

Cluster Based Distributed Mutual Exclusion Algorithms for Mobile Networks

Kayhan Erciyes

California State University San Marcos,
Comp. Sci. Dept., 333 S.Twin Oaks Valley Rd.,
San Marcos CA 92096, U.S.A.
kerciyes@csusm.edu

Abstract. We propose an architecture that consists of a ring of clusters for distributed mutual exclusion algorithms in mobile networks. The mobile network is partitioned into a number of clusters periodically using a graph partitioning algorithm called *Fixed Centered Partitioning* first. Each cluster is represented by a coordinator node on the ring which implements various distributed mutual exclusion algorithms on behalf of any member in the cluster it represents. We show the implementation of Ricart-Agrawala and Token-based algorithms on this architecture. The message complexities for both algorithms are reduced substantially using the proposed architecture . . .

1 Introduction

Mobile ad hoc networks do not have fixed infrastructure and consist of mobile wireless nodes that have temporary interconnections to communicate over packet radios. Clustering, that is, partitioning of the mobile network graph into smaller subgraphs, can be used to solve various problems such as routing and mutual exclusion in such networks. In general, distributed mutual exclusion algorithms may be classified as permission based or token based. In the first case, a node would enter a critical section after receiving permission from all of the nodes in its set for the critical section. For token-based algorithms however, processes are on a logical ring and possession of a system-wide unique token would provide the right to enter a critical section. Susuki-Kasami's algorithm [10] (N messages) and Raymond's tree based algorithm [7] ($\log(N)$ messages) are examples of token based mutual exclusion algorithms. Examples of nontoken-based distributed mutual exclusion algorithms are Lamport's algorithm [5] ($3(N-1)$ messages), Ricart-Agrawala (RA) algorithm ($2(N-1)$ messages) [8] and Maekawa's algorithm [6]. *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.

Distributed mutual exclusion in mobile networks is a relatively new research area. A fault tolerant distributed mutual exclusion algorithm using tokens is discussed in [12] and a *k-way* mutual exclusion algorithm for ad hoc wireless networks where there may be k processes executing a critical section at any time

is presented in [13]. In this study, we propose a model to perform distributed mutual exclusion algorithms in mobile networks. We first partition the mobile network into a number of clusters using the multilevel graph partitioning heuristic we developed called *Fixed Centered Partitioning* (FCP) [1] that is executed by a special node called the *Central Coordinator*. Upon any changes of configuration or periodically gathering of the changes, the central coordinator starts a new configuration process by partitioning the network graph into new clusters. The nodes in the cluster including the nodes that have connections to other clusters are called *neighbor nodes*. The coordinator chooses one of the neighbor nodes in each cluster as the cluster *coordinator* and sends the cluster connectivity information and neighbor connectivity information to it. These coordinators perform the required critical section entry and exit procedures for the nodes they represent. Using this architecture, we improve and extend the RA and Token Passing algorithms described in [2] to mobile networks 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. The rest of the paper is organized as follows. Section 2 provides the background by describing FCP along with a review of performance metrics of mutual exclusion. The extended RA algorithm on the proposed model called *Mobile_RA* is described in Section 3. The second algorithm implemented on the model uses Token Passing and is called *Mobile_TP* as briefly described in Section 4. Finally, discussions and conclusions are outlined in Section 5.

2 Background

2.1 Partitioning of the Mobile Network

Graph partitioning algorithms aim at providing subgraphs such that the number of vertices in each partition is averaged and the number of edges cut between the partitions is minimum with a total minimum cost. An arbitrary network can be constructed as an undirected connected graph $G = (V, E, w)$ where V is the set of routing nodes, E is the set of edges giving the cost of communication between the nodes and $w: E \rightarrow \mathfrak{R}$ is the set of weights associated with edges. *Multilevel partitioning* is performed by coarsening, partitioning and uncoarsening phases [3]. During the coarsening phase, a set of smaller graphs are obtained from the initial graph. In the maximal matching, vertices which are not neighbors are searched. In Heaviest Edge Matching (HEM), the vertices are visited in random order, but the collapsing is performed with the vertex that has the heaviest weight edge with the chosen vertex. In Random Matching (RM) however, vertices are visited in random order and an adjacent vertex is chosen in random too. The coarsest graph can then be partitioned and further refinements can be achieved by suitable algorithms like Kernighen and Lin [4]. Finally, the partition of the coarsest graph is iteratively reformed back to the original graph.

