# Efficient Algorithms for Distributed Mutual Exclusion in Mobile Ad-Hoc Network

R.Karthick[1] , B.Gopinathan[2]
[1]PG Scholar, Department of Computer Science and Engineering
Adhiyamaan College of Engineering,
Hosur,Tamilnadu
India.
[2]Asstistant Professor, Department of Computer Science and Engineering
Adhiyamaan College of Engineering,
Hosur,Tamilnadu
India.
[1]karthick.kasarr@gmail.com
[2]gopinathanme@gmail.com

***Abstract:*** **For distributed mutual exclusion problem in mobile environment we presented two algorithms. Both the algorithms are token based. The first algorithm is providing tokens to the nodes which are in critical section and the second one is hierarchical clustering based. A mobile ad-hoc networks (MANET) is hierarchically (two level) clustered to get logical tree network through which the token is passed from one node to another. However, we simulate the second algorithm only. The evaluation of the proposed algorithms show that its message requirement is optimal, and thus the second algorithm is an energy efficient algorithm.**

***Key terms:*** **mobile ad-hoc networks, Distributed Mutual Exclusion, token based algorithms.**

## 1. INTRODUCTION

A mobile ad-hoc network (**MANET** is a self-configuring infrastructureless network of mobile nodes connected by wireless links. Each node in a MANET is free to move independently in any direction, and will therefore change its links to other nodes frequently. Each must forward traffic unrelated to its own use, and therefore be a router. The primary challenge in building a MANET is equipping each node to continuously maintain the information required to properly route traffic. Such networks may operate by themselves or may be connected to the larger internet.

MANETs are a kind of wireless ad-hoc networks that usually has a routable networking environment on top of a Link Layer ad hoc network.

The mutual exclusion problem involves a group of processes, each of which intermittently requires access to a resource or a piece of code called the *critical section* (CS). At most one process may be in the CS at any given time. Providing shared access to resources through mutual exclusion is a fundamental problem in computer science, and is worth considering for the ad hoc environment, where stripped down mobile nodes may need to share resources. Distributed mutual exclusion algorithms that rely on the maintenance of a logical structure to provide order and efficiency may be inefficient when run in a mobile environment, where the topology can potentially change with every node movement. We present an algorithm which dynamically modifying the logical structure to adapt to the changing physical topology in the ad hoc environment.

Here we presented two algorithms to provide an energy efficient mutual exclusion algorithm in mobile ad hoc networks which is token based and forming a hierarchically clustered network.

## 2. TOKEN BASED ALGORITHM

A network having N number of nodes is splitted into M groups. We consider for simplicity M = N. A node can execute the critical section if that node holds the token. We assume that there is only one token in the network. In this scheme to pass the token within a group, every members of that group execute an algorithm at the same time and the token is passed from one member to another member within a group without any message passing. Each member of the group knows that which member of the group holds the token. Any member can execute the critical section if it gets the token after completion of that algorithm. If at that time any other members of a group want to execute the critical section and if that node is not a token holding node then it sends a request message to that node which have the token and belongs to the same group. If at that time the token holder is not executing the critical section, then that node can grant permission to the member whose request received first. Every node in the network holds a list. The list receives request from other members within its group or from other groups. Size of the list is one, so one node receives only one token-request. When execution of the critical section is completed then the token holding node will sends release message to every member of that group.

After receiving the release message every member of the group including the token holding node again start the algorithm. After completion of execution of the algorithm the token will pass to a new node without any message passing. New token holding node will send a message to the last token holder of other groups that he is the new token holder of that group. New token holding node can execute the critical section or can grant permission to first requesting member of the group to execute the critical section. If the token holder has already grant permission to other member of the group to execute the critical section, then that member will send a release message to the token holder after executing the critical section. After receiving that release message the token holder sands release message to every member of that group. The release message contained the ID of last token holding node of other groups. If the token is not within a group and if any last token holders of that group want to execute the critical section then it sends a request message to the token holding node in other group. If at that time the token holder remains free then he will release the token to one of the last token holder of other groups whose request received first. If other members of that group want to execute the critical section then it has to sends a request to the last token holding node of its group and then that node sends a request message to the node in other group who holds the token. If the token holding node not remains free then he will not send any message. We next propose a new algorithm, specially designed for MANET.

## 3. AN ENERGY EFFICIENT ALGORITHM FOR DISTRIBUTED MUTUAL EXCLUSION IN MOBILE AD-HOC NETWORKS

In this section, we report a new energy efficient mutual exclusion algorithm for mobile ad-hoc networks. We have addressed the issue of mobility of nodes extensively. Utilizing the hierarchical structure of the network, the proposed algorithm significantly reduces the message requirement per mutual exclusion entry. Therefore, overall energy requirement for message communication of the system is optimized. The whole network is hierarchically clustered to get a logical tree structure. The proposed algorithm is token based and works on the logical tree to obtain the mutual exclusion. A mobile ad-hoc network (MANET) is a collection of mobile nodes that communicate through wireless links. So the topology of such type of networks can dynamically be changed. The complexity of mutual exclusion problem is much higher because there no static topology can be considered. According to the algorithmic principle, the mutual exclusion algorithms for distributed systems can broadly be classified into two categories

