A Software Architecture for Shared Resource Management in Mobile Ad hoc Networks

Orhan Dagdeviren and Kayhan Erciyes

Izmir Institute of Technology
Computer Eng. Dept., Urla, Izmir 35340, Turkey
{orhandagdeviren, kayhanerciyes }@iyte.edu.tr

Abstract. We propose a three layer software architecture for shared resource management in mobile ad hoc networks(MANETs). At the lowest layer, the Merging Clustering Algorithm(MCA)[11] partitions the MANET into a number of balanced clusters periodically. At the second layer, the Backbone Formation Algorithm(BFA) provides a virtual ring using the clusterheads found by MCA. Finally, an example resource management protocol which is a modified and scaled version of the Ricart-Agrawala algorithm implemented using the virtual ring structure is presented with the performance results.

1 Introduction

Mobile ad hoc networks do not have a fixed topology and the nodes of a MANET communicate using temporary connections with their neighbors. A MANET can be partitioned into a number of clusters to solve various problems such as routing and mutual exclusion in such networks. Mutual exclusion algorithms provide an efficient way of resource sharing in MANETS and also in distributed systems. Distributed mutual exclusion algorithms are either permission based or token based. A node would need permission from all of the related nodes to enter a critical section in a permission based algorithm. In token-based algorithms however, a node would require the possession of a system-wide unique token to enter a critical section. Susuki-Kasami's algorithm [8] (N messages) and Raymond's tree based algorithm [5] (log(N) messages) are examples of token based mutual exclusion algorithms. Examples of non-token based distributed mutual exclusion algorithms are Lamport's algorithm [3] (3(N-1) messages), Ricart-Agrawala (RA) algorithm (2(N-1) messages) [6] and Maekawa's algorithm [4]. *Safety*, *liveness* and *fairness* are the main requirements for any mutual exclusion algorithm. Lamport's algorithm and RA algorithm are considered as one of the only fair distributed mutual exclusion algorithms in literature. A distributed mutual exclusion algorithm using tokens is shown in [9] and a *k-way* mutual exclusion algorithm for ad hoc wireless networks where there may be at most k nodes executing a critical section at one time is described in [10].

In this study, we propose a three layer architecture for resource management in MANETs. At the lowest layer, a clustering algorithm provides dynamic clusters of the MANET, using the previously designed MCA [11]. The Backbone

Formation Algorithm at the second layer provides a virtual ring architecture of the coordinators of the clusters formed by MCA [11]. Finally, we show the implementation of the Distributed Mutual Exclusion Algorithm described in [1, 2] as the third layer application which uses the virtual ring structure. We first partition the MANET into a number of clusters periodically using the *Merging Clustering Algorithm (MCA)*. The nodes in the cluster that have direct connections, that is, in the communication ranges of the nodes of other clusters are called *neighbor nodes*. The MCA also provides the leader for every cluster which we will call *coordinator* here. Secondly, we construct a directed ring architecture across *coordinator*s. To achieve this goal, we propose the backbone formation algorithm. After formation of the ring, the coordinators perform the required critical section entry and exit procedures for the nodes they represent. Using this architecture, we improve and extend the RA algorithm described in [2] to MANETs and show that these algorithms may achieve an order of magnitude reduction in the number of messages required to execute a critical section at the expense of increased response times and synchronization delays which may also be useful in environments that use wireless sensor networks where energy efficiency, therefore message complexity is of paramount importance. The rest of the paper is organized as follows. Section 2 provides the background. Section 3 reviews the extended RA algorithm on the proposed model called *Mobile_RA*. The implementation results are explained in Section 4 and the discussions and the conclusions are outlined in Section 5.

