

A Quorum Based Distributed Mutual Exclusion Algorithm for Multi-Level Clustered Network Architecture

Mohammad Ashiqur Rahman¹ and Md. Mostofa Akbar²

¹ Military Institute of Science and Technology, Mirpur Cantonment, Dhaka

² Bangladesh University of Engineering and Technology, Dhaka

Abstract. Different permission-based algorithms have been proposed for the solution of the Mutual Exclusion problems. With the emergence of peer-to-peer computing, the distributed applications spread over a large number of nodes. Cluster-based solutions are scalable for large number of participants. Some algorithms are proposed using cluster topology. But the number of participating nodes is increasing everyday. So here we propose a general permission-based solution for multi-level clustered network. We also find the optimal level of clustering for an especial case.

1 Introduction

In distributed systems different processes are running on different nodes of the networks and they often need to access shared data and resource, or need to execute some common events. Their uses should be consistent and so any access to them should be mutually exclusive. The portion of an event or application, where any shared components or common events are needed to be used, is the Critical Section (CS). Mutual Exclusion (ME) algorithms ensure the consistent execution of CS. As the shared memory is absent in distributed systems the solutions of the ME problem is not straight forward. Due to the enormous importance of ME and the difficulty of its solution, this is an extensive research area since last three decades. The classic algorithms for Mutual exclusion that have been proposed for fixed networks can be classified in two types: centralized and distributed approaches. In the centralized solutions a node is designated as coordinator to deliver permission to the other nodes to access their Critical Section (CS) while in the distributed solutions the permission is obtained from consensus among all network nodes.

On the distributed systems, distributed mutual exclusion algorithms are mainly classified in two categories: token based [1–3] and permission based [4–7]. Permission based mutual algorithms impose that a requesting node is required to receive permissions from other nodes (a set of nodes or all other nodes). In the token-based mutual exclusion algorithms, a unique token is shared among the set

M. Kaykobad and Md. Saidur Rahman (Eds.): WALCOM 2007, pp. 121–135, 2007.

of nodes. The node holding the token is allowed to enter its critical section. The basic idea of token-based algorithms is simple: a node must own the unique token (sometimes cited as privilege message) before entering the CS. So, in the best case, no communication is necessary since the token may be available locally. Otherwise, a mechanism is needed to locate the token. In [2], a spanning tree of network for locating the token is used and it shows that the average number of messages exchanged in this protocol is $O(\log n)$. But token-based algorithms suffer from poor failure resiliency. In particular, if the node holding the token fails, complex token regeneration protocols must be executed [8].

Ricart and Agrawala proposed the fair algorithm [4] that needs $2(n-1)$ messages for a node to use the critical section. This algorithm is the first permission-based ME algorithm where a node need to collect permission from all other nodes for CS access. Though the algorithm is deadlock and starvation free, it is vulnerable to node and communication failures and it is expensive in communication cost too.

There is elegant class of permission-based algorithms [7] that use concept of quorums to achieve mutually exclusive access of CS. A node needs to achieve permissions from all of the nodes of a quorum to access CS. Quorum based algorithms are resilient to node and communication failures and often network partitioning and usually have lower communication cost. Communication cost of these algorithms is proportional to the quorum size. Therefore these algorithms try to achieve the two goals: small quorum size with high degree of fault tolerance. Its basic idea is to collect enough permission (votes) to guarantee the mutual exclusion. The majority quorum algorithm [13] can be considered as the first algorithm of this kind, where to attain mutual exclusion a node must obtain permission from a majority of nodes in the network. Maekawa [5] proposed an ME algorithm by imposing a logical structure on the network. In this scheme, a set of nodes is associated with each node, and this set has a nonempty intersection with all other sets corresponding to the other nodes, which guarantee the ME. The size of each of these sets is \sqrt{n} and so the algorithm costs \sqrt{n} order.

Garcia-Molina and Barbara [14] have properly defined the concept of quorums with the notion of coterie. A coterie is a set of sets with the property that any two members of a coterie have a nonempty intersection. Combining the idea of logical structures and the notion of coterie an efficient and fault tolerant quorum generation algorithm for ME is proposed by Agrawal and Abbadi [6]. Here the nodes form a logical binary tree which is used to generate quorums. The quorum forming in this algorithm is recursive. It can be regarded as attempting to obtain permissions from nodes along a root-to-leaf path. If the root fails, then the obtaining permissions should follow two paths: one root-to-leaf path on the left subtree and one root-to-leaf path on the right subtree. The algorithm tolerates both node failures and network partitions while in the best case incurring logarithmic costs considering the size of the network. But the cost increases with the increase of node failures.