We provide a partitioning method, FCP [1], where several fixed centers are chosen and the graph is then coarsened around these fixed centers by collapsing the heaviest or random edges around them iteratively. Different than [3], FCP

does not have a matching phase, therefore iterations are much faster. FCP requires the initial marking of the fixed centers. One possible solution is to choose the fixed centers randomly so that they are all at least some bounded distance from each other. The heuristic we used is $h = 2d / k$ where d is the diameter of the network and k is the number of partitions (clusters) to be formed. The time complexity of the total collapsing of FCP is $O(n)$ and FCP provides much favorable partitions than CM and RM in terms of the average edge cost, time to partition a graph and the quality of the partitions experimentally [1]. Fig. 1 shows a mobile network of 20 nodes which is partitioned into clusters A, B, C and D using FCP. Nodes 16, 20, 2 and 14 are the initial centers, 17 is the central coordinator and also the cluster coordinator for cluster D, and the coordinators for clusters A, B, C and D are 10, 15 and 19 and they form a ring together with 4 and 7.

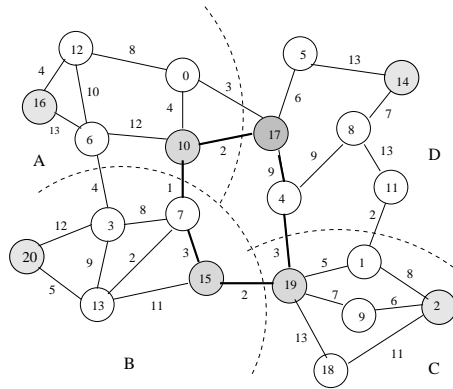


Fig. 1. Partitioning of the Mobile Network

2.2 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 [9].

3 Extended Ricart-Agrawala Algorithm

For distributed mutual exclusion in mobile networks, we propose a hierarchical architecture where nodes form clusters and each cluster is represented by a *coordinator* in the ring as in Fig. 1. The relation between the cluster coordinator and an ordinary node is similar to a central coordinator based mutual exclusion algorithm. 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 algorithm for the Mobile_RA coordinator is shown in Fig. 2.

```

Process Mobile_RA_Coord;
begin
  repeat
    msg=receive_msg();
    switch (msg.type):
      case Node_Req : send(Coord_Req, next_coord);
                    insert_msg(Node_Req, wait_queue);
      case Coord_Req : If Idle OR all local requests > timestamps
                      send(Coord_Rep, next_coord)
                      Else Block the Coord_Req message;
                        insert_msg(Coord_Req, wait_queue);
      case Coord_Rep : send(Node_Rep, node);
      case Node_Rel  : If wait_queue <> empty and top_queue <> local
                      For every external request with < timestamps
                        send_all(Coord_Replies, next_coord);
    until forever
  end.

```

Fig. 2. Mobile_RA Coordinator Algorithm

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. Fig. 3 shows an example scenario for the Mobile_RA Algorithm in the network of Fig. 1. The following describes the events that occur :

- Nodes 12, 2 and 5 in clusters A, C and D make critical section requests with messages $Node_Req_{12}(1)$, $Node_Req_2(3)$ and $Node_Req_5(2)$ with the shown timestamps.
- The coordinator for clusters A, C and D (C_A , C_C , C_D) form request messages R_{12} and R_2 and R_5 and send these to the next coordinators on the ring, Steps 1 and 2 are shown in Fig. 3.(a).
- C_B passes R_{12} to its successor C_C as it has no pending requests in its cluster. C_C also passes R_{12} as R_{12} has a lower timestamp than the pending request in its cluster ($Node_req_2(3)$). C_D passes R_{12} back to C_A as this request has a lower time stamp than its own too, but it blocks the request R_2 by C_C as this incoming request has a higher timestamp.
- C_A now has received its original request back meaning all of the coordinators request back meaning that either they have no pending requests or their pending requests all have higher timestamps. C_A now sends a $Coord_Rep$ message to node 12 which can enter its critical section. Steps 3 and 4 are depicted in Fig. 3.(b).

