A Permission-based Clustering Mutual Exclusion Algorithm for Mobile Ad-Hoc Networks

Abhilasha Gupta*, B.V.R. Reddy **, Udayan Ghosh*, Ashish Khanna******
*, **, *** (University School of Information Technology, Guru Gobind Singh Inderprastha University, New Delhi)
**** (M.A.I.T, Guru Gobind Singh Inderprastha University, New Delhi)

ABSTRACT

In the last ten years a lot of research has been done on resource allocation in Ad-hoc networks. The Classical approaches of mutual exclusion and its variants need to be modified to suit the dynamic topology, low bandwidth and low processing capabilities of mobile ad-hoc network (MANET).The distributed mutual exclusion in MANETs is comparatively less explored area of research. In this paper, we propose a new approach for mutual exclusion in MANETs which is based on clustering and the concept of weight throwing. The algorithm uses cluster based hierarchal approach which also helps in reducing the message complexity of the algorithm.

Keywords - **Ad-hoc network, Clustering, Critical Section, Mutual Exclusion (MUTEX), Voting**

1. Introduction

A mobile ad- hoc networks is a network which has no fixed infrastructure and is combination of mobile nodes and some immobile infrastructure. Ad-hoc network has dynamic topology. Nodes in ad-hoc system can communicate directly only with the nodes that are immediately within their transmission range. To communicate with the other nodes, an intermediate node is required to forward the packet from the source to the destination. Therefore, in ad hoc system, nodes are required to cooperate in order to maintain connectivity and each node may act as a router in routing data through the network.
Commonly suggested applications for MANETs include disaster management, Battlefields, and environmental data collection. Although lots of hardware challenges have been solved, programming application for MANETs remains a tedious task.

The resource allocation problem is one of the most important problems in MANETs. However, according to Badrinath-Acharya-Imelinski [1], due to the special characteristics of mobile computing environment, the algorithms proposed for static distributed systems needs to be modified before these can be applied in mobile computing environment. For that purpose, they proposed two-tier principle to restructure the distributed algorithms to make them suitable for mobile environment. Moreover, MANETs are an important class of mobile computing systems and because of its infrastructure less nature two-tier principal cannot be applied directly to MANETs. Hence, the algorithms required to solve a resource allocation problem in MANETs, has to be designed considering the special characteristics of MANETs.
Mutual exclusion (MUTEX) is a fundamental problem in distributed systems, where collections of nodes intermittently need entering the Critical Section (CS) in order to exclusively process few critical operations, e.g. accessing the shared resource. A solution to the MUTEX problem must satisfy the following three correctness properties:
(i) Mutual Exclusion (safety): No two processes can be inside their CS simultaneously.
(ii) Deadlock Free (liveness): At any point of time, at least one node able to take an action and enter CS.
(iii) Starvation Free (Fairness): Every node wanting to enter CS must eventually be able to enter CS.
The performance of a mutual exclusion (ME) algorithm can be judged based upon various performance parameters like ***Waiting time***, ***Synchronization delay***, ***Message complexity***, ***Message size*** [2].
Permission-based algorithms [3] need cycles of message exchange among the nodes to get the permission to enter CS. The main concept on which permission-based algorithms are based is as follows: When a process wants to execute its CS, it sends request to other nodes for their permission. A process on getting a request, it grants permission if it is not interested in CS. If it is interested in CS, the priority of the incoming request is located against its own request. Commonly, priority decisions totally depend upon the timestamps. Total ordering of events is done with the help of Lamport's [4] logical clocks for having clear time difference between the request time

stamps. Permission-based algorithms can be further classified into coterie-based algorithms and voting-based algorithms. In voting-based algorithms, each node in the current system is allocated a vote (a nonnegative integer). A node wanting access to CS must get permission from a suitable number of nodes, i.e., a number of nodes whose total votes comprises a majority of the total number of votes allocated to the system. In coterie-based algorithms, a group of sets (of the nodes of the system), called a coterie, is pin to the system. A node wanting access to CS must get permission from each and every node of a set from the coterie. Each of these two categories may further be classified into static and dynamic algorithms .The paper [5] is proposed for solving the Group mutual exclusion (GME) by using clustering concept. In [6] [7], R.Mellier et J-F. Myoupo and Stefano Basagni have presented a MUTEX protocol for MANETs which takes advantages of the cluster structure offered by the partitioning techniques.