2 Background

2.1 Clustering using Merging Clustering Algorithm

An undirected graph is defined as $G = (V, E)$, where V is a finite nonempty set and $E \subseteq V \times V$. The V is a set of nodes v and the E is a set of edges e. A graph $G_S = (V_S, E_S)$ is a spanning subgraph of $G = (V, E)$ if $V_S = V$. A spanning tree of a graph is an undirected connected acyclic spanning subgraph. Intuitively, a minimum spanning tree (MST) for a graph is a subgraph that has the minimum number of edges for maintaining connectivity [16]. Merging Clustering Algorithm MCA [11] finds clusters in a MANET by merging the clusters to form higher level clusters as mentioned in Gallagher, Humblet, Spira's algorithm [17]. However, we focus on the clustering operation by discarding minimum spanning tree. This reduces the message complexity as explained in [11] . The second contribution is to use upper$(2 * K)$ and lower(K) bound heuristics for clustering operation which results balanced number of nodes in the cluster formed. $Cluster\_leader$ is the node with the greatest $node\_id$ in a cluster. $Cluster\_id$ is equal to the $node\_id$ of the $cluster\_leader$.

2.2 Backbone Formation Algorithm

Backbone Formation Algorithm constructs a backbone architecture on a clustered MANET [12]. Different than other algorithms, the backbone is constructed

as a directed ring architecture to gain the advantage of this topology and to give better services for other middleware protocols [18–20, 2]. The second contribution is to connect the clusterheads of a balanced clustering scheme which completes two essential needs of clustering by having balanced clusters and minimized routing delay. Beside these, the backbone formation algorithm is fault tolerant as the third contribution. Our main idea is to maintain a directed ring architecture by constructing a minimum spanning tree between clusterheads and classifying clusterheads into *BACKBONE* or *LEAF* nodes, periodically. To maintain these structures, each clusterhead broadcasts a *Leader_Info* message by flooding. In this phase, clustermember nodes act as routers to transmit *Leader_Info* messages. Algorithm has two modes of operation; hop-based backbone formation scheme and position-based backbone formation scheme. In hop-based backbone formation scheme, minimum number of hops between clusterheads are taken into consideration in the minimum spanning tree construction. Minimum hop counts can be obtained during flooding scheme. For highly mobile scenarios, an agreement between clusterheads must be maintained to guarantee the consistent hop information. In position-based backbone formation scheme, positions of clusterheads are used to construct the minimum spanning tree. If each node knows its velocity and the direction of velocity, these information can be appended with a timestamp to the *Leader_Info* message to construct better minimum spanning tree. But in this mode, nodes must be equipped with a position tracker like a GPS receiver. Every node in the network performs the same local algorithm as shown in [12].

2.3 Performance Metrics

Performance of a distributed mutual exclusion algorithm depends on whether the system is *lightly* or *heavily* loaded. If no other process is in the critical section when a process makes a request to enter it, the system is lightly loaded. Otherwise, when there is a high demand for the critical section which results in queueing up of the requests, the system is said to be heavily loaded. The important metrics to evaluate the performance of a mutual exclusion algorithm are the Number of Messages per request (M), Response Time (R) and the Synchronization Delay (S). M can be specified for high load or light load in the system. The Response Time R is measured as the interval between the request of a node to enter critical section and the time it finishes executing the critical section. The synchronization delay S is the time required for a node to enter a critical section after another node finishes executing it. The minimum value of S is one message transfer time T since one message suffices to transfer the access rights to another node [7].

2.4 The Proposed Architecture

We propose a three layer architecture for MANETs as shown in Fig. 1. Implementations of other higher level functions on top of the lower two layers are possible. The lowest layer is where the clustering takes place at the end of which,

balanced clusters are formed. The second layer inputs these clusters and form a virtual ring of the coordinators of these clusters. Finally, the third layer shows the implementation of the Mobile_RA Algorithm on top of these two layers.

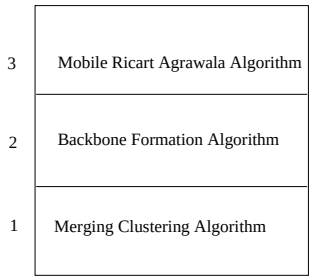

Fig. 1. Proposed Architecture

3 Mobile Ricart Agrawala Algorithm

For distributed mutual exclusion in MANETs, we proposed a hierarchical architecture where nodes form clusters and each cluster is represented by a *coordinator* in the ring [1]. The relation between the cluster coordinator and an ordinary node is similar to a central coordinator based mutual exclusion algorithm.

