A Token based Distributed Group Mutual Exclusion Algorithm with Quorums for MANET

Pratvina Talele, Milind Penurkar, Saranga Bhutada, Harsha Talele

Abstract*—The group mutual exclusion problem extends the traditional mutual exclusion problem by associating a type (or a group) with each critical section. In this problem, processes requesting critical sections of the same type can execute their critical sections concurrently. However, processes requesting critical sections of different types must execute their critical sections in a mutually exclusive manner. A distributed algorithm is used for the group mutual exclusion problem in asynchronous message passing distributed systems for MANET. This algorithm is based on tokens, and a process that obtains a token can enter a critical section. To reduce message complexity, it uses a coterie as a communication structure, when a process sends a request messages. Informally, a coterie is a set of quorums, each of which is a subset of the process set, and any two quorums share at least one process. Performance of the proposed algorithm is presented. In particular, the proposed algorithm can achieve high concurrency, whch is a performance measure for the number of processes that can be in a critical section simultaneously.

Index Terms*—Distributed systems, critical section, mutual exclusion.

I. INTRODUCTION

Distributed mutual exclusion is one of the most fundamental problems in distributed systems. In this problem, access to a shared resource (that is, execution of critical section) by different processes must be synchronized to ensure its integrity by allowing at most one process to access the resource at a time. Because it is a fundamental problem in distributed computing, many generalized problems have been proposed in the literature. The problem of distributed group mutual exclusion is considered, which is a generalization of the distributed mutual exclusion problem such that only processes in the same group can enter a critical section (CS) simultaneously. In other words, no two processes in different groups enter a CS at a time.

Mobile ad hoc networks (MANETs) do not have fixed infrastructure and consist of mobile wireless nodes that have temporary interconnections to communicate over packet radios. The facility that ensures that only one node is in its critical section (CS) at any given time, namely mutual exclusion in a distributed system such as a MANET has received attention from various researchers recently. In general, distributed mutual exclusion algorithms may be classified as permission based or token based.

Manuscript received on March, 2013.

Ms. Pratvina Talele, IT Department, MIT College of Engineering, Pune University, Pune, India.

Mr. Milind Penurkar, IT Department, MIT College of Engineering, Pune University, Pune, India.

Mrs. Saranga Bhutada, IT Department, MIT College of Engineering, Pune University, Pune, India.

Ms. Harsha Talele, IT Department, SSBT's College of Engineering, North Maharashtra University, Jalgaon, India.

In the permission-based algorithms, a node would enter a CS after receiving permission from all of the nodes in its set for the CS. In token-based algorithms however, the possession of a system-wide unique token would provide the right to enter a CS.Concurrency and waiting time are important performance measures for a distributed group mutual exclusion algorithm. Concurrency means the number of processes that are executing their respective code simultaneously, and higher concurrency is always better for system throughput. Waiting time is the time that a process must wait to enter the CS after it issues a request. The design of an algorithm that achieves high concurrency and small waiting time is nontrivial when processes in different groups make requests simultaneously. Consider the situation that some processes are in the CS and a process in the same group makes a request. If the request is granted immediately, concurrency increases. On the other hand, requests by other processes in different groups must wait to be granted permission to CS. The difficulty of algorithm design is the trade-off between concurrency and waiting time.

Many distributed group mutual exclusion algorithms have been proposed in the literature, and they are categorized as follows: Decentralized permission type algorithm:- When a process wants to enter a CS, it sends request messages to some processes, and it enters the CS if it obtains permissions from some set of processes defined in advance. Typically, a quorum or its variant is used to define a set of processes from which a process must obtain permission.

Privileged token type algorithm: - A virtual object, called a token, is maintained by processes, and only a process that obtains a token may enter the CS. In the algorithm, a process sends request messages to all processes so that a request arrives at a process that holds a token.

Hybrid type algorithm:- The first process to enter the CS obtains permission from some set of processes (decentralized permission type), and the process grants other processes in the same group to enter the CS by sending a token (privileged token type). Here, we have proposed a new distributed algorithm TQGmx that is of privileged token type for MANET. This algorithm uses coteries as a communication structure to reduce message complexity.

II. RELATED WORK

If In the literature, there are two popular approaches i.e. token-based and permission-based approach [2][3][4]. The DME algorithms proposed for MANETs are extensions of these classic algorithms to suit the requirements of MANETs. In the token-based algorithms [3], there exists a unique token in the system and only the node holding the token may access the critical section. In permission-based algorithms, the nodes wishing to enter the CS must explicitly gather permission from all other sites that are participating in the CS before it can enter the CS.