This paper is organized as follow: the section 2 discusses related work and the basic idea of clustering concept. Section 3 presents our proposed algorithm & its pseudo code. We prove the algorithm correctness in section 4. Section 5 gives the performance analysis of proposed algorithm. Conclusion and future work are offered in section 6.

.2. Related work

The origin of the mutual exclusion problem can be traced back to 1965 when Dijkstra [8] described and solved the mutual exclusion problem. Dijkstra stated that any solution to the mutual exclusion problem must satisfy 4 constraints. Dijkstra’s algorithm guaranteed mutual exclusion and was deadlock- free, but it did not guarantee fairness. That is, it was possible that a process seeking access to its critical section waited indefinitely while others entered and exited their critical sections frequently. Knuth [9] proposed the first fair solution to the mutual exclusion problem. Thereafter, a number of algorithms were proposed which guaranteed mutual exclusion and were deadlock- free and fair. Each of these algorithms aimed at improving performance in terms of synchronization delay, the period of time between the instant a site invokes mutual exclusion and the instant when it enters CS. Some of these algorithms are De Bruijns’s algorithm [10], Eisenberg and McGuire’s algorithm [11] and Peterson’s algorithm [12]. All these algorithms were designed for the centralized systems, the systems possessing a central memory that all processes can access simultaneously for reading and writing.

A number of DME algorithms have been developed, all aiming at enhanced performance with respect to one performance metric or the other. Based on the technique used, DME algorithms can be classified as token based algorithms and permission-based algorithms as suggested by Raynal [13], or as token-based algorithms and non-token-based algorithms as suggested by Singhal [14].In token-based algorithms, a token is passed among all the nodes. A node is allowed to enter the CS only if it possesses the token. In a permission-based algorithm, the node requesting for the CS must first obtain the permissions from other nodes by exchanging messages. Some examples of token-based algorithms are Helary et al.’s [15] and Suzuki and Kasami’s [16] algorithms (broadcast based, static), Singhal’s [17] and Yan et al.’s [18] algorithms (broadcast-based, dynamic), Raymond’s [19] and Neilson and Mizuno’s [20] algorithms (logical structure-based, static) and Chang et al.’s [21], Helary et al.’s [22] and Naimi et al.’s [23] algorithms (logical structure-based, dynamic).

During the past several years, algorithms for solving the mutual exclusion problem in MANETs have been proposed. The entire algorithm makes use of a token circulated along a logical ring or passed in a logical tree consisting of all the nodes.

A token-based mutual exclusion algorithm, named RL (Reverse link) [24], for ad-hoc network is proposed. In the RL algorithm, when a node wishes to access the shared resource, it sends a request message along one of the communication link. The RL algorithm totally orders nodes so that the lowest ordered node is always the token holder. The algorithm guarantees the safety and liveness property. But it did not guarantee the network partitioning. Malpani et al [25] proposed a parametric token based algorithm with many variations. In the algorithm, a dynamic logical ring is imposed on the nodes the successor of a node in the ring is computed on- the-fly.

Weigang Wu et al [26] proposed the first permission-based MUTEX algorithm for MANETs. In order to reduce the message cost, the algorithm uses the so called”look-ahead” technique, which enforces MUTEX only among the hosts currently competing for the critical section (C.S). The algorithm is based on the well-known Ricart-Agrawala algorithm [27] in which, when a nodes wants to enter CS, it sends a request to all the other nodes to collect permissions.

Several MUTEX algorithms for MANETs have been proposed and nearly all of them use the token- based approach [28] [29] [25] [30]. Compared with the permission-based approach [26], the token-based approach has many desirable features, e.g. nodes only need to keep the information about their neighbors and few messages are needed to pass the privilege of entering CS. Both token-circulating (ring-based) and token-asking (tree-based or graph-based) approaches

have been used in MUTEX algorithm for MANETS. The fatal problem of token loss makes these algorithms not robust. In MANETs, the mobility and disconnections of nodes make token loss a more serious problem and the maintenance of a tree or ring topology more difficult.
Compared with the token-based approach, permission based algorithm have the following advantages:-

- There is no need to maintain the logical topology to pass the token.
- There is no need to propagate any message if no node request to enter CS.