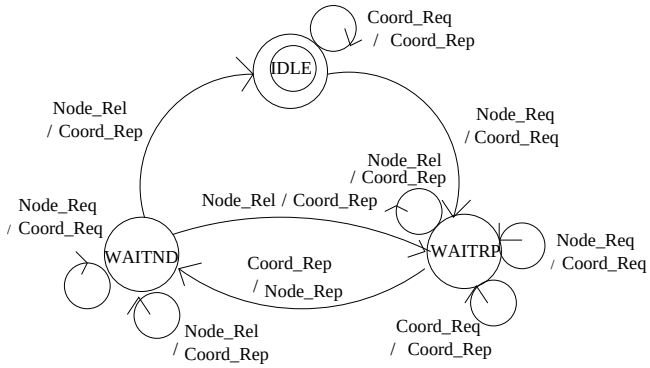

Fig. 2. FSM of the Mobile_RA Coordinator

The types of messages exchanged are *Request*, *Reply* and *Release* where a node first requests a critical section and upon the reply from the coordinator, it enters its critical section and then releases the critical section. The finite state machine representation of the Mobile_RA coordinator is shown in Fig. 2[1, 2].

The coordinator sends a critical section request (*Coord_Req*) to the ring for each node request (*Node_Req*) it receives. When it receives an external request (*Coord_Req*), it performs the operation of a normal RA node by checking the timestamps of the incoming request by the pending requests in its cluster and sends a reply (*Coord_Reply*) only if all of the pending requests have greater timestamps than the incoming request. When a node sends a *Node_Rel* message, the coordinator sends *Coord_Rel* messages to all of the requests in the *wait_queue* that have smaller timestamps than the local pending ones.

3.1 Illustration of the Mobile_RA Algorithm

Fig. 3 shows an example scenario for the Mobile_RA Algorithm in the network where the network of 20 nodes is partitioned into clusters 19, 14, 17 and 18 using MCA. K parameter is selected as 4. Nodes 19, 14, 17 and 18 are the cluster leaders and the cluster coordinators of clusters 1, 2, 3 and 4. They form a ring together with 0,3 and 10. Node 6, node 4, node 16 makes request for critical section respectively at 3.75s, 3.85s, 3,90s. Execution Time of critical section is taken as 350ms. The following describes the events that occur :

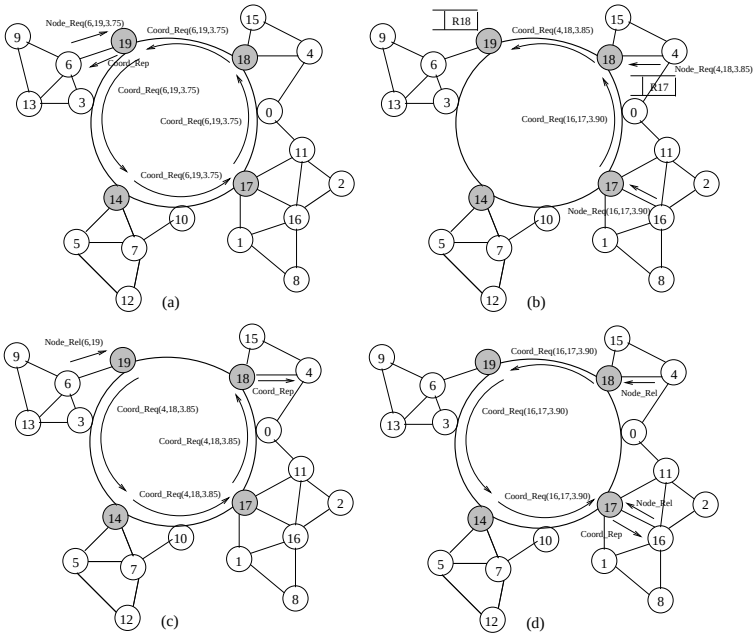

Fig. 3. Operation of the Mobile_RA Algorithm

1. Node 6 in cluster 19 makes a critical section request at 3.75s by sending *Node_Req (6,19,3.75)* message to node 19 which is the cluster coordinator.