*i.* *permission based algorithms and*
*ii.* *token based algorithms*

The permission based algorithms (e.g. [15], [11], [9], [10] etc.) require two (or more) successive rounds of message exchanges among the competing processes. A process can enter in the critical section only after receiving permission from other process(es) in the system. Lamport's algorithm requires approximately 3(n - 1) a message per mutual exclusion invocation where n is the number of distributed processes [8]. An algorithm, proposed by Ricart and Agrawala [15] for this problem, requires 2(n - 1) messages per CS entry. In this algorithm only after receiving permission from all the other processes, one process can enter into the CS. Mamoru Maekawa proposed a distributed mutual exclusion algorithm [11] which requires only $3\sqrt{n}$ to $5\sqrt{n}$ messages per mutual exclusion. On the other hand, in token based mutual exclusion algorithms a unique token or a privilege message is shared among the processes. Token gives the authority to a node to execute the CS. Suzuki and Kasami's token based algorithm, based on the concept of node privilege, requires n messages per CS entry [16]. In Raymond's algorithm, the nodes are arranged in an un-rooted tree structure [14] which normally is a minimal spanning tree of the network. The message complexity of the algorithm is O(log n) under light demand. Algorithm proposed by Pranay, Chaudhuri, Mehmet Hakan Karaata [6] achieves the message complexity as $O(n^{1/3})$ per mutual exclusion. They assumed that the n node network is available in the form of a three-dimensional mesh. However, all the algorithms, reported above, are developed for static networks. Those algorithms cannot be efficiently utilized for MANET. The first mutual exclusion algorithm for the semi ad-hoc cellular networks was proposed by B. R. Badrinath and Arup Acharya [3]. In this algorithm, base station acts as a proxy for nodes attached to it and the request queue is maintained only by the base station. The base stations are arranged in a logical ring. J.Walter and S. Kini's algorithm is a token based algorithm [18] for ad-hoc networks. The algorithm defines the Direct Acyclic Graph (DAG) to map the physical topology of network. The algorithm maintains multiple paths leading to the token holding node through DAG. They considered that the mobility of the nodes is slow and they do not considered the token loss. J. Walter, J. Welch and Vaidya assumed that communication channels are FIFO with no loss [19]. Each node dynamically chooses their neighbor with lowest elevation as its preferred next link to the token holder. This algorithm also acts with the same assumptions as [18]. R. Bladoni, A. Virgillito's algorithm [4] is based on a dynamic logical ring and combines the two methods of token asking and token circulation. The algorithm reduces the power consumption by reducing the number of hops traversed per CS execution. This algorithm needs n number of messages per CS entry under light load. N. Malpani, N. H. Vaidya proposed a parametric algorithm [12] with many variants. It uses dynamic logical ring and the size of the ring may vary at every round. The main idea of the algorithm is the method of choosing the next node to which the token will be sent. The variant's policies applied to determine the next node. The algorithms in [4] and [12] are not aware of the nodes mobility. Romain Mellier, Jean-Frederic Myoupo presented a token based mutual exclusion algorithm for multi-hop mobile network with O(n) broadcast rounds [13]. Ranganath Atreya, Neeraj Mittal's algorithm [2] achieves the message complexity of O(q) and a bit-message complexity of O(bqr), where q is the maximum size of a quorum, b is the maximum number of processes from which a node can receive critical section requests, and r is the maximum size of a request while maintaining both synchronization delay and waiting time at two message hops. In this token based algorithm for MANET we extensively address the issue of mobility and token regeneration. Next we report the properties of the system under consideration, for which the algorithm is developed. The properties of the system for which the algorithm is designed, are reported next.

## 4. PROPERTIES OF THE SYSTEM UNDER CONSIDERATION

The proposed algorithm works on a clustered network. The clustering technique, adopted in this work, is discussed in this section. Moreover, we have few assumptions regarding the system. Those assumptions are mentioned next.

### *4.1 Assumptions*

The system contains a number of independent nodes. The nodes are mobile and they communicate with each other only through message passing. Here we assume some characteristics of the system in which we run our algorithm.

*Assumption 1:* Communication is reliable, i.e., there is no message loss during transmission and the receiving node can receive message without any error or distortion.

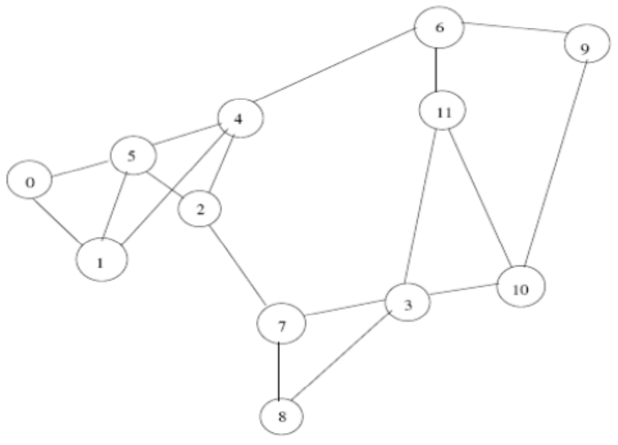


Fig 1: a MANET with 12 nodes

*Assumption 2:* The message passing from one node to another node is time bounded and the time is negligible.