These advantages make the permission-based approach will suitable for MANETs where all the resources, e.g. the network bandwidth and the battery power of the nodes are limited.
A problem of the permission-based approach is the large number of message to be exchanged between the nodes. Therefore to design a efficient permission-based algorithm , we use the "clustering concept", in which only clusterleader of the respective cluster is responsible for taking & giving the permission to enter the CS, which reduce the number of message exchanged among the clusterleaders.

2.1Clustering Concept

Partitioning the nodes in to cluster is called clustering. In addition, clustering is crucial for managing the spatial reuse of the shared channel, for reducing the amount of data to be exchanged in order to maintain routing and control information in a mobile environment, as well as for constructing and maintaining cluster-based virtual network architecture. In existing solutions for clustering of ad-hoc networks [5], the clustering is performed in two phases: clustering initialization and clustering maintenance. Each cluster comprises clusterleader and in-range nodes (direct communicates with its own clusterleader).A clustering algorithm is required to partition the nodes of the network so that the following *ad-hoc clustering properties* are satisfied:

I. Every in-range node has at least a clusterleader as neighbor (dominance property).
II. Every in-range node affiliates with the neighboring clusterleader that has the bigger weight.
III. No two clusterleader can be neighbors (independence property)

Election of clusterleader depends either on the basis of lower id or on the basis of weight. Weight based criteria is better way of deciding clusterleader rather than deciding it on the basis of lower id. The heavier the weight of a node, the better that node for the role of clusterleader [5].The main benefit of this approach is that, by representing with the weights mobility-related parameters of the nodes, we can pick for the role of clusterleader those nodes that are better suited for that role. For instance, when the weight of a node is inversely proportional to its speed, the less mobile nodes are confirmed to be clusterleader. Since these nodes either do not moves or move lesser than the other nodes, their cluster is guaranteed to have a durable life, and consequently the overhead attached with the cluster maintenance in the mobile environment is reduced.

2.2 Initialization of clusters

Initially out of all nodes heaviest weight node is chosen, after which nodes in direct range of that node forms a cluster and this process is repeated till all nodes are part of any cluster.

3. ALGORITHM

The algorithm is assumed to execute in a system consisting of n nodes and m clusters, each cluster contains one clusterleader. Nodes are labeled as 0, 1…….n-1, and clusterleaders are labeled as 0, 1……m-1. We assume there is a unique time-stamp generated with every "Req_CS" message by node i.

3.1 Assumptions

This algorithm takes the following assumptions on the mobile nodes and clusters.

- All nodes have unique ids.
- No new cluster will be formed after initialization.
- A link level protocol ensures that each node is aware of the set of nodes with which it can currently communicate by providing indications of link formation and failures.
- Global synchronize clock is maintained on each node.
- Each node has inbuilt singleton vote.
- Node can move in their respective cluster only.
- No node can move while the cluster formation is in progress.
- Each node has a different weight.
- No two clusters overlap with each other.

3.2 Requirements

As all nodes have singleton vote allocated to them, so a node can enter C.S only when its respective clusterleader acquire more than 50% of singleton vote of whole system. Equation (1) should be met accordingly if value of weight is even or odd.

$$Maj\zeta = \begin{cases} \frac{1}{2}\,Tot\zeta + 1 & \text{if } \sum Tot \text{ is even} \\ \frac{1}{2}\,(Tot\zeta + 1) & \text{if } \sum Tot \text{ is odd} \end{cases}$$

Where Majζ: number of majority weighted votes in the system.

$Tot\zeta = \sum_{i=1}^{N} v_i$ Gives the total number of singleton votes in the system.

V_i = singleton vote of node i.

3.3 Data structures

3.3.1Data structures of clusterleader

1. Status: - indicate whether node is in the Remainder, Waiting or C.S. Initially status = Remainder
2. Clust_id$_i$:- id of a particular clusterleader.
3. Clust_V$_i$:- Each clusterleader has singleton vote. Singleton vote associated to Clusterleader is used it to give permission.
4. Clust_Rq$_i$:- Request list stores the request of all nodes of cluster's and requests Received by other clusterleaders of their respective nodes with their unique time stamp.
5. Cluster$_{i:}$ - The set of all nodes id in direct wireless contact with cluster$_i$.
6. Allower_info$_i$:- The set of all clusterleaders with their respective singleton vote which have allowed the requesting cluster to enter C.S.
7. Clust_Tot_Vote:- total singleton vote of in-range nodes of cluster$_{i.}$
   SYS_Tot_Vote:- total singleton vote of the system.
8. Allow:- Boolean array indicating whether the cluesterleader has given permission to requesting node or not.