Node 19 receives the message at 3.76s and changes its state to $WAIT\_RP$. Node 19 sends a *Coord_Req (6,19,3.75)* message to next coordinator(node 14) on the ring. Node 14, which is in $IDLE$ state and has no pending requests in its cluster, receives the *Coord_Req (6,19,3.75)* message at 3.78s and forwards the message to the next coordinator(node 17) on the ring. The message traverses the ring and received by node 19 which is in $WAIT\_RP$ state at 3,82s meaning all of the coordinators have confirmed that either they have no pending requests or their pending requests all have higher timestamps. Node 19 sends a *Coord_Rep* message to node 6 and changes its state to $WAIT\_ND$. Node 6 receives the *Coord_Rep* message at 3.83s and enters the critical section. Step 1 is depicted in Fig. 3.(a).

2. Node 4 in cluster 18 makes a critical section request by sending a *Node_Req_ (4,18,3.85)* at 3.85s. Node 18 receives the *Node_Req (4,18,3.85)* message at 3.86s and sends a *Coord_Req (4,18,3.85)* message to its next coordinator(node 19) on the ring. Node 19, which is in $WAIT\_ND$ state, receives the message and enqueues the *Coord_Req (4,18,3.85)* at 3.87s. Node 16 makes a critical section request at 3.90s. Node 18 which is in $WAIT\_RP$ state receives the *Coord_Req (16,17,3.90)* message and enqueues the message at 3.93s. Step 2 is depicted in Fig. 3.(b).
3. Node 6 exits from critical section at 4.18s and sends a *Node_Rel* message to node 19. Node 19 which is in $WAIT\_ND$ state receives the message at 4.19s and makes a transition to $IDLE$. Node 19 dequeues and forwards *Coord_Req (4,18,3.85)* message to next coordinator(node 14). The *Coord_Req (4,18,3.85)* message is forwarded by node 17 since its request has higher timestamp. Node 18 receives its original request at 4.25s and sends a *Coord_Rep* message to node 4. Node 4 enters the critical section at 4.26s. Step 3 is depicted in Fig. 3.(c).
4. Node 4 finishes to execute critical section at 4.61s. Node 18 receives the *Node_Rel* message at 4.62s. Node 18 dequeues and forwards the *Coord_Req (16,17,3.90)* message to its next coordinator(node 19) on the ring. Operation is continued as explained before. Node 17 receives *Node_Rel* message from node 16 at 5.03s. The Step 4 is depicted in Fig. 3.(d).

If there are multiple requests within the same cluster, time stamps are checked similarly for local request. The order of execution in this example is nodes $6 \rightarrow 4 \rightarrow 16$ in the order of the timestamps of the requests.

We briefly state the following properties of the Mobile_RA Algorithm which were described and proven in [2]

- The total number of messages per critical section using the Mobile_RA Algorithm is $k + 3d$ where k is an upper bound on the n umber of neighbor nodes in the ring including the cluster coordinators and d is an upperbound on the diameter of a cluster.
- The Synchronization Delay (S) in the Mobile_RA Algorithm varies from $2dT$ to $(k + 2d - 1)T$.

– In the Mobile_RA Algorithm, the response times are R_{light}=$(k+3d)T+E$ and R_{heavy} varies from $w(2dT+E)$ to $w((k+2d-1)T+E)$ where k is the number of clusters and w is the number of pending requests.

Table 1. Performance of Mobile_RA Algorithm

M_{light}	M_{heavy}	R_{light}	$R_{heavy-min}$	S_{min}	S_{max}
$k+3d$	$k+3d$	$(k+3)dT+E$	$w(2dT+E)$	$2dT$	$(k+2d-1)T$

Since the sending and receiving ends of the algorithm are the same as of RA algorithm, the safety, liveness and fairness attributes are the same. The performance metrics for the Mobile_RA Algorithm is summarized in Tab. 1.