*Assumption 3:* The channel is a FIFO channel i.e., message passed through the channel in a first come first serve manner and the communication links are bidirectional links.

### *4.2 Clustering*

For the execution of the proposed algorithm, the network is to be clustered hierarchically [5],[7],[1],[20],[17]. It is shown in literature that the hierarchical clustering is energy efficient [17]. The proposed algorithm is independent from clustering algorithms, but the message requirement per mutual exclusion may slightly vary with the choice of clustering algorithm. In this work, we have considered two levels hierarchical clustering to get the best performance level 1 consists of the cluster heads, and one cluster head is chosen as level 2 cluster head. Considering level 2 cluster head as root, the whole network looks like a logical tree. We consider, the nodes in a cluster can directly communicate with the cluster head, and it requires at most p messages for communication between the root and a cluster head. The performance of the algorithm is better if p is low.

The network in Fig.1 consists of 12 nodes. At first we have to cluster this network hierarchically. Here node 2, node 4, node7 and node 11 are the gateway nodes. The hierarchical clustering algorithms are based on two stages; initial and extended. In the initial state, each node announces itself as a cluster head with in its transmission range with probability p. When other nodes received this announcement and if the node itself not a cluster head then it adds itself as a member of the cluster of closest cluster. Distance between the cluster head and the non cluster head node is one hop. So in the initial step, single level clustering is done.

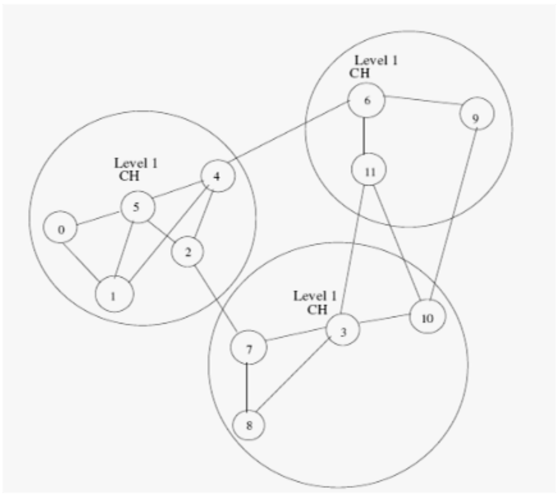


Fig 2: Level 1 clustering

Consider the network of Fig.1. Suppose, node 3, 5 and 6 are selected as cluster heads (level 1). Now, each cluster head advertises itself as a cluster head within its transmission range. As a result, 3 clusters are formed - (3, 7, 8, 10), (5, 0, 1, 2, 4) and (6, 9, 11) (Fig.2). In the extended state multilevel clustering is done. The clustering process is recursively repeated in level of cluster head. As a result, an addition level is added to the cluster. Among the level 1 cluster heads (node 3, 5 and 6), suppose, node 6 is chosen as a level 2 cluster head (root) (Fig.3). We consider level 2 cluster head as the root. If we, therefore, consider the hierarchical structure of the network, then the whole network looks like a logical tree - the root on the top, the cluster heads, and the ordinary nodes under the cluster heads. The corresponding logical tree of Fig.3.1 is shown in Fig.3.4. The proposed algorithm exploits this logical tree structure. Here, the cluster heads 3 and 5 require two messages to communicate with the root (node 6).

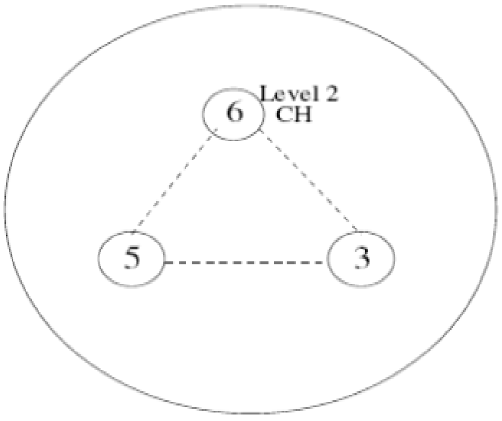


Fig 3: Level 2 clustering with 3cluster heads.

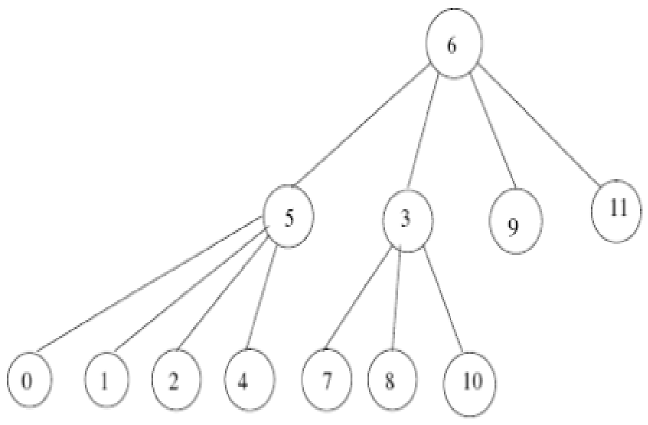


Fig 4: Logical structure of the 12 node MANET

## 5. THE ALGORITHM

This section reports the proposed distributed mutual exclusion algorithm in detail. The algorithm requires a number of data structure and control messages. We next report such requirements.