A. A Distributed Mutual Exclusion Algorithm for Mobile ad hoc network [2]

The proposed architecture has four layers for distributed mutual exclusion in mobile ad hoc networks. Distributed applications can be implemented on top of these layers. The lowest layer is the routing layer that can be one of the ad hoc network routing protocols like AODV, DSR or DSDV.

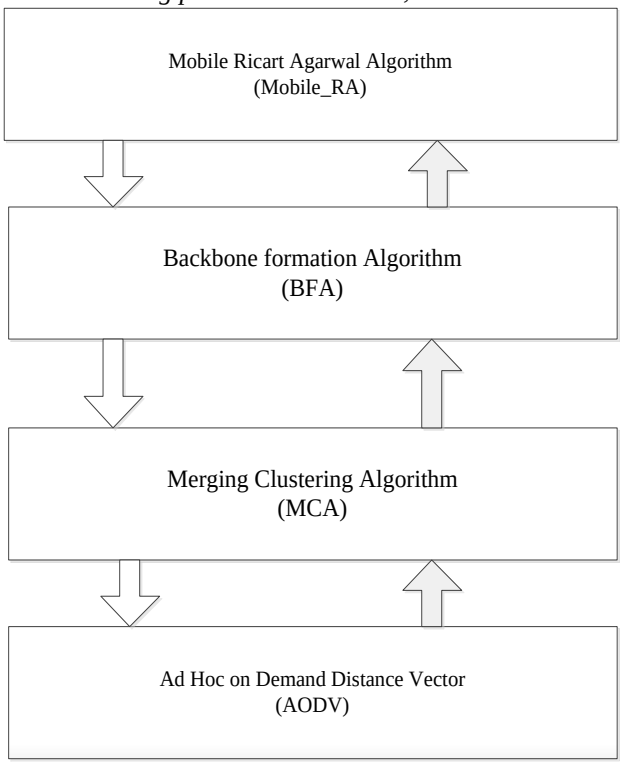

Figure 1 The Mobile_RA Architecture

In [2], authors used AODV since it is a widely used routing protocol which also has a stable network simulator-2 (ns-2) release. The second layer (the clustering layer) uses MCA. In the third layer or backbone formation layer, BFA is used that inputs these clusters and builds a directed ring of the cluster heads. BFA can be implemented on top of any clustering algorithm that produces one cluster head for each cluster. MCA and BFA handle the link failures caused by the mobility of the nodes and also produce stable topologies against varying node densities. Finally the last layer is the Mobile_RA for distributed mutual exclusion. Any distributed algorithm running on top of ring architecture can be implemented as the last layer.

The main idea of the Mobile_RA algorithm is to form coordinators as interface of other nodes to the ring. The relation between the cluster coordinator and an ordinary node is similar to a central coordinator based mutual exclusion algorithm. Each node is in idle state initially. When a node wants to enter CS, an internal request event occurs, upon this event the node sends a request to its coordinator and makes a state transition to wait state. After the node receives reply message from the coordinator, it executes CS. When the node finishes the execution of CS, node sends a release message to its coordinator.

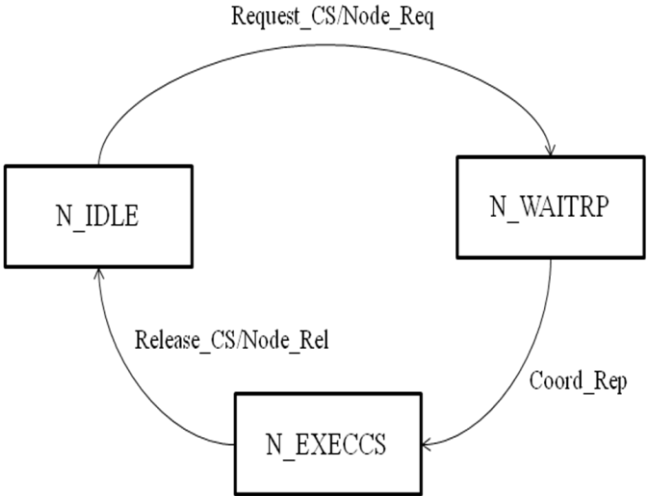

Figure 2 Finite State Machine of Mobile_RA Node Algorithm

The communication infrastructure to run the algorithm consists of a number of clusters of mobile nodes where each cluster is represented by a coordinator and the coordinators are connected to form a ring. The cluster formation and ring formation can be handled efficiently by the Merging Clustering Algorithm and the Backbone Formation Algorithm. Due to this hierarchical structure, significant gains in total message complexities are obtained with respect to the original Ricart-Agrawala algorithm.