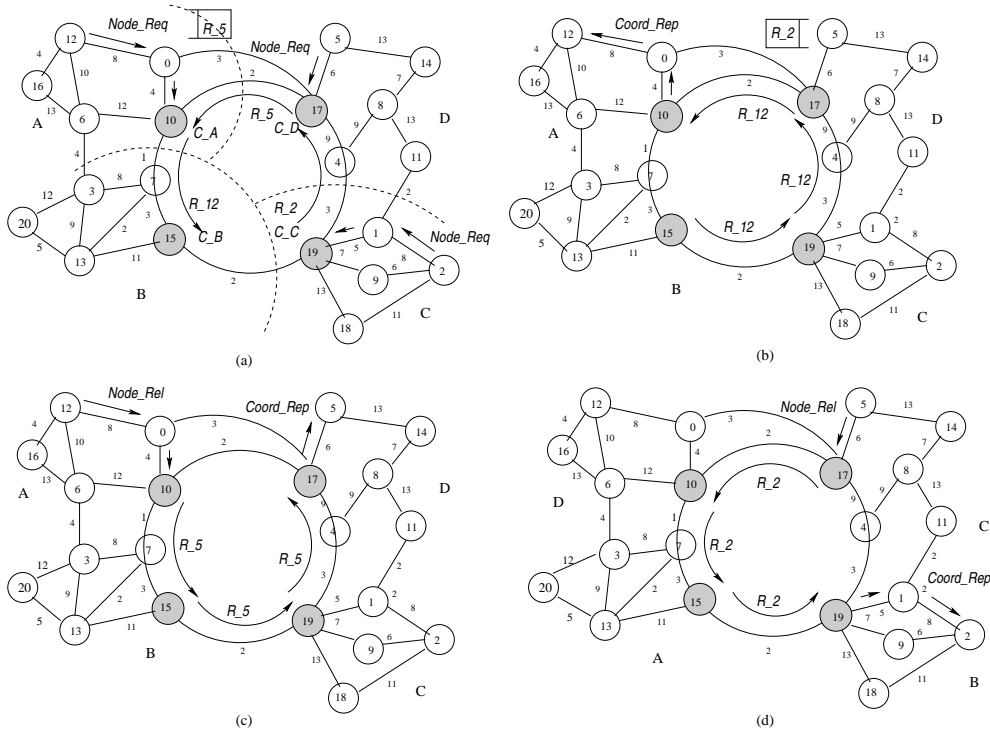


Fig. 3. Operation of the Mobile_RA Algorithm

- Node 12 in cluster A sends $Node_Rel$ message to its coordinator C_A to inform that it has finished executing its critical section. C_A now sends a

- reply to the request it was blocking (R_5). C_C passes R_5 to C_D as it has a lower timestamp than its request R_2 .
6. C_D now has its reply now for request $Node_Req_5(2)$ and sends a reply message ($Coord_Rep$) to node 5 which can then enter its critical section. Steps 5 and 6 are shown in Fig. 3.(c).
 7. When node 5 finishes executing its critical section, it sends a release ($Node_Rel$) message to its coordinator C_D which now sends the reply message R_2 it was blocking to the ring.
 8. C_A and C_B pass this message and C_C now has the reply from the ring and can send the reply to node 2 ($Coord_Rep$) which can enter its critical section. Steps 7 and 8 are shown 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 $12 \rightarrow 5 \rightarrow 2$ in the order of the timestamps of the requests.

Theorem 1. *The total number of messages per critical section using the Mobile_RA Algorithm is $k+3d$ where k is an upper bound on the number of neighbor nodes in the ring including the cluster coordinators and d is an upperbound on the diameter of a cluster.*

Proof. An ordinary node in a cluster requires three messages (*Request*, *Reply* and *Release*) per critical section to communicate with the coordinator. Each of these messages would require maximum d transfers between a node and the coordinator. The full circulation of the coordinator request (*Coord_Req*) requires k messages resulting in $k + 3d$ messages in total.

Corollary 1. *The Synchronization Delay (S) in the Mobile_RA Algorithm varies from $2dT$ to $(k + 2d - 1)T$.*

Proof. When the waiting and the executing nodes are in the same cluster, the required messages between the node leaving its critical section and the node entering are the *Release* from the leaving node and *Reply* from the coordinator resulting in $(2dT)$ message times for S_{min} . However, if the nodes are in different clusters, the *Release* message has to reach the local coordinator in d steps, circulate the ring through $k-1$ nodes to reach the originating cluster coordinator in the worst case and a *Reply* message from the coordinator is sent to the waiting nodes in d steps resulting in $S_{max}=(k-1)T + 2dT=(k + 2d - 1)T$.