### *5.1 Data structure and control messages Pointers*:

Each node maintains a pointer which either points to its higher level node or lower level node, if any. The pointer actually provides the direction where from the token can be received. Each non-cluster head member points to its cluster head as the node can get the token from its cluster head only. On the other hand, a cluster head either points to a member of the cluster if the member is having the token, or points to the root.

*Lists:* Each node maintains a list to store the request messages coming to the node or generated by the node for the token. Requests are served according to the request generation time.
*Request:* It is a control message that contains a node number with the time of request.
*Token:* It is a control message which has a special bit sequence and a request message. The request is called dummy request. The dummy request (discussed latter) contains a node ID and the time at which dummy request is sent. However, the dummy request field may be NULL if there is no such request.

### *5.2 Overview of the scheme*

The proposed algorithm is a token based algorithm. A node can execute the critical section (CS) if it gets the token. For simplicity we consider that the number of CS is 1. Since there is a single token in the network, only one node can execute the CS. Each node generates a request message while it wants to execute the CS. The hierarchical structure (tree) of the network helps a node to get the token. Each node points to some other node in the tree where from the node can get the token. The request is forwarded to the pointed node. While a node receives a request message, it forwards the message to some other node through the edges of tree, if the node does not have the token. Otherwise, the node returns back the token. Each node keeps the request in the list, including the self request, if the node can not immediately grant the request by returning the token. The node deletes the request from the list if the token is passed to the requesting node. While the token is passed, the pointers are organized accordingly to point the new token location. After getting the token, the requesting node executes the CS, and after completion of execution the token remains with the node. In the proposed scheme, the token is not circulated if it is not requested. To minimize the number of messages, the algorithm does not always forward the request message. If a node A has already requested the token and during the waiting time another node makes another request to A, then A adds the request in its list but does not forward it.

Consider, a non-cluster head node (A) generates a request message. The node A forwards the request to its cluster head, say C. If C has the free token, it immediately returns back the token to A. If C has token but not free, then the request is added in the list of C. Otherwise, the token can be received either from any non-cluster head member of that cluster, or from the root. The node C can get such direction from the pointer. The request is forwarded according to the content of the pointer. However, the scheme may lead the system to starvation. Since a cluster head or root does not always forward the request, therefore, when a node gets the token, the token may remain with the node for an arbitrary long time. To avoid this situation, we introduce the concept of dummy request in our algorithm.

*Dummy Request:* If a node N forwards the token to some other node and its list is not empty, N puts its ID in the token with the lowest requesting time in the list to form a dummy request. When a node receives a token with dummy request, then the node adds a new request in its list. The token message, therefore, has to have a provision to add a dummy request. The dummy request contains the node ID and the latest time of the pending requests stored in the list.

The following example illustrates the execution of the algorithm.

*Example 1*: let us consider the network of Fig.1 and the corresponding logical structure shown in Fig.4. Here we assume that the number of CS is 1 and the execution time of CS is considered as unit time. We also assume that the token is with node 0. Consider the nodes 0, 7, 8 and 5 make request for token at time 0.51 units, 0.56 units, 0.67 unit and 0.78 unit respectively. While node 0 makes the request for token, the request is immediately granted. The token, therefore, is released at time 1.51. During the execution of CS by node 0, all the requests have arrived in the system. The requests are reached to node 0 following the pointers that direct to the token holding node. For example, node 7 forwards its request to node 3, node 3 forwards the same to node 6, node 6 forwards it to node 5, and finally the request is reached to node 0. The request is stored in the requesting node as well as the intermediate nodes. The scenario is depicted in Fig.5. In the figure, L notes the list of requesting nodes with the time of request (noted in bracket). It can be noted that node 3 gets two requests from node 7 and 8, but it forwards only the first one.

After completion of execution of node 0, it sends the token to node 5. Node 5 then deletes the first node from its list. It is node 6, and after deletion, the list is still non-empty. Then node 5 sends the token to node 6 with a dummy request. Node 6 sends the token to node 3 and also sends a dummy request with the token for the above reason. Node 3 sends the token to the first requesting node 7 with the dummy request. After receiving the token node 7 starts execution of the critical section. The scenario is depicted in Fig.6. In the figure it indicates sending time of request or token.

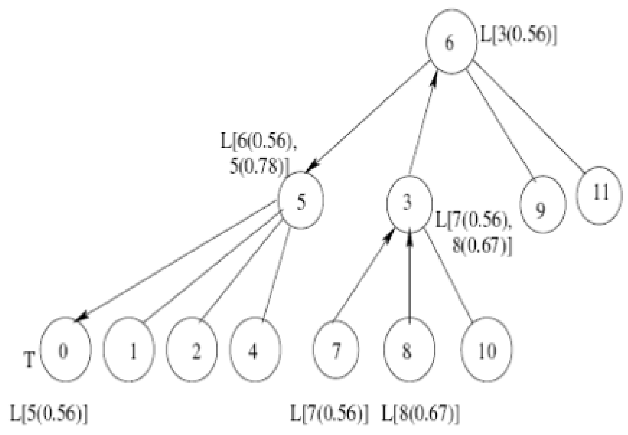


Fig 5: The logical tree after requests of nodes 7, 8 and 5.