B. A Novel Permission-Based Reliable Distributed Mutual Exclusion Algorithm For MANETS [4]

This part describes how this algorithm behaves when it enters CS, requests for CS, receives a REQUEST message, receives a HOLD message, and the adaptive timeout mechanism.

In Figure 3, a simple operation of the algorithm is illustrated. The site S0 is initially interested in entering into CS, and it sends REQUEST message (in solid arrow) to all other sites in its Info_Set, S1 and S2. Since both the sites are not currently interested in entering into CS, they immediately respond with REPLY message (in dashed arrow). Since S0 gets REPLY from all the requested sites, it enters CS. While it is in CS, S2 is interested in entering CS, and sends REQUEST message to S0 and S1. Since S1 is still not in CS, it simply sends back a response. S0 is already in CS, so it sends back a HOLD message. As soon as S0 exits CS, it sends back REPLY message to S2. S2 can now enter into CS, as it has received REPLY message from all sites.

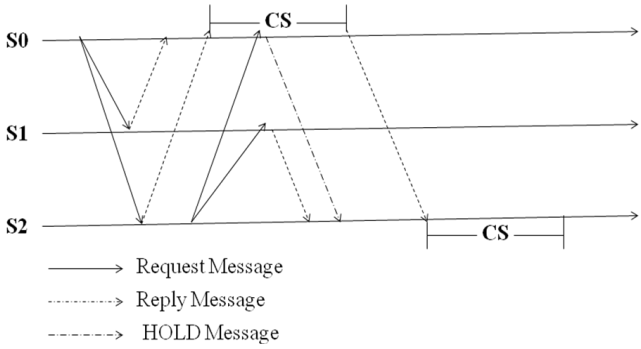

Figure 3 Illustration of an algorithm

Requesting for CS: When a site Si wants to enter CS, it will send a REQUEST message to all the nodes in the Info_set$_i$. Unlike waiting for a REPLY message that may come immediately or with a significant delay (if the other site is in CS), in this algorithm the requesting site is either sent with a REPLY message or a HOLD message in response to its REQUEST. For every site S_j to which Si has sent the REQUEST message, it will set $TO_{REQ[j]}$ as the expected round-trip time between S_i and S_j.

Receiving REQUEST message: When a site S_i receives a REQUEST message from another site S_j, S_i first checks whether it itself is in CS. If S_i is not in CS or it is not contending for CS, then it immediately sends a REPLY message back to S_j. If S_i is contending for CS, and it determines that S_j's request has a higher priority than its own request, then it sends back a REPLY message. On the other hand, if S_i is having a higher priority than S_j or S_i itself is in CS, it sends a HOLD message with the expected time for completing CS and the timestamp of its REQUEST message tsreq back to S_j. The expected time includes the propagation delay for sending the message back to S_j and the value of T_{CS_DONE}. S_i also adds the site S_j to Q_{HOLD}.

Receiving HOLD Message: When a site S_i receives a HOLD message from site S_j, it understands that either S_j is in CS or it is having a lower precedence than S_j for entering CS. So it will update its HOLD message timeout TO_{HOLD}, as the maximum of the timeout value, τ, from the message and the current value of TO_{HOLD}.

Since this algorithm uses an additional message, the performance of this algorithm is more than 2(Φ-1). The message complexity will depend on the number of times the HOLD message has to be sent. If the node is in CS miscalculates the timeout m times, and there are w nodes in QHOLD, then the message complexity of the algorithm will tend to be 2(Φ-1) + m*w. The ability to recover from any node failure offsets the additional message overhead incurred.

C. A Distributed Mutual Exclusion Algorithm for Mobile Ad-Hoc Networks [3]

A general DMUTEX algorithm presents features that allows it to be effectively used over a MANET. The algorithm aims at maintaining device power consumption as low as possible by reducing the number of hops traversed per CS execution and by not sending any control message when no processes request the CS. Mobility is addressed by exploiting the information of the routing table in order to send each message to the closest node in terms of number of hops.

This DMUTEX algorithm can be classified as "token based": Token-based algorithms allow the process holding the token to enter its CS. These algorithms can be further split into two families: "Token- Asking" and "logical ring".

Token-asking: A requesting process broadcasts a request for the token to all the processes. The process owning the token inserts the request in a request queue and when it leaves the CS, it sends the token to the first process in the queue, together with the request queue. When the request queue is empty, the process stops and waits for a request. In such a case we say that the system enters in an idle state. In other words idle states mean that no requests and no messages get exchanged by the DMUTEX algorithm.

