Cluster based distributed mutual exclusion algorithm for mobile Ad hoc Network

Prof .Dudhagi Rupali R., Prof. Valsang Aarti B., Prof. Khureshi M. A.
Department of Computer Science and Engg.
A.G Patil institute of technology, Solapur, India

***Abstract*- A mobile ad hoc network is a network wherein a pair of nodes communicates by sending messages either over a direct wireless link, or over a sequence of wireless links including one or more intermediate nodes. In this paper, the proposed algorithm is suitable for large distributed application. It is combination of permission based and token based broadcast algorithm and a network having N number of nodes is splitted into M groups termed as clusters. This algorithm requires nodes to communicate with only their current neighbors in the cluster, making it well-suited to the ad hoc environment. 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. With this algorithm synchronization delay and starvation problem is reduced to minimal.**

***Index Terms*- Mutual exclusion, distributed system and ad hoc networks**

I. INTRODUCTION

A mobile ad-hoc network (**MANET** is a self-configuring infrastructure less 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. 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.

According to the algorithmic principle, the mutual exclusion algorithms for distributed systems can broadly be classified into two categories. *Permission based algorithms and. token based algorithms*.

In permission based algorithm, a process can enter in the critical section only after receiving permission from other process (es) in the system. In token based algorithm, a unique token is shared among all nodes. A node is allowed to enter CS if it possesses the token. Token based algorithms use sequence number with every request for token.

II. LITERATURE SURVEY:

In the Lamport's algorithm, a node which needs to enter CS should broadcast its CS request, wait for acknowledge from all nodes, and finally enter the CS. After exiting CS, the node should broadcast a release message indicating *release of CS*. This algorithm sends $3(N-1)$ messages for each CS [1].In order to reduce message complexity, Ricart Agrawala has made improvements to Lamport's algorithm which sends only $2(N-1)$ messages per each CS [2]. Ricart Agrawala achieved this by removing *release message* step of Lamport's algorithm. Instead, the node exiting the CS only sends release message to the nodes which have sent request messages and wait for permission. A queue for holding requests that come from other nodes is also added to the algorithm. Agrawal and El Abbadi [7] and Maekawa [6] have proposed quorrum-based algorithms which dramatically reduce the message complexity and belong to permission-based approach [7, 6]. Agrawal and El Abbadi use tree-structured quorums which require permission from only O (log2 (N)) nodes in best case, and $O(N)$ in the worst case. Maekawa proposed a new DMX algorithm which only uses c 2 $\sqrt{N}$messages to create a mutual exclusion in a computer network. The network consists of number of subsets whose intersection set is not empty. Additionally, there are also a number of algorithms which use token-based DMX approach [3, 4]. Main idea of token-based algorithms is that the node having the token will have opportunity to enter CS. One of the most popular approaches is Suzuki-Kasami's algorithm [3] which uses N messages per each CS. Another one is Raymond's tree based algorithm [4] which reduces the message complexity using its dynamic tree structure.

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.

Utilizing the clustered 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.

III. BACKGROUND

A. Backbone formation in MANET

The backbone in wireless ad hoc networks is a path connecting cluster heads that supports a network-wide infrastructure for routing and inter cluster operations. There are energy-efficient [9] , multicast oriented[10] , tree based[11] and dominating set based backbones which are proposed for MANETs in literature.

B. Clustering in MANET

Clustering is a fundamental approach to manage the MANET services. In clustered networks, nodes are either classified as cluster members, cluster heads or optionally cluster gateways. A cluster member is an ordinary cluster node which sends its request to its cluster head. A cluster head is responsible for managing intra cluster requests and participating in inter cluster operations. The most significant benefit of clustering is that the network load is distributed more balanced in clustered networks compared to the networks with no infrastructure. Lastly, the clustering method supports a hierarchical management scheme which upper layers can take advantage of it

C. 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 CS when a process makes a request to enter it, the system is lightly loaded. Otherwise, when there is a high demand for the CS 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, response time and the synchronization delay as described below:

• Number of Messages per Request (M): 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.

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

IV. PROPOSED ALGORITHM

This algorithm combines Suzuki kasami's algorithm and Ricart agrawala algorithm. For a large distributed application involve n number of nodes, using either of two algorithm leads to increase in synchronization delay and number of messages per request. So to minimize these, in this algorithm the nodes are grouped in clusters and each cluster has a cluster leader. Suzuki-Kasami's algorithm is used inside clusters and Ricart-Agrawala algorithm is used to get permission among the cluster leaders.