Initially Allow ≠ TRUE (has not given permission)

Allow = TRUE (has given permission)

3.3.2 Data structures of node$_i$

1. N_id$_i$:- Each node has unique identifier.
2. N_V$_i$:- The singleton vote allocated to node i.
3. T_Sreq :- Unique time stamp at which the request is generated that is also used to set the priority of request with which they will be served.
4. Status:- indicate whether node is in the Remainder, Waiting or C.S. Initially status = Remainder
5. Clusterleader_info$_i$:-Clusterleader_info$_i$ list maintain the information about cluster$_i$

3.4 Message used in the algorithm

- *Req_CS():* when a node i whishes to enter the CS. It send out Req_CS() to the its own clusterleader.
- *Cluster_Vote:* a message used for clusterleaders to transfer their singleton vote for giving the permission to enter the CS to requesting clusterleader.
- *Allow_CS ():* a message for node to enter the CS. This message is received by requesting node from its own clusterleader.
- *Release ():* a message for node i to release the CS, it sends Release () to the own clusterleader.
- *Release_Cluster_Vote:* when a requested clusterleader's request queue = φ, then it will return their vote to the respective clusterleader which has given permission.

3.5 Principle of the algorithm

In this paper, a mutual exclusion algorithm is proposed which is permission based. This algorithm works on the concept of clustering and uses a voting based hierarchal approach. Here initialization is done on the basis of *weight throwing scheme* [5]. Where one by one cluster are made till the last node is part of any cluster and each cluster has one clusterleader. Total number of nodes in the cluster decides the number of votes acquired by any clusterleader. Each node in the system can send its request only to its respective clusterleader which further forwards that to the other clusterleader for getting more than half number of votes to allow its requesting node to enter to enter C.S. Each clusterleader maintains request list of whole system. Request generated has unique time stamp dependency upon lamport's logical clock [4]. The proposed algorithm is event-driven. An event at node i consists of receiving a message from clusterleader. Each event triggers a procedure, which is assumed to be executed atomically. Below, we present the overview of the event-driven procedure.

- ***Node i wants to enter CS:***

When node i want to enter the CS, it first sets T_Sreq to the current time and sends the "Req_CS" message to the clusterleader and sets status to waiting.

- ***Cluster leader i receives request from node j:***

When a Req_CS (j, T_Sreq) message sent by a node j is received at clusterleader i.

Request stored in request-queue of clusterleader i with its time-stamp and node id. if the singleton vote of cluster i is greater than half of the total system singleton vote and Allow ≠ TRUE then clusterleader i sends "Allow_CS" message to requesting node j and sets Allow = TRUE otherwise it sends Req_CS($n_{id,}$T_Sreq) to all clusterleader.

- ***Cluster leader i receives request from clusterleader j***

When a Req_CS (n_{id}, T_Sreq) message sent by a clusterleader j is received at clusterleader i, if the request list of clusterleader i is empty and Allow ≠ TRUE then its sends the Cluster_Vote (Clust_id, Clust_Tot_Vote) to requesting clusterleader j and sets $Clust\_Tot\_Vote_i = 0$, sets Allow = TRUE otherwise insert the request in the request list at its appropriate position after sorting the list.

- ***Clusterleader i receives singleton vote message from Clusterleader j.***

When Cluster_Vote (Clust_id, Clust_Tot_Vote) message sent by a clusterleader j is received by clusterleader i. clusterleader i increments the value of its own singleton vote by adding the singleton vote of cluster j. if the singleton vote of clusterleader i is greater than the half of the total singleton vote of the system and Allow ≠ TRUE then it sends the Allow_CS message to lowest T_Sreq node and set Allow = TRUE otherwise wait for the new request.

- ***Node i receives Allow_CS message from clusterleader j:***

When node i received Allow_CS message from clusterleader j. node i sets status = C.S. and enter C.S.. After processing it come out from C.S.

- ***Node i exits from the C.S:***

When node i comes out from the C.S. it sends "Release" message to clusterleader j and sets status = Remainder.