Logical ring: The token circulates perpetually around a logical ring using point-to-point messages. A process receives the token from its predecessor in the ring, accesses the CS, if needed, and then sends the token to its successor on the ring. In the best scenario, CS access can be achieved with very low overhead in terms of messages exchanged when all processes issue a request the token in the same round, i.e. one message for CS entry.

This algorithm combines the best from the two families of token based algorithms (i.e., token-asking and circulating token) gets under heavy request load a number of hops traversed per CS very close to an optimal value. Moreover it is based on a dynamic logical ring, built on-the-fly by using a policy P at each process that selects as the next process in the ring the closest one in terms of numbers of hops. This paper shows that in a mobile ad-hoc networks P reduces as many as possible the number of hops experienced per CS execution in an algorithm round.

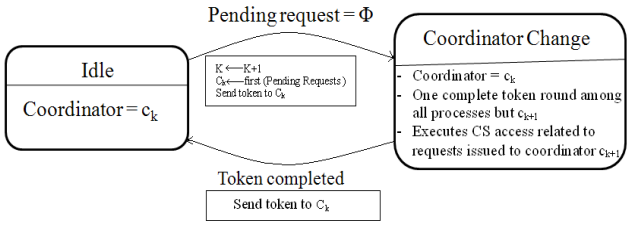

Figure 4 Algorithm state diagram

The algorithm, from an external viewpoint, continuously executes transitions between two states: Idle and Coordinator-Change. The algorithm evolves in a sequence of rounds during which the two states alternate. A new round starts when the state switches to a Coordinator-Change state. For each round of the algorithm, one process is declared the coordinator. Each time coordinator changes from ck-1 to ck, there is a transition between Idle and Coordinator-Change state and the process pi that provokes the transition becomes the current coordinator ck. The latter is the only process enabled to execute the next transition from Coordinator-Change to Idle state. The state diagram is described in Figure 4.

III. PROPOSED ALGORITHM

A. Algorithm

The proposed algorithm is based on the algorithm introduced in paper [1]. This algorithm uses two classes of tokens:- main token and sub-token. The main token is maintained as unique in the system. A sub-token is generated by the holder (process) of the main token and the number of sub-tokens varies. This algorithm assumes that network is reliable.

Steps involved:

When a node P_i wishes to enter the critical section (CS), it sends a request message to each node (P_j) in a quorum and it waits for a token to arrive.

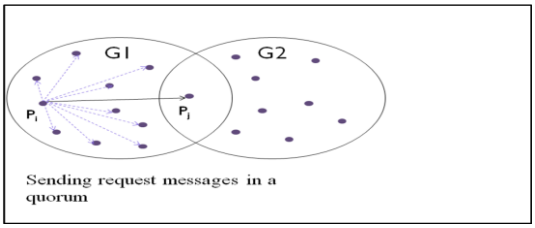

Figure 8 Requesting for Token

When a node P_j receives a request message from P_i, it forwards the request to the holder P_k of the main token if P_j knows such P_k. Otherwise the request is buffered until P_j knows such P_k.

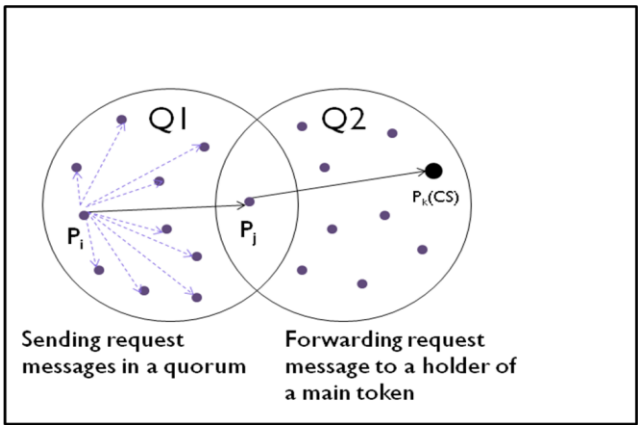

Figure 9 Forwarding request message

When the holder P_k of the main token receives a request issued by P_i and forwarded by P_j, the arrived request is in any case, put up into the queue of the main token. Each request in the queue is granted according to the following rules:

If no node is executing the CS, the main token is transferred by a token message to a requesting node.

If some processes are in the CS already and the requested group is the same as the current group, P_k sends a sub-token message to P_i, as long as there is no request in another group with a higher priority.

Otherwise, the request is kept in the queue of the main token. The main token may be transferred to another process to grant the request of the process, and this may happen several times. The request of P_i is eventually granted by a process that holds the main token, say, P_l, which may not be the same as P_k.