4 Results

We implemented the protocol stack with the $ns2$ simulator. A random load generator is implemented to generate high, medium and low loads for different number of nodes. Different size of flat surfaces are chosen for each simulation to create small, medium and large distances between nodes. Very Small, Small and Medium surfaces vary between 310m × 310m to 400m × 400m, 410m × 410m to 500m × 500m, 515m × 515m to 650m × 650m respectively. Random movements are generated for each simulation. Low, medium and high mobility scenarios are generated and respective node speeds are limited between 1.0m/s to 5.0m/s, 5.0m/s to 10.0m/s, 10.0m/s to 20.0m/s. K parameter of merging clustering algorithm is changed to obtain different size of clusters. Response times and synchronization delays as measured with respect to load, mobility, distance and K are recorded. Execution of critical section is selected as 100ms.

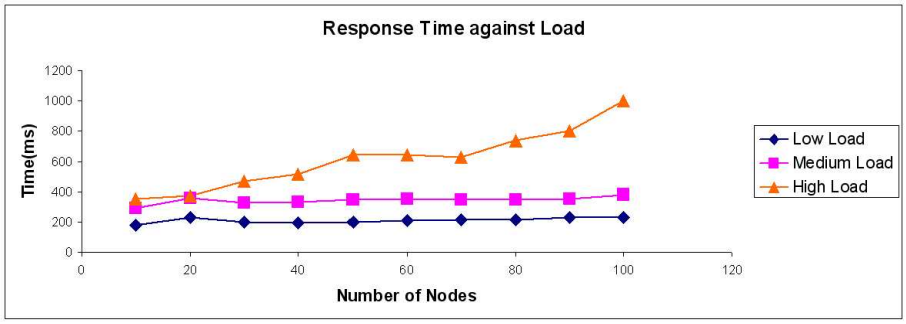

Fig. 4. Response Time against Load for Mobile_RA

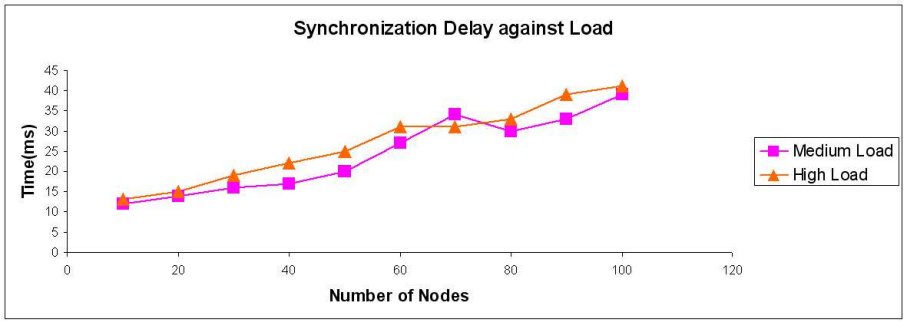

Fig. 5. Synchronization Delay against Load for Mobile_RA

Response time behaves as expected in low load scenarios as shown in Fig. 4. Synchronization delay values are smaller in medium load as shown in Fig. 5. The synchronization delay is 0 in low load scenarios since there will be no waiting requests in the queues. When the load is increased, response time increases due to the waiting times of requests in the queue. Also, the response time and the synchronization delay increase due to collisions and routing delays caused by high network traffic as shown in Fig. 4 and Fig. 5. Response time and synchronization delay values are scalable against against the mobility as shown in Fig. 6 and Fig. 7.