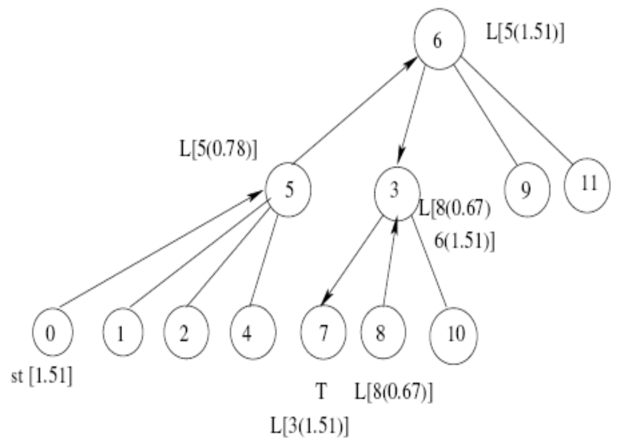
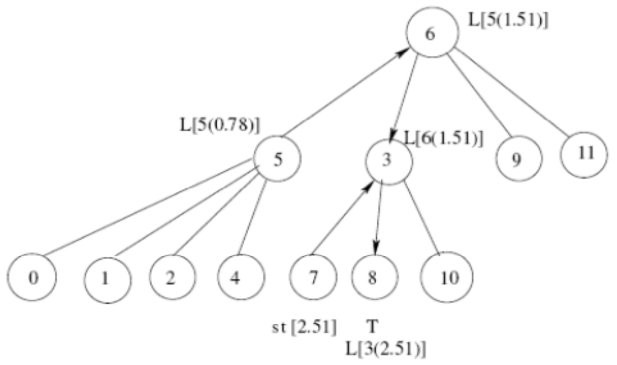


Fig 6: Token is passed to nod



e 7.

Fig 7: Token is passed to node 8.

After completion of execution of the node 7, it returns the token to the node 3 without dummy request. Node 3 sends the token to the node 8 with a dummy request. After receiving the token node 8 will start execution of the critical section (Fig.7).

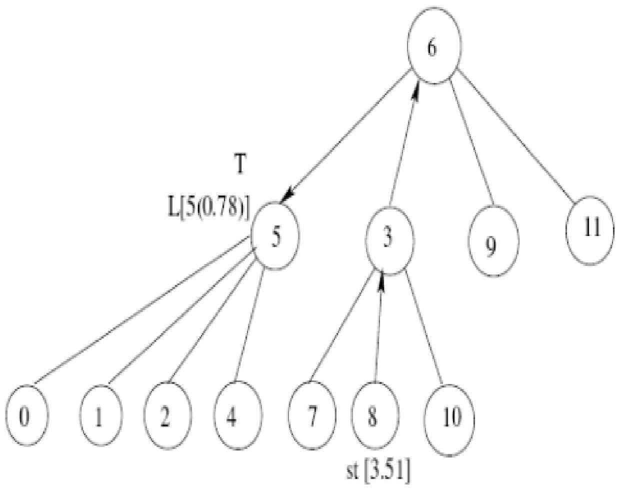


Fig 8: Token is passed to node 5.

After completion of execution, the node 8 sends the token to the node 3. Node 3 sends the token to the first requesting node 6. Node 6 sends the token back to node 5. After receiving the token node 5 starts execution of the critical section (Fig.8).

### *5.3 Mobility and Token regeneration*

Due to the mobility of nodes, the physical as well as the logical topology of a MANET may be altered. Moreover, the token may get lost for the same. However, the proposed algorithm can tolerate the mobility in various ways and is capable to regenerate the token. We discuss the following scenario which may arise due to the node's movement.

*1. A new node is added in the network:* The node joins in some cluster. The node, therefore, is added in the logical tree structure accordingly.

*2. A node moves within the cluster:* It does not affect the proposed scheme as the logical tree structure of the network remains unchanged.

*3. A node moves from one cluster to another cluster:* In this case, following three different actions are to be taken based on the node's type.

*3.a.* The node is non-cluster head node: Due to the movement, the node joins to some other cluster, say Ci. The cluster Ci considers the node as a new node. The node, therefore, submits a fresh request for the token, if required, to the cluster head of Ci. During the movement, if the token is with the node, the token as well as the list, if any, are destroyed. However, if the node is busy in executing the CS, and if it is possible to continue, the node finishes the execution. The corresponding cluster head regenerates the token while expected time of execution has been elapsed.