P_1	P_2	P_l	P_i	

Figure 10. Queue of main-token

Process P_i enters the CS by receiving the main token (a token message) or a sub-token (a sub-token message).

When process P_i exits the CS, if P_i is the holder of the main token it decrements the group size by one. Otherwise, i.e., P_i holds a sub-token, it returns a sub-token by sending a release message.

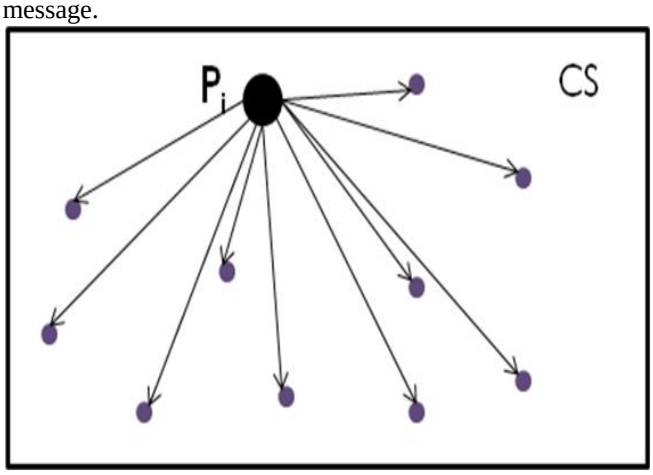

Figure 11. Releasing the CS

B. Fault Tolerance

If node failed when it was in the CS, the current status of the CS is not saved.

If node (holding the main token) fails before passing main token to requesting node, the main token will be issued to the first requesting node having the temporary queue that is maintained at each node in the quorum.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

A. Performance Metrics

Performance of a distributed mutual exclusion algorithm depends on whether the system is lightly or heavily loaded. the system is lightly loaded If no other process is in the CS, when a process makes a request to enter it. Otherwise, when there is a high demand for the CS that results in queuing 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, response time and the synchronization delay as described below:

• Number of Messages per Request: The total number of messages required to enter CS is an important and useful parameter to determine the required network bandwidth for that particular algorithm.

• Response Time (R): The Response Time R is measured as the interval between the request of a node to enter a CS and the time it finishes executing the CS. When the system is lightly loaded, twice the number of message transfer times and the execution time of the CS success results in Rlight = 2T +E units. Under heavy load conditions, assuming at least one message is needed to transfer the access right from one node to another, then Rheavy = w (T + E) where w is the number of waiting requests.

• Synchronization Delay (S): The synchronization delay S is the time required for a node to enter a CS once another node finishes executing it. The minimum value of S is one message transfer time T, since it requires one message success to transfer the access rights to another node.

B. Performance Evaluation

The implementation of a token based distributed group mutual exclusion algorithm with quorum for MANET is done using the Network Simulator 2 (ns-2) simulator. IEEE 802.11 standards are used for medium access control and physical layer, where the transmission range of a mobile node is 30m. Total number of nodes is selected from 20 to 100 nodes.

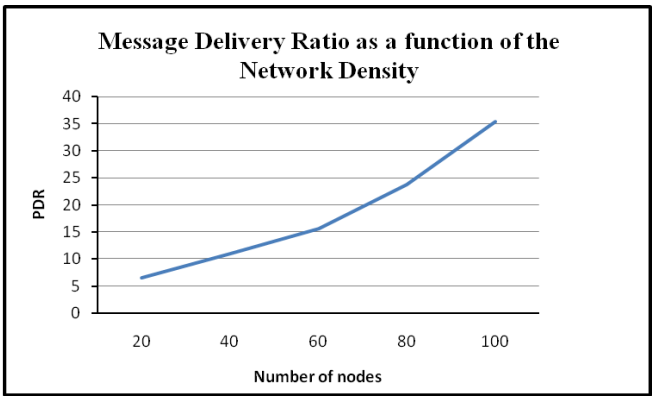

Figure 12 Average Message Delivery Ratio against Density

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

As the number of nodes increases in the network, the packet delivery ratio increases from 6 to 35 as seen in Figure 12.