Corollary 2. *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.*

Proof. According to Theorem 3, the total number of messages required to enter a critical section is $k + 3d$. If there are no other requests, the response time for a node will be $R_{light}=(k+3d)T+E$ including the execution time (E) of the critical section. If there are w pending requests at the time of the request, the minimum value R_{heavy_min} is $w(2dT + E)$. In the case of S_{max} described in Corollary 1, R_{heavy_max} becomes $w((k + 2d - 1)T + E)$ since in general $R_{heavy}=w(S + E)$.

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 summarised in Tab. 1.

Table 1. Performance of Mobile_RA Algorithm

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

4 Extended Token Passing Algorithm

We propose an extended implementation of the Token Passing (TP) Algorithm for mobile networks. Token is circulated in the ring only and the coordinator for each cluster consumes the token if it has a pending request for the token from any of its nodes, otherwise it passes the token to the next coordinator in the ring. A node that finishes execution of its critical section returns the token to its coordinator which may forward it to any other node waiting for the token in its cluster or pass it to the next coordinator in the ring. Since there is only one token, mutual exclusion is guaranteed. The message complexity for Mobile_TP is $O(k + 3d)$ since $(k + 3d)$ is an upperbound on the number of messages depending on the current location of the token when a request for a critical section is made. The Synchronization Delay (S) and the Response Time values of Mobile_TP are the same as the Mobile_RA Algorithm.

5 Conclusions

We proposed a framework to implement distributed mutual exclusion algorithms in mobile networks where the network is represented by a dynamically changing graph and this graph is partitioned into clusters at regular intervals. We showed that this model provides improvement over message complexities of Ricart and Agrawala and Token-based algorithms and also the time required to execute a critical section. 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. The order of magnitude of improvement over the classical RA and the Token-Based algorithms using our model is achieved at the expense of increased response times and synchronization delays. 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 algorithms as in [11] which is not discussed here. Our work is ongoing and we are looking into evaluating these algorithms

and implementing k-way distributed mutual exclusion algorithms in simulated mobile networks. Partitioning of the network graph using FCP can also be performed in parallel by the coordinators which would improve performance.

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

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

References

1. Erciyes, K, Marshall, G. : A Cluster-based Hierarchical Routing Protocol for Mobile Networks, ICCSA 2004, SV-Lecture Notes in Computer Science, to appear, (2004)
2. Erciyes, K, : Distributed Mutual Exclusion Algorithms on a Ring of Clusters, ICCSA 2004, SV-Lecture Notes in Computer Science, to appear, (2004)
3. Karypis, G., Kumar, V. : Multilevel k-way Partitioning scheme for irregular graphs. Journal of Parallel and Distributed Computing, Vol. 48, (1998), 96-129
4. Kernighan, B., Lin, S. : An Effective Heuristic Procedure for Partitioning graphs, The Bell System Technical Journal, (1970), 291-308
5. Lamport, L. : Time, Clocks and the Ordering of Events in a Distributed System, CACM, Vol. 21, (1978), 558-565
6. Maekawa, M. : A sqrt(n) Algorithm for Mutual exclusion in Decentralized Systems, ACM Transactions on Computer Systems, Vol. 3(2), (1985), 145-159
7. Raymond, K. : A Tree-based Algorithm for Distributed Mutual Exclusion. ACM Trans. Comput. Systems, 7(1), (1989), 61-77
8. Ricart, G., Agrawala, A. : An Optimal Algorithm for Mutual Exclusion in Computer Networks, CACM, Vol. 24(1), (1981), 9-17
9. Shu, Wu : An Efficient Distributed Token-based Mutual Exclusion Algorithm with a Central Coordinator, Journal of Parallel and Distributed Processing, Vol.62(10), (2002), 1602-1613
10. Susuki, I., Kasami, T. : A Distributed Mutual Exclusion Algorithm, ACM Trans. Computer Systems, Vol. 3(4), (1985), 344-349
11. Tunali, T, Erciyes, K., Soysert, Z. : A Hierarchical Fault-Tolerant Ring Protocol For A Distributed Real-Time System, Parallel and Distributed Computing Practices, Vol. 2(1), (2000), 33-44
12. 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
13. 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 POMC 2001, (2001)