*3.b.* It is cluster head: While the cluster head of a cluster Ci moves to another cluster Cj (i 6= j), a new cluster head for Ci is selected from the members of Ci. The members of Ci then resubmit their old requests, if any, to the newly selected cluster head. The old cluster head of Ci, now treated as a new node of Cj, destroys the token and the request list, if any. The newly selected cluster head of Ci searches the token within the cluster. To do this, the cluster head issues a request to all its members simultaneously. The request is ignored by an ordinary node if it does not really have the token. If the cluster head does not get the token within the expected time, it concludes that the token is not within the cluster. As a next step, the cluster head asks the token from the root to get only the direction of getting token. Cluster head chooses the oldest time of request from its list and add the time with request. If the token was with Ci, then the root sends a new request to the newly elected clusterhead, of Ci. The root also replaces the previous clusterhead by the new one from its list if it has. So new cluster head gets the token at the time that has been scheduled by previous cluster head. However, if the cluster head gets the similar request from the root, it regenerates the token, concluding the token was within the Ci. In the mean time, the cluster head may get requests from its members also. The token is now passed to the oldest request.

*3.c.* The node is root: The root is also a cluster head. So like previous, the same actions are taken. However, a new root is also chosen from the cluster heads. The root regenerates the token only if the token is not within its own cluster and all other cluster head request the root for the token. Cluster head sends new request to the root and also put the time of oldest request from its list.

*4. An existing node leaves the network:* This is a special case of scenario 3. Here three different situations may also arise depending on the type of the nodes. The node which has left the network is removed from the tree accordingly. If the node is a cluster head or the root, like previous, new cluster head or root is selected. Unlike the last scenario, the node is not treated as a new node of any cluster as the node has already left the network.

*Example 2:* Consider the MANET of Fig.1 and its corresponding logical tree structure, shown in Fig.4. Suppose that while node 7 is executing the CS, then the nodes 8, 10, 2 and 4 submit their requests for token at time 0.2, 0.4, 0.7 and 0.8 respectively. The logical structure of the network as well as the lists formed in intermediate nodes is noted in Fig.9. Let us now consider that the nodes 1, 3, 4 and 7 move and the logical tree structure

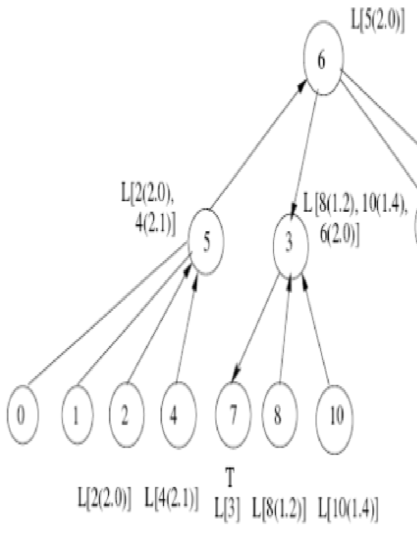


Fig 9: During the 7's execution of CS, 8, 10, 2 and 4 submits their request for token.

is reorganized (Fig.10). Moreover, we consider that during the movement the node 7 continues its execution of the CS. After completion, 7 destroy the token as well as its list (Fig.10). Similarly, the node 3, a cluster head in Fig.9, deletes the whole list and is considered a new node under the cluster head 5 (Fig.10). However, the nodes 1, 4, 8 and 10 forms a new cluster and node 8 is selected as the cluster head. While this is informed to the root, the root asks the node 8 for the token. Since the node 8 cannot get back the token within the expected time, 8 regenerates it. In the mean time, the 8 and 10 resubmit their request. However, since node 4 moves to another cluster head, it generates a fresh request and submits the request to node 8. Now node 8 passes the token according to the ordering of requests (Fig.10). To show the mobility resistance capability of the proposed scheme, we further consider that while the token is utilized by node 10 (Fig.11), the topology of the network is dramatically changed due to the movement of the nodes. Before such movement, the lists of request at various nodes are noted in Fig.11. Almost all the nodes except a few like nodes 2, 4 and 7 change their locations. As a result, the MANET is reclustered, and we get a new logical tree structure (Fig.12). Like previous, here also 3 cluster with the cluster heads 1, 3 and 5 are formed. The node 5 is again selected as the second level cluster head or root. The request lists of the nodes which have moved are destroyed and the token is also destroyed. Since the nodes 2 and 4 have not moved, these nodes resubmit their requests to the corresponding cluster heads (node 3 and 1). Both the requests are forwarded to the root (node 5). However, the root does not have the direction of getting the token. So, the root regenerates the token (Fig.12).