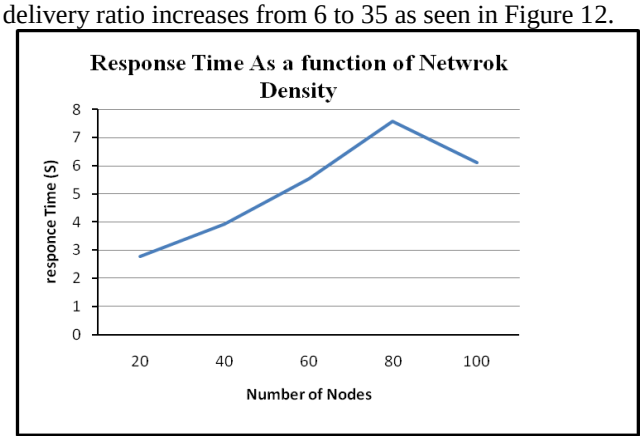

Figure 13 Average Response Time against Density

As the number of nodes increases in the network, the response time increases as shown in Figure 13. It reaches the optimum value and then starts decreasing. This indicates that for lower response time, number of nodes should be greater than 80. The average response time vary from 2.79s to 7.57s.

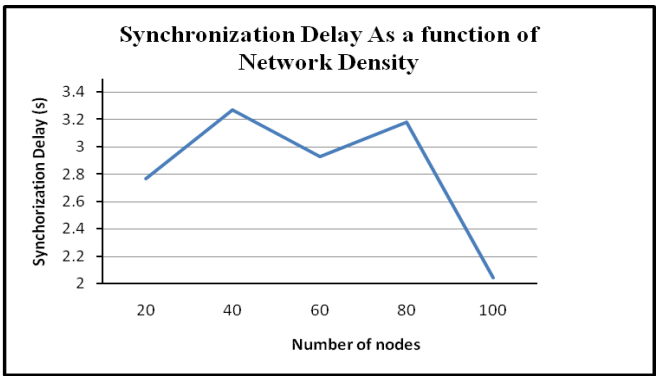

Figure 14 Average Synchronization Delay against Density

As the number of nodes increases in the network, the synchronization delay varies from 2.04s to 3.27s.
Response times and synchronization delays as measured with respect to load, mobility are recorded. Response time decreases linearly with the load as seen in Figure 15. In low loaded networks, average response times is 4,92s whereas in high loaded networks average response times is 0.61s as seen in Figure 15. The synchronization delay is 2.51s in low load scenarios. When the load is increased, synchronization delay linearly decreases due to no waiting time of requests in the queue as seen in Figure 16.