- ***Clusterleader i receives release message from node j:***

When "Release" message sent by a node j is received by clusterleader i. firstly clusterleader sets Allow ≠ TRUE after that it checks request list and send "Release_Cluster_Vote" message to the other clusterleader or to its any other node whichever is having lowest time stamp.

- ***"Release_Cluster_Vote" message is received by clusterleader i.***

When "Release_Cluster_Vote" message is received by clusterleader i. clusterleader i restore the value if its own singletons vote. If the request list of clusterleader i is not empty and the singleton vote of clusterleader i is greater than the total singleton vote of the system and Allow ≠ TRUE then it sends "Allow_CS" message to that node having lowest T_Sreq.

Pseudo code: Clustering Permission based MUTEX algorithm, code for n_i.

1. **Node i wants to enter C.S:**

Initially when node i wants to enter C.S then timestamp at which request is generated is also stored and forwarded to respective clusterleader.

```
$T\_Sreq_i = t_1$ (current value of clock)
State: = Waiting

Node i send Req_CS ($n_{id}$, T_Sreq) to clusterleader j.
```

2. **Clusterleader i receives request from node j**

```
Request stored in request_list of clusterleader with its timestamp and node id.

If (Clust_Tot_Vote > ½ SYS_Tot_Vote) && Allow ≠ TRUE
{
Send Allow_CS() to node j
Allow: = TRUE
}
Else
{
Send Req_CS ($n_{id}$, $T\_Sreq_i$) to all clusterleader.
}
```

3. **Clusterleader i receives request from clusterleader j**

```
If ($Cluster\_req_i = \varphi$ && Allow ≠ TRUE)
{
Send Cluster_ Vote (Clust_Tot_Vote) to clusterleader j
Set Clust_Tot_Vote = 0
Set Allow = TRUE

// comparison made between the time-stamp of arrived request and request which is positioned at the top on the list //
Else if ( $T\_Sreq_i < T\_Sreq_i$)
{
Send Cluster_Vote (Clust_Tot_Vote) to clusterleader j
Else
{
Insert the request in the request list at its appropriate position after sorting the list.
}
}
}
```

4. **Cluster leader i receives vote message from clusterleader j.**

```
// clusterleader i updates the value of its own singleton vote//

$Clust\_Tot\_Vote_i=Clust\_Tot\_Vote_{i+}\ Clust\_Tot\_Vote_j$
```

```
(Current)                    (Received
vote)

If(Clust_Tot_Vote$_i$ > ½ SYS_Tot_Vote && Allow
≠ TRUE)
{
Send Allow_CS () to lowest T_Sreq node.
Set Allow = TRUE
Else
{
Wait for additional singleton vote
}
}
```

5. **Node i receives allow message from clusterleader j**

```
State = C.S
Enter C.S
Exit C.S
```

6. **Node i exits from the C.S**

```
Node i send_ Release () to clusterleader j
State: = Remainder
```

7. **Clusterleader i receives release message from node j**

```
Set Allow ≠ TRUE
```

Checks request queue and send Release_Cluster_Vote () to the other clusterleader or to its any other node whichever is having lowest time stamp.

8. **When Release_Cluster_Vote () message is received by clusterleader i.**

```
Clust_Tot_Vote$_i$ = Clust_Tot_Vote$_i$ (received vote)
(Current vote)


If (Clust_Tot_Vote$_i$ > ½ SYS_Tot_Vote && Allow ≠
TRUE)
{
Send Allow_CS () to that node having lowest
T_Sreq.
}

Else
{
Wait for the new request
}
```

4. Correctness of the proposed algorithm

In this section we prove the correctness of the proposed algorithm that the three correctness requirements for distributed MUTEX algorithm are satisfied.

Lemma 5.1 Once the node i want to enter the C.S. it eventually gets the access of C.S. [4].

Node i enter the CS when the following three conditions are satisfied:

L1: Clusterleader of n_i has received a message with timestamp larger than (T_Sreq$_i$, i) from all other clusterleader.

L2: Node n_i has lowest timestamp.

L3: Respective Cluster leader should have majority votes
(Equation 1)

Theorem 1: At most one node can be in the CS at any time (*safety*).

Argument: we prove the theorem by contradiction. Assuming two nodes n_i and n_j are executing the CS simultaneously. From equation (1), every requesting node must have more than 50% of singleton votes of the total system vote. If both n_i and n_j are in CS simultaneously, means both have 51% of majority singleton votes which sums up to102%. However, total singleton votes of the system can never be exceeds by 100%.This is a contradiction.