A. System model and Data structures

In Ricarta-agrawala-suzuki's algorithm, we have two types of nodes, namely, ordinary and cluster leader. Ordinary nodes only have information about their own neighbors and use Suzuki-Kasami's algorithm to enter the CS. These nodes broadcast a CS request to nodes in its respective cluster. Leader nodes are dedicated to get permission from other clusters. The leader nodes can process two threads in order to act as leader and ordinary nodes simultaneously. Their job is to send external request message when their cluster needs to enter CS and give the permission when other clusters need to enter CS.

Data structures used in algorithm 1 and 2

Send_msg – requesting broadcast the message to all the nodes in its cluster.

Get_token - Requesting node sends the message to leader node to get the permission from the other clusters.

ExReq-Cluster leader broadcast request message to other cluster leader for getting permission.

req[j] – denotes the array of integer for the sequence number of the latest request message maintained at node j, which is used for requesting to get the token from the other nodes.

last[j] – denotes the sequence number of *the latest visit* to CS for process j.

state – denotes the state of the critical section

T – denotes the unique timestamp generated by the node for request message.

Pi – denotes the node of different cluster.

Pl & pi – denotes the leader nodes.

Algorithm 1 [Intracluster]

{Program of process j}

Initially, ∀i: req[i] = last[i] =0; timeout = 60;

*** Entry protocol ***

state :=WANTED;
req[j] := req[j] + 1;
Send_msg (j, req[j]) to all;
Wait until token (Q, last) arrives && timeout = = 0;
If timeout <= 0 then
Get_ token message to leader node;
state:= HELD;
Critical Section
*** Exit protocol** *
last[j] := req[j]
∀k ≠ j: k is in Q ∧ req[k] = last[k] + 1 → append k to Q;
if Q is not empty → send (tail-of-Q, last) to head-of-Q;
state:= RELEASED;
Upon receiving a request (k, num)
req[k] := max(req[k], num)

When there is an internal Suzuki-Kasami's algorithm request, InReq, inside the cluster then Entry protocol of algorithm1 is invoked. In absence of token within the cluster, respective node sends Get_token message to the cluster leader.

Algorithm 2 [Intercluster]
* Entry protocol *
state := RELEASED;
To enter the section
state := WANTED;
Broadcast request to all processes; processing deferred here
T := request's timestamp;
Wait until (number of replies received = (N – 1));
state := HELD;
generates the token and sends to requested pj
* Exit protocol *
state := RELEASED;
reply to any queued requests;
Upon receiving a request <Ti, pi> at pl (i ≤ l)
if (state = HELD or (state = WANTED and (T, pl) < (Ti, pi))) then
queue request from pi without replying;
else
reply immediately to pi;
end if

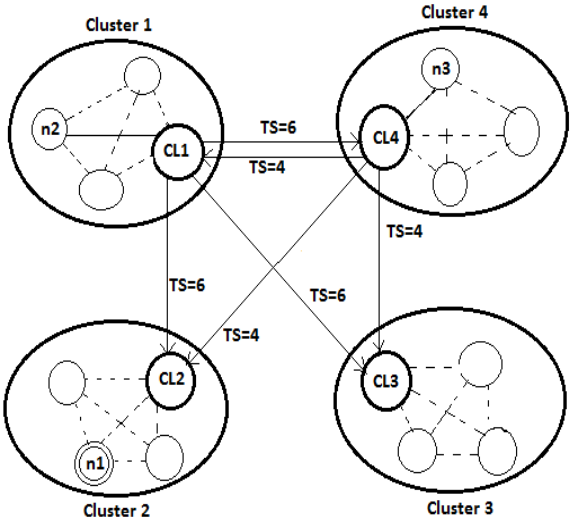

Fig 1:System structure

In fig 1, n1 node is executing in CS,n2 and n3 wants to enter CS so sends Get_Token message to its respective cluster leader. Then cluster leaders broadcasts request message with timestamp.
When the cluster leader having token receives the external request, ExReq, it sends an ordinary Suzuki-Kasami's request to its cluster for token ,once it receives the token, keeps it with itself and sends reply or gives permission to the requester. Cluster leaders add a level of indirection and obtain the illusion that there are only cluster leaders which are implementing Ring algorithm.

V. ANALYSIS OF THE PROPOSED ALGORITHM

In this section, we present the theoretical analysis of Ricart-agrawala-suzuki's algorithm along with relevant proofs. This theoretical analysis contains correctness, energy consumption, synchronization delay, and response time for the proposed algorithm
.
A. Correctness of Ricart-agrawala-suzuki's *Algorithm.*
The correctness of Ricart-agrawala-suzuki's algorithm is examined according to safety and liveness attributes of the algorithm.
1) Safety. Safety of Ricart-agrawala-suzuki's algorithm is analyzed from the single token existence and mutual exclusion points of view, which will be discussed in Theorem 3.