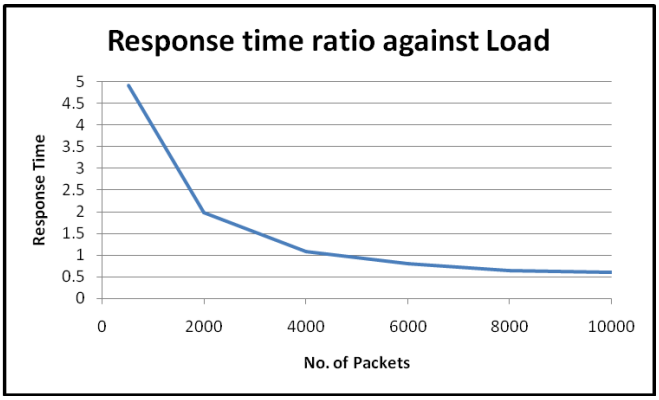

Figure 15 Average Response Time against Load

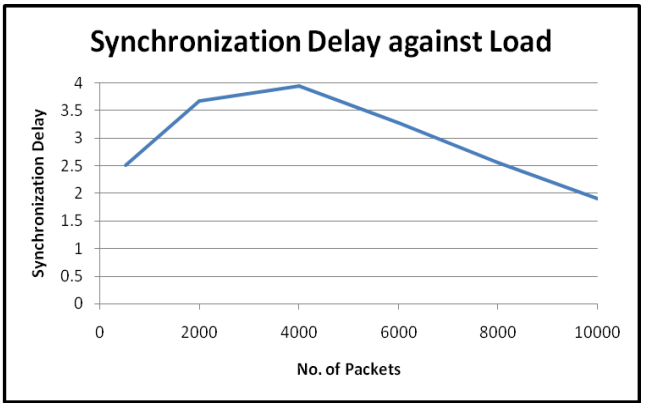

Figure 16 Average Synchronization Delay against Load

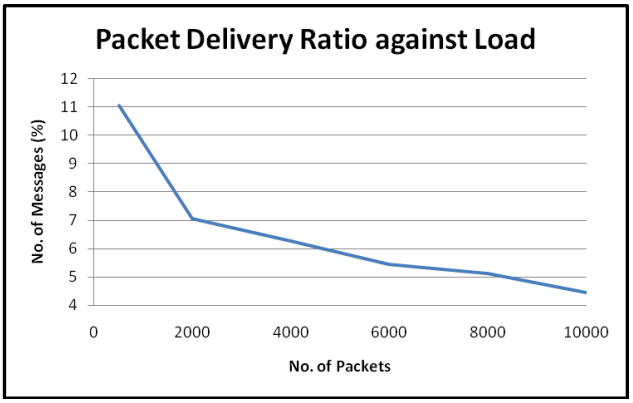

Figure 17 Average Packet Delivery Ratio against Load

Random movements are generated for each simulation and random waypoint model is chosen as the mobility pattern. Low, medium and high mobility scenarios are generated and the mobility speed is varied by 5 m/s, 10 m/s, 15 m/s, 20 m/s, 25 m/s.

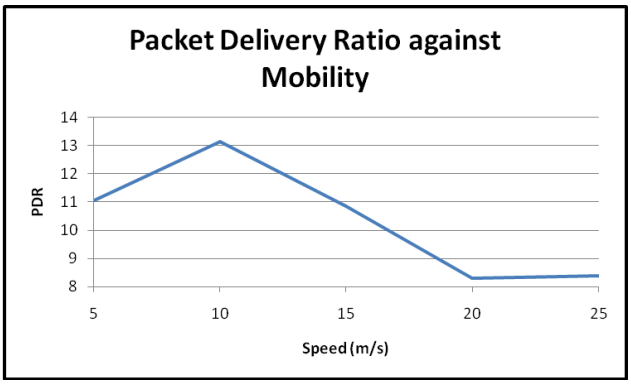

Figure 18 Average Packet Delivery Ratio against Mobility

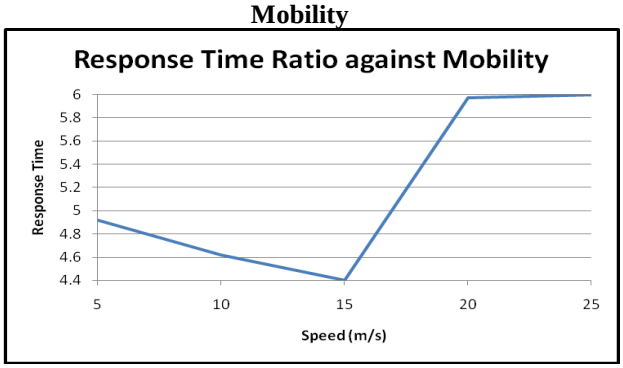

Figure 19 Average Response Time against Mobility

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

As the mobility increases the packet delivery ratio decreases and response time, synchronization delay increases due to rapid change of network topology.

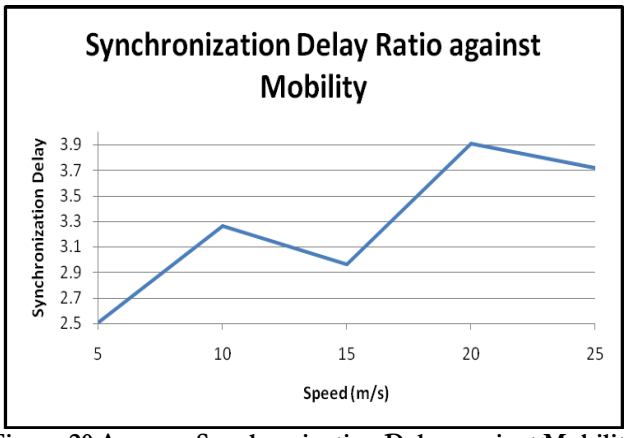

Figure 20 Average Synchronization Delay against Mobility

Figure 21 and Figure 22 shows the comparison of three algorithms in terms of response time and synchronization delay. The algorithms considered for comparison are token based group mutual exclusion algorithm, Ricart-Agrawala algorithm for MANET and the Ricart-Agrawala algorithm for wired network.