Theorem 2: The algorithm is deadlock- free (*liveness*).

Argument: A deadlock occurs when there is a circular wait and there is no “RELEASE” in transit. This means that each node in the cycle is waiting for a “RELEASE” from its successor node in the respective request queue. According to our assumption each node has unique timestamp. Our algorithm says eventually each request with unique timestamp will be added to every request list. Therefore we can say that request list is maintained globally. Above discussion proves that to allow a node to enter CS will be a global decision without any deadlock.

Theorem 3: The algorithm is starvation-free (*fairness*).

Proof A proposed mutual exclusion algorithm for MANETs is fair if the requests for CS are executed in the order of their timestamps [4]. Whenever a node request for CS its request is forwarded to all the cluster leaders eventually with the respective unique timestamp, therefore clusterleaders are able to come to the global decision that which requests time stamp is lowest. In response to this decision eventually every request gets fair chance to enter CS in order. The proof is by contradiction. Suppose a node n_i's request has a smaller timestamp than the request of another node n_j and n_j is able to execute the CS before n_i. For n_j to execute the CS, it has to satisfy the conditions L1 L2 and L3 (by lemma 5.1). This implies that at some instant in time n_j has its own request at the top of its queue and it has also received a message with timestamp larger than the timestamp of its request from all other nodes. But request_list at a node is ordered by timestamp, and according to our assumption n_i has lower timestamp. So n_i's request must be placed ahead of the n_j's request in the request_list_j. This is a contradiction. Hence this algorithm is starvation-free mutual exclusion algorithm.

5. Performance Analysis of Algorithm

In this section, the performance of proposed algorithm has been analyzed with respect to the following performance metrics, namely, message complexity, message size, waiting time, synchronization delay.

Message size: The size of message used in proposed algorithm has been given in the table (1).

Message type	**Size**
Req_CS	O(1)
Allow_CS	O(1)
Cluster_Vote	O(m)
Release_CS	O(1)
Release_Cluster_Vote	O(m)

Table (1)

Performance parameter: the performance parameter used in proposed algorithm has been given in the table (2)

Performance parameter	**Best case**	**Average case**	**Worst case**
Waiting time	2T		2T(m+1)
Message complexity	0 if clusterleader itself. O(2) if any node		O(n)
Synchronization delay	2T		2T(m+1)

Table (2)

Where, T is the maximum message propagation delay.

m is the number of clusterleader.

n is the number of nodes.

6. Conclusion and Future Work

In this paper, we described a permission-based clustering mutual exclusion algorithm in mobile ad-hoc networks. To reduce the number of messages exchanged, the "Clustering concept" is used. This algorithm is independent from logical topology so as to reduce the cost of maintaining logical topology. Simulation is left as a future work.

REFERENCES

[1] B. R. Badrinath, A. Achaya, and T. Imielinski, Desinging Distributed Algorithm for Mobile Computing Networks. *Computer Communication, Vol. 19, No. 4, April 1996.*

[2] A.Swaroop, *Efficient group mutual exclusion protocols for message passing distributed computing system,* doctoral diss, National institute of technology, Kurukshetra, India, 2009.

[3] Saxena, P.C., Rai, J., A survey of permission-based distributed mutual exclusion algorithms. *Computer standards & interfaces, vol. 25, no. 2, pp. 159-181, 2003.*

[4] Lamport, L., Time, clocks, and the ordering of events in a distributed systems. *Communications of the ACM, vol. 21, no. 7, pp. 558-565, 1978.*

[5] Ousmane thiare and Mohamed Naimi. "A Weight- throwing Clustering Group Mutual Exclusion Algorithm for Mobile Ad Hoc Networks" *International Journal of Digital Content Technology and its Applications. Volume 5, Number 4, April 2011.*

[6] Romain Mellier and Jean-Frederic Myoupo, "A clustering mutual exclusion protocol for multi-hop mobile ad hoc networks", *Networks, 2005. Jointly held with the 2005 IEEE 7th Malaysia International Conference on Communication., 13th IEEE International Conference*, 6 pages, 2005.