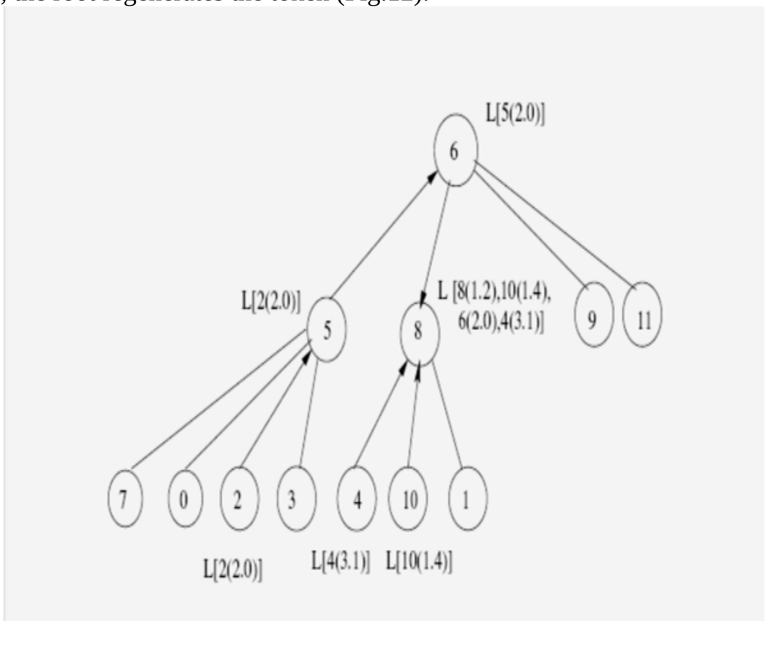


Fig 10: Reorganizations of the tree structure and
the lists at nodes.

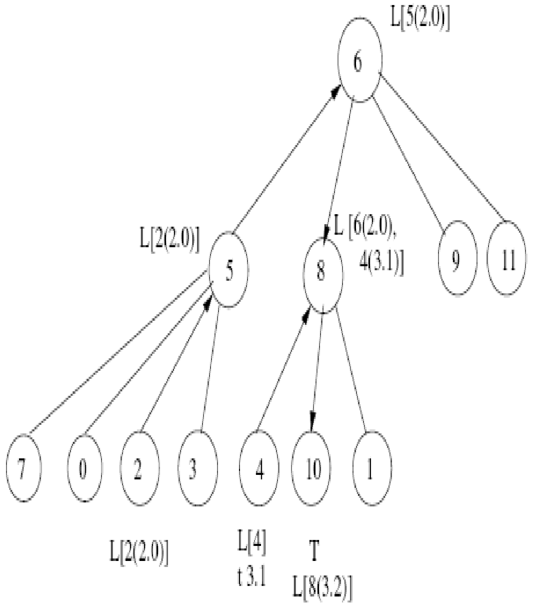


Fig 11: Node 10 is executing the CS, while nodes 2 and 4 are waiting for the token.

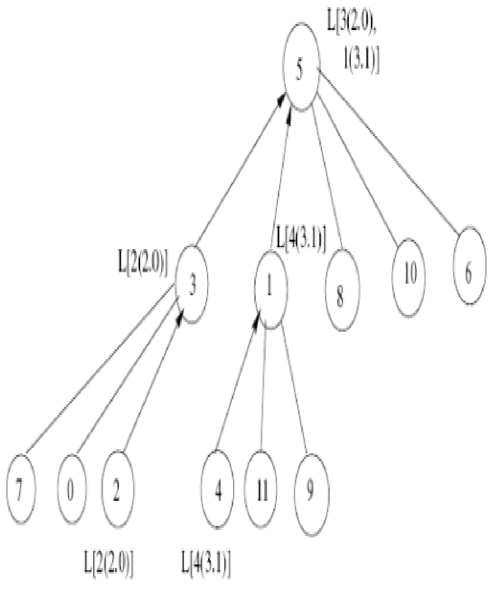


Fig 12: Network is reclustered after movement of nodes. Nodes 2and 4 resubmit their requests.

*5.4 Formal description*

Here we formally present the algorithm. The algorithm assumes the network is hierarchically (two level) clustered to look like a logical tree. Each node in the network is independent and issues the request as well as processes the token independently. The steps of a node (say N) of the algorithm are noted in Algorithm 5.1.
Algorithm 5.1 Steps-performed-by-a-Node

Step 1: If the node, N wants to execute the critical section (CS), then If N contains the token, it executes the CS. Otherwise, form a request message with the request generation time, add the request in its own list and then pass the message to the pointed node.

Step 2: While N receives the request message:

1. If N is an ordinary node in a cluster and does not have the token but its cluster head requests for the token, then the request is ignored.
2. If the node has free token, the token is returned to the node where from N receives the token.
3. Otherwise, the request (node number with requesting time) is appended in its list, and N issues a request to the pointed node if it has already not requested for the token.

Step 3: The node N receives the token:

1. If the token contains a dummy request, the request is added in its list.
2. If the list is empty, the token remains with the node.
3. Otherwise, pick up a node, say q, from the list having minimum requesting time.
4. If q = N, the node N executes the CS.
5. Otherwise, the token is sent back to q. If the list is still non-empty after deletion of q, a dummy request is added with the token.

Step 4: The node N leaves the cluster:

1. If N is in execution of CS, it continues the same, if possible.
2. N destroys the token as well as the list.
3. If N is now within the transmission range of some cluster, N considers itself as a member of the cluster. The N submits a fresh request for token to its present cluster head, if required.

Step 5: N identifies that the cluster head/root has left the cluster:

1. N as an ordinary node takes part to select a new cluster head.
2. If N is a cluster head and the root has left the cluster/network, N with other cluster heads select a new second level cluster head or root.