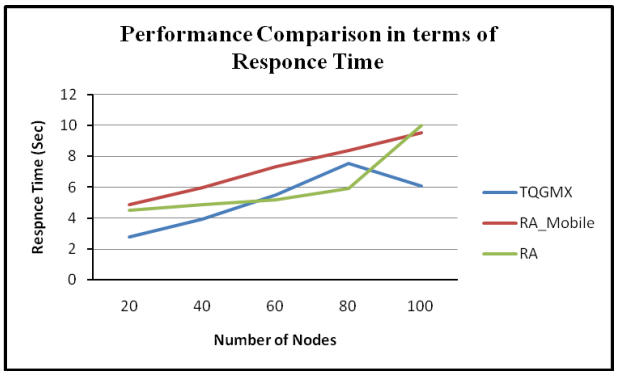

Figure 21 Performance Comparison of Response Time

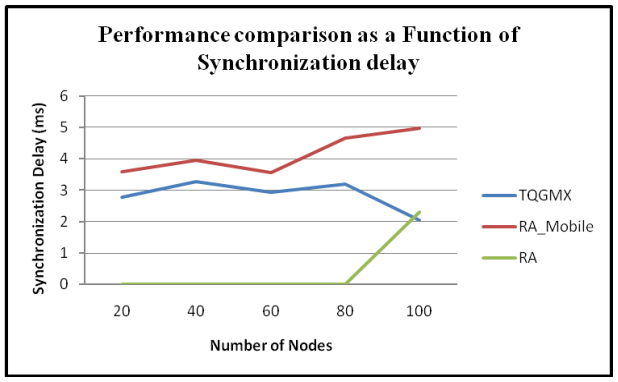

Figure 22 Performance Comparison of Synchronization Delay

The average response time and synchronization delay values can be seen in Figure 21 and Figure 22. The TQGMX is far more scalable than RA for MANETs as seen in Figure 21 and Figure 22. But the average response time and average synchronization delay against RA_Mobile decreases linearly.

Consequently, our results conform to the analysis that response time values against load decrease linearly with a small gradient. Synchronization delay values against load also decrease linearly. Synchronization delay values are not stable under different mobility.

V. CONCLUSION

There are many distributed mutual exclusion algorithms proposed for mobile ad-hoc networks. But most of them have not been implemented till now. The proposed algorithm is token based algorithm. This proposed algorithm achieves high concurrency which is an important factor for group mutual exclusion. Also it achieves low response time and synchronization delay as compared to Mobile_RA algorithm, permission based Distributed Mutual Exclusion algorithm and Distributed Mutual Exclusion algorithm.

REFERENCES

1. Hirotsugu Kakugawa, Sayaka Kamei, Toshimitsu Masuzawa, "A Token-Based Distributed Group Mutual Exclusion Algorithm with Quorums", *Published by the IEEE Computer Society* 1045-9219/08/$25.00 2008 IEEE.
2. Kayan Erciyes, Orhan Dagdeviren, "A Distributed Mutual Exclusion Algorithm for Mobile Ad Hoc Networks", *International Journal of Computer Networks & communications Vol.4, N0.2*, March 2012.
3. Roberto BALDONI, Antonino VIRGILLITO, Roberto PETRASSI, "A Distributed Mutual Exclusion Algorithm for Mobile Ad-Hoc Networks", *Dipartimento di Informatica e Sistemistica Universita di Roma "La Sapienza"* Via Salaria 113, 00198 Roma, Italia 2001.
4. Murali Parameswaran, Chittaranjan Hota, "A Novel Permission-based Reliable Distributed Mutual Exclusion Algorithm for MANETs", 978-1-4244-7202-4/10/$26.00 ©2010 IEEE.
5. M. Benchaiba, A. Bouabdallah, N. Badache, M. Ahmed-Nacer, "Distributed Mutual Exclusion Algorithm In Mobile Ad-Hoc Networks : An overview", *Published in 2003.*
6. Y. –J. Joung, "The Congenial Talking Philosophers Problem in Computer Networks", *Distributed Computing, vol. 15,* pp. 155-175, 2002.
7. Fransico J. Ros, Pedro M. Ruiz, "Implementing a New Manet Unicast Routing Protocol in NS2", *University of Mercia*, December 2004.

AUTHORS PROFILE

Ms. Pratvina Talele received the Bachelors of Engineering in Information Technology from North Maharashtra University in 2003. She is currently an Assistant Professor in MIT College of Engineering, Pune University. Her research interests are Distributed Systems and Data Structures.

Mr. Milind Penurkar received the BE, MTech. in CSE from Pune University in 2002 and 2008 respectively. He is pursuing his Ph.D. in CS from Nagpur University. He is currently an Associate Professor in MIT College of Engineering, Pune University. His research interests are Distributed Systems and Wireless Network.

Mrs. Saranga Bhutada received the Bachelors of Engineering in Information Technology from North Maharashtra University in 2003. She is currently an Assistant Professor in MIT College of Engineering, Pune University. Her research interests are Distributed Systems and Wireless Network.

Ms. Harsha Talele received the Bachelors of Computer Engineering from Pune University in 2009. She is currently pursuing master of engineering in SSBT.s College of Engineering, North Maharashtra University. Her research interests are Distributed Systems and Advanced Database.