[7] Stefano Basagni, "Distributed clustering for ad hoc networks", *Proceedings of I-SPAN, Australia 1999.*

[8] E.W. Dijkstra, Co-operating sequential processes, *in: F. Genuys (Ed.), Programming Languages, Academic Press, New York, pp. 43–112, 1965.*

[9] D.E. Knuth, Additional comments on a problem in concurrent programming control, *Communications of the ACM 9 (5), pp.321–322, 1966.*

[10] J.G. De Bruijn, Additional comments on a problem in concurrent programming control, *Communications of the ACM 10 (3), pp.137–138, 1967.*

[11] M.A. Eisenberg, M.R. McGuire, Further comments on Dijkstra's concurrent programming control problem, *Communications of the ACM 15 (11) (1972) 999.*

[12] G.L. Peterson, Myths about the mutual exclusion problem, *Information Processing Letters 12 (3), pp.115– 116, 1981.*

[13] M. Raynal, A simple taxonomy for distributed mutual exclusion algorithms, *ACM Operating Systems Review 23 (2), pp. 47–51, 1991.*

[14] M. Singhal, taxonomy of distributed mutual exclusion, *Journal of Parallel and Distributed Computing 18, pp. 94–101, 1993.*

[15] M. Helary, N. Plouzeau, M. Raynal, A distributed algorithm for mutual exclusion in an arbitrary network, *The Computer Journal 31 (4), pp. 289– 295, 1988.*

[16] I. Suzuki, T. Kasami, A distributed mutual exclusion algorithm, *ACM Transactions on Computer Systems 3 (4), pp. 344–349, 1985.*

[17] M. Singhal, A heuristically aided algorithm for mutual exclusion in distributed systems, *IEEE Transactions on Computers 38 (8), pp.651–661, 1989.*

[18] Y. Yan, X. Zhang, H. Yang, A fast token chasing mutual exclusion algorithm in arbitrary network topologies, *Journal of Parallel and Distributed Computing 35,pp.156–172,1996.*

[19] K. Raymond, A tree based algorithm for distributed mutual exclusion algorithms, *ACM Transactions on Computer Systems 7 (1), pp. 61– 77, 1989.*

[20] L.N. Neilson, M. Mizuno, A DAG based algorithm for distributed mutual exclusion, *International Conference on Distributed Computer Systems, pp. 354– 360, 1991.*

[21] Y.I. Chang, M. Singhal, M.T. Liu, An improved o(log(n))mutual exclusion algorithm for distributed systems, *International Conference on Parallel Processing,pp.295–302,1990.*

[22] M. Helary, A. Mostefaoui, M. Raynal, A general scheme for token and tree based distributed mutual exclusion algorithms, *IEEE Transactions on Parallel and Distributed Systems 5 (11),pp. 1185– 1196,1994.*

[23] M. Naimi, M. Trehel, A. Arnold, A log(n) distributed mutual exclusion algorithm based on path reversal, *Journal of Parallel and Distributed Computing 34,pp.1 – 13,1996.*

[24] Jennifer Walter, Jennifer Welch and Nitin Vaidya, "A mutual exclusion algorithm for ad hoc mobile network", *In Journal of Wireless Networks, vol. 7, pp. 585-600, 2001.*

[25] N.Malpani, N. H. Vaidya and J. L. Welch, Distributed Token Circulation on Mobile Ad Hoc Networks, *Technical report, Intel Corporation 505 E. Huntland Dr. Suite 550, Austin TX 78752.*

[26] Wu, W., Cao, J., Yang, J.: A Scalable Mutual Exclusion Algorithm for Mobile Ad Hoc Networks. *In: Proc. of ICCCN (2005).*

[27] G. Ricart and A. Agrawala, An optimal algorithm for mutual exclusion in computer networks, *Communications of the ACM, vol. 24, no. 1,pp. 9-17,1981.*

[28] Baldoni, R., Virgillito, A., Petrassi, R.: A Distributed Mutual Exclusion Algorithm for Mobile Ad-Hoc Networks. *In: Proc. of ISCC (2002).*

[29] Benchaïba, M., Bouabdallah, A., Badache, N., Ahmed-Nacer, M.: Distributed Mutual Exclusion Algorithms in Mobile Ad Hoc Networks: *an Overview. ACM SIGOPS Operating Systems Review 38(1) (2004).*

[30] Walter, J., Kini, S.: Mutual Exclusion on Multihop, Mobile Wireless Networks, *Texas A&M Univ., College Station, TX 77843-3112, TR97-014 (December 9, 1997).*