Lemma1. *In* Ricart-agrawala-suzuki's *algorithm at most one token exists in the cluster.*

Proof. Assume the contrary. In this case, there exists more than one token in the network concurrently. Assume that the nodes having tokens are in the same cluster. Suzuki- Kasami's algorithm is used in intracluster communication; thus multiple token existences are impossible [3]. On the other hand, assume that more than one node belonging to different clusters is in the CS at the same time. Since Ricart-Agrawala's algorithm is used for inter cluster communications, this is also not possible [6]. There is no other possibility; therefore we contradict our assumption.

Lemma 2. *At most one node can be in the CS at any time, ensuring mutual exclusion.*

Proof. Assume the contrary, that more than one node can execute CS at any time concurrently. In Algorithm1, when a node receives token, it enters CS. If there is more than one token in the network, then more than one node can
Execute CS at the same time. However, it is proven in Lemma 1 that this case is not possible. Therefore, we contradict our assumption.

Theorem: Ricart-agrawala-suzuki's a*lgorithm is deadlock- and starvation free.*

Proof. We use Suzuki-Kasami's algorithm for intra cluster communication and Ricart agrawala algorithm for intercluster, communication. Both algorithms are deadlock- and starvation free which is proven in [5, 6]. Thus, Raysuz's algorithm is deadlock and starvation-free.

B. Synchronization delay and response time

The synchronization delay of Suzuki-Kasami's algorithm is NT for a network consisting of N nodes, and T to indicate the unit time for sending a message.

The synchronization delay of Ricart-agrawala's algorithm is NT. Due to clustering of the whole distributed network the synchronization delay and the response time is reduced in the cluster if token is already present in that cluster and this is the best case of our proposed algorithm.

If the token is not present in that cluster and it is present in the different cluster of the network and that node is just entered in the CS then synchronization delay is 2 (NT) of Ricart-agrawala-suzuki's algorithm at worst case

VI. CONCLUSION

In this work, we proposed a hierarchical distributed mutual exclusion algorithm for mobile ad hoc networks. The communication infrastructure to run the algorithm consists of a number of clusters of mobile nodes where each cluster is represented by a leader and the *leaders* are connected to form a logical ring. Due to this hierarchical structure, significant gains in total message complexities, response times and synchronization delays conform to theoretical analysis

REFERENCES

[1] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Communications of the ACM*, vol. 21, no. 7, pp. 558–565, 1978.

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

[3] I. Suzuki and T. Kasami, "A distributed mutual exclusion algorithm," *ACM Transactions on Computer Systems*, vol. 3, no.4, pp. 344–349, 1985.

[4] K. Raymond, "A tree-based algorithm for distributed mutual exclusion," *ACM Transactions on Computer Systems*, vol. 7, no.1, pp. 61–77, 1989.

[5] M. Singhal, "A heuristically-aided algorithm for mutual exclusion for distributed systems," *IEEE Transactions on Computers*, vol. 38, no. 5, pp. 70–78, 1989.

[6] M. Maekawa, "A root N algorithm for mutual exclusion in decentralized systems," *ACM Transactions on Computer Systems*,vol. 3, no. 2, pp. 145–159, 1985.

[7] D. Agrawal and A. El Abbadi, "An efficient solution to the distributed mutual exclusion problem," *ACM Transactions on Computer Systems*, vol. 9, no. 1, pp. 1–20, 1991.

[8] S. Lodha and A. Kshemkalyani, "Afair distributedmutual exclusion algorithm," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 6, pp. 537–549, 2000.

[9] L. Haitao & R. Gupta, (2004) "Selective Backbone Construction for Topology Control in Ad Hoc Networks". In Proc. of the Intl. Conf. on Mobile Ad-hoc and Sensor Systems, pp. 41-50. International Journal of Computer Networks & Communications (IJCNC) Vol.4, No.2, March 2012 147

[10] W. Ya-feng, X. Yin-long, C. Guo-liang, & W. Kun, (2004) "On the Construction of Virtual Multicast Backbone for Wireless Ad Hoc Networks". In Proc. of the IEEE Intl. Conf. on Mobile Ad-hoc and Sensor Systems, pp. 25-27.

[11] S. Srivastava & R. K. Ghosh, (2002) "Cluster based Routing using a k-tree Core Backbone for Mobile Ad hoc Networks". In Proc. of the 6th Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications. pp. 14-23.