algorithm when group access is non-uniform " Journal of parallel and Distributed computing received in 2 January 2007
[3] J.Beauquier, S.cantarell, A.k.Datta, F.petti,"Group mutual exclusion in tree net works, J.Inform.sci.Eng (JISE) 19 (3) (2003) 415-432
[4] Y.Manabe, j.park, Aquorum-based extend group mutual exclusion algorithm without unnecessary blocking, in: proceeding of the international Conference on Parallel and distributed Systems (ICPADS) Newport Beach, California, USA, 2004, pp.341-48.
[5] Hirotsugu Kakugawa, Member, IEEE, Sayaka Kamei, Member, IEEE, andToshimitsuMasuzawa, Member, IEEE"A Token-Based distributed Group Mutual Exclusion Algorithm with Quorums "IEEETRANSACTION AND DISTRIBUTED SYSTEMS SEPTEMBER 2008
[6] Ranganatha Atreya, Neeraj Mittal, Member, IEEE Computer Society, and Sthya Peri "A Quorum-Based Group Mutual Exclusion algorithm for a Distributed System with Dynamic Group Set" IEEE Transaction on parallel and Distributed systems Vol.18, no.10, October 2007
[7] Mohammad Moallemi,Yasser Mansouri,Amin Rasoulifard and mahmoud Naghibzadeh" Fault-Tolerant Hierarchical Toekn-Based Mutual Exclusion Algorithm " Department of Computer Science and Engineering faculty of Engineering Ferdowsi University of Mashhad, Mashhad, Iran
[8] M.Maekawa,"A $\sqrt{N}$ Algorithm for Mutual Exclusion in Decentralized Systems "ACM Trans. Computer Systems, Vol.3, no.2, pp.145-159, May 1985
[9] Benchaiba.M,Bouabdallah.A,Badache.N Ahmed-Nacer" Distributed Mutual exclusion Algorithm In Mobile Ad Hoc Networks" communication of the ACM, operating system review ,Vol.38 Issue January 2004 ,PP:77-89.
[10] Weigang Wu, Jiannong Cao, Jin Yang "A fault tolerant mutual exclusion algorithm for mobile ad hoc networks" pervasive and mobile computing 4(2008) 139-160
[11] M.Bertier, I.Arants and P.Sens "Hierarchical token based mutual exclusion algorithm". In 4th IEEE/ACMCCgrid 04, 10.
[12] E.Gafni and D.Bertsekas,"Distributed algorithms for generating loop free routes in networks with frequently changing topology," IEEE Transactions and Communications, C-29(1):11-18, 1981
[13] J.E. Walter and S.kini,"Mutual exclusion on multihop, mobile wireless networks," Texas A&*M Univ., College Station, TX 77843-3112, TR97-014, DEC9, 1997
[14] Divyakant Agrawal and Amr EL Abbadi "An Efficient and Fault – Tolerant solution for Distributed Mutual Exclusion" University of California.
[15] Gifford, D.K.Weighted voting for replicated data. In proceeding of the seventh ACM Symposiums on Operating Systems Principals (Dec.1979), 150-159
[16] Thomas, R H. A majority consensus approach to concurrency control for multiple copydatabases.ACM Trans .Database Syst.4,2(June 1979),180-209.
[17] Hirotsugu kakugawa, Toshimitsu Masuzawa"A Token –Based Distributed Group Mutual Exclusion Algorithm with Quorums "IEEE Transactions on parallel and distributed Systems VOL.19 No.9, SEPT 2008
[18] Lisa Higham, Jalal Kawash "Tight Bounds for Critical Sections in processor consistent platforms" IEEE Transactions on parallel distributed Systems, Vol.17, No 10, October 2006
[19] Weigang Wu, Jiannong Cao, Jin Yang " A fault tolerant mutual exclusion algorithm for mobile ad hoc networks" Available on line at www.sciencedirect.com pervasive and mobile computing 4(2008) 139-160
[20] Dan Hakey, Shan Appajodu, Mike Larkin "wireless Java Programming for Enterprise Application" Book
[21] D. Agrawal, A.E. Abbadi, An efficient and fault-tolerant solution for distributed mutual exclusion, ACM Transactions on Computer Systems (1991) 1–20.

AUTHORS

First Author – Sri B.Mallikarjuna, Deportment of Computing Science and Engg, NBKR Institute of Science Technology, Vidyanagar, India. Email id - mallkarjuna.basetty@gmail.com

Second Author- Sri V.Saritha working as an Associate Professor (Sel Grade) School of Computing Sciences VIT University Vellore, India.

Third Author – Smt Ch.Puspalatha, presently she is doing MBA sree padmavathi mahila university Tirupathi She is completed BSCComputers