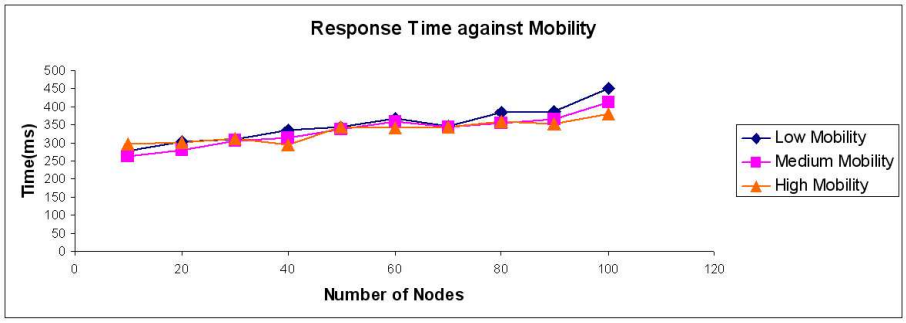

Fig. 6. Response Time against Mobility for Mobile_RA

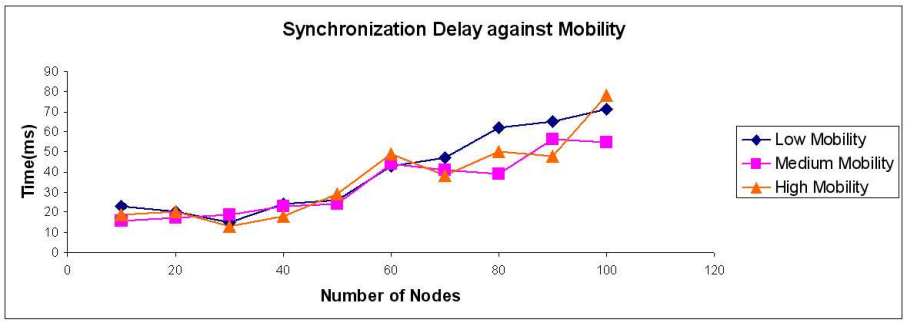

Fig. 7. Synchronization Delay against Mobility for Mobile_RA

Fig. 8 and Fig. 9 shows the effects of distance between nodes to response time and synchronization delay. As the distance between nodes increases the connectivity is decreased. This situation causes greater delays. K parameter is selected between 3 to 8 in a MANET with 60 node. In fixed number of nodes, as the cluster size increases, total number of clusters in the network decreases. This also reduces the number of cluster leaders forming the ring and routing delay which causes decrease in the response time and the synchronization delay as shown in Fig. 10 and Fig. 11.

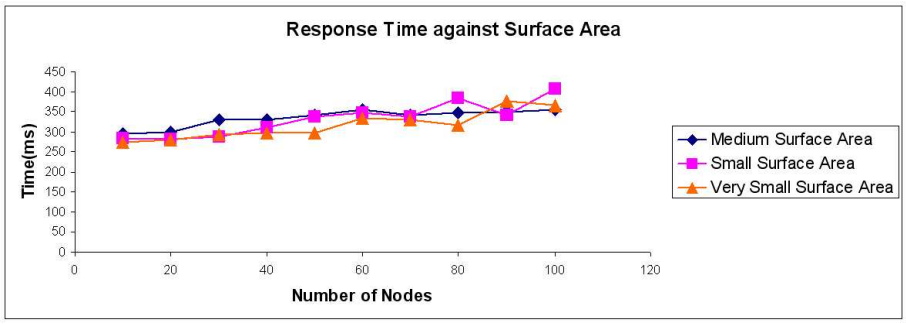

Fig. 8. Response Time against Surface Area for Mobile_RA

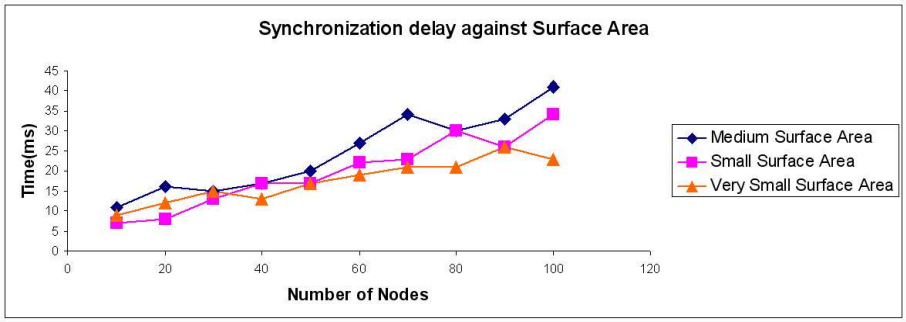

Fig. 9. Synchronization Delay against Surface Area for Mobile_RA

Consequently, our results conform with the analysis that response time against low and medium loads increases linearly with a small gradient. Synchronization delay values against medium and high load also increase linearly. Response time against high load makes a sharp increase due to high network traffic. Response time and synchronization delay values are stable under different mobility and surface area conditions. Response time and synchronization delay value decrease linearly against the number of clusters in the MANET.