Step 6: If N is a newly selected cluster head, then:

1. N issues requests for token to all its cluster members simultaneously, and waits to get the token for an estimated time of executing the CS.
2. If N gets the token within the estimated time, the node keeps the token.
3. Otherwise, N issues a request for the token to the root. If the same request is received by N from the root, N regenerates the token.
4. If N itself is the root, the node N regenerates the token when it does not get the token from its cluster members and is requested by all other cluster heads.

## 6. CONCLUSION

In this report, we have presented two token based mutual exclusion algorithms for mobile ad-hoc networks (MANET). For the second algorithm, the network is hierarchically (two level) clustered to get a logical tree structure. The proposed algorithm works on the logical tree network. The message complexity of the algorithm in worst case is 4 + 4p, where p is the maximum message requirement for the communication between a cluster head and the root (second level cluster head). The proposed algorithm can tolerate the mobility of nodes in a wide extent.

## 7. REFERENCES

[1] Anantha P. Chandrakasan and Arthur C. Smith and Wendi Beth Heinzelman and Wendi Beth Heinzelman. An application - specific protocol architecture for wireless micro sensor networks. IEEE Transactions on Wireless Communications, 1:660{670, 2002.

[2] Ranganath Atreya, Neeraj Mittal, and Sathya Peri. A quorum-based group mutual exclusion algorithm for a distributed system with dynamic group set. IEEE Trans. Parallel Distributed Syst., 18(10):1345{1360, 2007.

[3] B. R. Badrinath, Arup Acharya, and Tomasz Imielinski. Structuring distributed algorithms for mobile hosts. In ICDCS, pages 21 {28, 1994}.

[4] R. Baldoni and A. Virgilito. A token - based mutual exclusion algorithm for mobile ad-hoc networks. In Dipartimento di Informatica e Sistemistica, Universita di Roma La Sapienza, Viasalaria 113, 00198 Roma, Italia, Technical Report 28-01.

[5] S. Bandyopadhyay and E. Coyle. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003), San Francisco, California, April 2003.

[6] Pranay Chaudhuri and Mehmet Hakan Karaata. An o(n1/3) algorithm for distributed mutual exclusion. Journal of Systems Architecture, 45(5):409{420, 1998.

[7] M. Hossein Fotouhi Ghazvini, Maryam Vahabi, Mohd Fadlee A. Rasid, and Raja Syamsul Azmir Raja Abdullah. Optimizing energy consumption in hierarchical clustering algorithm for wireless sensor networks. In Proceedings of the 2007

[8] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. Commun. ACM, 21(7):558{565, 1978.

[9] Hsiou.-Mien. Lien and Shyan.-Ming. Yuan. A new approach of constructing information structure for mutual exclusion in distributed systems. In Proceedings of the 1994 International Conference on Parallel and Distributed Systems, pages 588{591, Washington, DC, USA, 1994. IEEE Computer Society.

[10] Sandeep Lodha and Ajay Kshemkalyani. A fair distributed mutual exclusion algorithm. IEEE Trans. Parallel Distrib. Syst., 11(6):537{549, 2000.

[11] Mamoru Maekawa. A pN algorithm for mutual exclusion in decentralized systems. ACM Trans. Comput. Syst., 3(2):145{159, 1985.

[12] Navneet Malpani, Yu Chen, Nitin H. Vaidya, and Jennifer L. Welch. Distributed token circulation in mobile ad hoc networks. IEEE Transactions on Mobile Computing, 4(2):154{165, 2005.

[13] Romain Mellier and Jean-Frederic Myoupo. A clustering mutual exclusion protocol for multi-hop mobile ad hoc networks. In IEEE International Conference on Networks (ICON 2005), IEEE Press, pages 250{255, 2005.

[14] Kerry Raymond. A tree-based algorithm for distributed mutual exclusion. ACM Trans. Comput. Syst., 7(1):61{77, 1989.

[15] Glenn Ricart and Ashok K. Agrawala. An optimal algorithm for mutual exclusion in computer networks. Commun. ACM, 24(1):9{17, 1981.

[16] Ichiro Suzuki and Tadao Kasami. A distributed mutual exclusion algorithm. ACM Trans. Comput. Syst., 3(4):344{349, 1985.

[17] Farzad Tashtarian, A. T. Haghighat, Mohsen Tolou Honary, and Hamid Shokrzadeh. A new energy - efficient clustering algorithm for wireless sensor networks. In Proc. of Software, Telecommunications and Computer Networks, Soft- COM 2007, pages 1{6, September 2007.

[18] Jennifer Walter and Savita Kini. Mutual exclusion on multihop, mobile wireless networks. Technical report, College Station, TX, USA, Dec 1997.

[19] Jennifer E. Walter, Jennifer L. Welch, and Nitin H. Vaidya. A mutual exclusion algorithm for ad hoc mobile networks. Wirel. Netw., 7(6):585{600, 2001.

[20] Hongwei Zhang and Anish Arora. Gs3: scalable self-configuration and self-healing in wireless sensor networks. Comput. Netw., 43(4):459{480, 2003.

[21] P. H. Enslow. What is a "distributed" data processing system? Computer, 11(1):13{21, 1978.