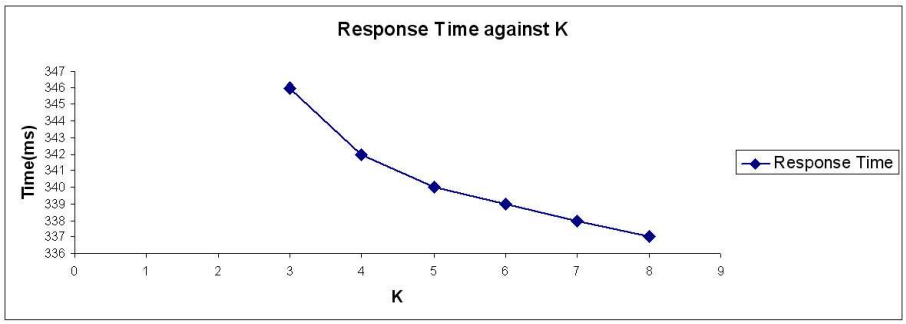

Fig. 10. Response Time against K for Mobile_RA

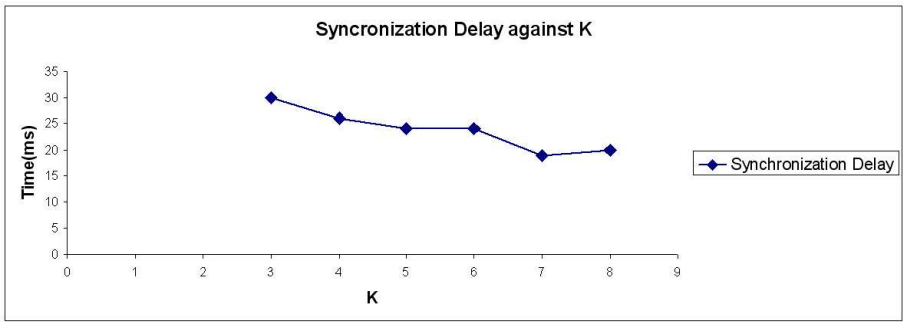

Fig. 11. Synchronization Delay against K for Mobile_RA

5 Conclusions

We proposed a three layer architecture for resource management in a MANET and the implementation results of the Mobile_RA Algorithm for MANETs. The MANET is partitioned into clusters at regular intervals by the MCA which also provides connected clusterheads. Ring architecture is constructed by Backbone Formation Algorithm. The Mobile_RA Algorithm, together with the architecture that it is executed on, provides improvement over message complexities of Ricart and Agrawala and other distributed mutual exclusion algorithms. A comparison of the two algorithms with their regular counterparts in terms of their message complexities is shown in Tab. 2. If we assume k=m=d for simplicity, the message complexities of the mobile algorithms are in the order of $sqrt(N)$ where N is the total number of nodes in the network [2]. From the test results, we observe that response time R is scalable with respect to the number of mobile nodes for all load states in the MANET as high, medium or low loads. R is also scalable with respect to node mobility and the distance between the mobile nodes. The coordinators have an important role and they may fail. New coordinators may be elected and also any failed node member can be excluded from the clusters using Backbone Formation Algorithm. Our work is ongoing and we are looking into implementing this algorithm in wireless sensor network architectures where preserving energy is important, hence low message complexities are required. We are also considering k-way distributed mutual exclusion algorithms in MANETs.

Table 2. Comparison of the Mobile Mutual Exclusion Algorithms with others

	Regular	Mobile Algs.	Mobile (k=m=d)
Ricart-Agrawala Alg.	$2(N-1)$	$k+3d$	$\Theta(4sqrt(N))$
Token Passing Alg.	N	$O(k+3d)$	$O(4sqrt(N))$

References

1. Erciyes, K. : Distributed Mutual Exclusion Algorithms on a Ring of Clusters, ICCSA 2004, Springer-Verlag, LNCS 3045, (2004), 518-527.
2. Erciyes, K., Cluster-based Distributed Mutual Exclusion Algorithms for Mobile Networks , EUROPAR 2004, Springer-Verlag, LNCS 3149, (2004), 933-940.
3. Lamport, L. : Time, Clocks and the Ordering of Events in a Distributed System, CACM, 21, (1978), 558-565.
4. Maekawa, M. : A sqrt(n) Algorithm for Mutual exclusion in Decentralized Systems, ACM Transactions on Computer Systems, 3(2), (1985), 145-159.
5. Raymond, K. : A Tree-based Algorithm for Distributed Mutual Exclusion. ACM Trans. Comput. Systems, 7(1), (1989), 61-77.
6. Ricart, G., Agrawala, A. : An Optimal Algorithm for Mutual Exclusion in Computer Networks, CACM, Vol. 24(1), (1981), 9-17.
7. Shu, Wu : An Efficient Distributed Token-based Mutual Exclusion Algorithm with a Central Coordinator, Journal of Parallel and Distributed Processing, 62(10), (2002), 1602-1613.
8. Susuki, I., Kasami, T. : A Distributed Mutual Exclusion Algorithm, ACM Trans. Computer Systems, Vol. 3(4), (1985), 344-349.
9. Walter,J., E., Welch, J., L., Vaidya, N., H. : A Mutual Exclusion Algorithm for Ad Hoc Mobile Networks, Wireless Networks, Vol. 7(6), (2001), 585-600.
10. Walter, J., E., Cao, G., Mohanty, M. : A K-way Mutual Exclusion Algorithm for Ad Hoc Wireless Networks, Proc. of the First Annual workshop on Principles of Mobile Computing, (2001).
11. Dagdeviren, O., Erciyes, K., Cokuslu, D., : Merging Clustering Algorithms, LNCS 3981, 681-690, ICCSA, (2006).
12. Dagdeviren, O., Erciyes, K. : A Distributed Backbone Formation Algorithm for Mobile Ad hoc Networks , to be published in the Proc. of ISPA06, (2006).
13. West, D. : Introduction to Graph Theory , Second edition, Prentice Hall, Upper Saddle River, N.J., (2001).
14. Chen, Y. P., Liestman, A. L. : Approximating Minimum Size Weakly-Connected Dominating Sets for Clustering Mobile Ad Hoc Networks , Proc. 3rd ACM Int. Symp. Mobile Ad Hoc Net. and Comp., 165-72, (2002).
15. Haynes, T. W., Hedetniemi, S. T., Slater, P. J. : Domination in graphs, Advanced Topics , Marcel Dekker Inc., (1998).
16. Grimaldi, R. P. : Discrete and Combinatorial Mathematics, An Applied Introduction, Addison Wesley Longman, Inc., (1999).
17. Gallagher, R. G., Humblet, P. A., AND Spira, P. M. : A Distributed Algorithm for Minimum-Weight Spanning Trees, ACM Transactions on Programming Languages and Systems 5, (1983), 66-77.
18. Baldoni, R. Virgillito, A., Petrassi, R. : A distributed mutual exclusion algorithm for mobile ad-hoc networks, Computers and Communications, (2002), 539-544.
19. Delmastro, F. : From Pastry to CrossROAD: CROSS-layer ring overlay for ad hoc networks, Third IEEE International Conference on Pervasive Computing and Communications Workshops, (2005), 60-64.
20. Yang, C. Z. : A token-based h-out of-k distributed mutual exclusion algorithm for mobile ad hoc networks, 3rd International Conference on Information Technology, (2005),73-77.