However, at present the number of distributed nodes has become very large. And with the emergence of peer-to-peer computing, the distributed applications

have been spread over a large number of nodes. Again the latency gaps, between nodes interconnects, are very important issue. So number of nodes participating in any application as well as their location is crucial. But the classical algorithms illustrated above do not consider these matters. So, algorithms that reduce the number of participating nodes are needed.

Sometimes the nodes in a network are divided into several groups where each group is often called a cluster. According to this concepts some group based hybrid ME solutions [8, 9] are proposed. Actual goal of these proposed algorithms was to combine two different approaches in different layers- intra-group and inter-group. Two distributed ME solutions are presented by Erciyes [10] using a logical structure where clusters are arranged on a ring. In [11] Bertier proposed two token-based algorithms taking into account the hierarchical network topology, which reduce both latency cost and number of message. But these three solutions are modification of Naimi's token-based algorithm [3] for proxy-based cluster. As these algorithms are basically token-based, they suffer due to token failure. However, all of the above algorithms use two-layer network topology and run algorithm inside the cluster and among the clusters so that mutually exclusive access prevails. Though the number of participating nodes is reduced in the above approaches and so the cost, it is not sufficient according to the present situation. The networks are continuously growing and so they are becoming larger and larger day by day.

Here, we propose a multi-level clustered network architecture and give a general distributed ME solution based on this proposed network. It improves the performance by reducing participating nodes in ME algorithm. The algorithm for achieving ME is extension to that of [5] and [6]. Due to the hierarchical network structure we incorporate a coordination algorithm with the algorithm as well as modify the format of the messages. In section II we describe the proposed network architecture. In section III, the ME algorithm followed by its analysis is illustrated. Section IV includes the theoretical analyses of the algorithm. Performance analysis is given in section V.

2 Proposed Network Architecture

2.1 Description of the system

The distributed system is a collection of nodes where each node will execute a process. These nodes communicate by exchanging messages. A logical channel connects each pair of nodes. Communication will be taken as reliable. The message delay is finite. Moreover, each channel is assumed to have infinite capacity, and to be FIFO (First-In First-Out). The processes fail in accordance with Fail-Stop model [15].

2.2 Network Architecture

The nodes in a network are partitioned into several nonintersecting groups. Each group is often called a cluster. In multilevel clustering, within a cluster, the nodes

can form some smaller clusters again and so forth. In the multilevel clustering, Level of Clustering is introduced. Another important feature is Message Router which plays an important role in the proposed algorithm.

Level of Clustering: Here l -level of clustering is proposed where l can be any positive integer number. Figure 1, presenting in the next page, shows the topology for 2-level of clustering. In the figure, P, Q and R act as Level 1 clusters while A, B and C act as Level 2 clusters. A collection of nodes forms a Level 1 cluster. A collection of nodes forms a Level l cluster. And a collection of Level l clusters forms a Level $(l - 1)$ cluster and so on. Level 0 is considered as a cluster with whole network in this research. In the figure 1, P, Q and R form the Level 0 cluster. There is one and only one cluster at level 0. It is the topmost cluster and so can be called as root cluster. Level l clusters are the bottommost clusters and so can be called as leaf clusters.

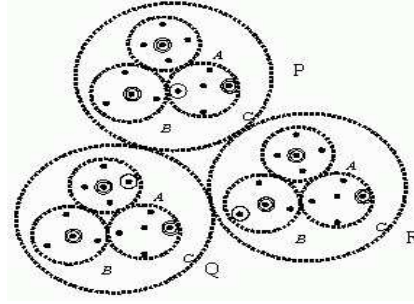


Fig. 1. A network with 2-level of clustering.

If the l is 0, then there is only one cluster with all the nodes of network. So in this case, we can say that no clustering is done to the network. The number of clusters at a level decreases from l to 0 Level. However, l -level of clustering forms a hierarchical structure of nodes.

Message Router: In each cluster of any level ($l > 0$) there is a message router, representing the cluster. In the figure 1 the message routers of the Level 2 clusters (all A, B and C clusters) are surrounded by two circles while the message routers of the Level 1 clusters (P, Q and R) are surrounded by single circle. The main and only task of a message router is to represent the cluster at upper level cluster and according to this role to communicate between different clusters. As some Level $k + 1$ clusters form a Level k cluster, the members of a Level k cluster are also some clusters. However, in this k cluster not all the nodes of the member clusters but only their message routers play for their associated clusters. Therefore, it can simply be said that the Level k cluster is made up with the message routers

of some Level $k + 1$ clusters. Level 0 cluster needs no message router as this is the only cluster in this topmost level. Actually if any information is needed to transmit from a cluster of a level to a cluster of different level then it is done through the message router.

3 Proposed Solution

We now present the brief outline of the proposed algorithm followed by a detailed description of the algorithm along with the state diagram of the proposed algorithm.

3.1 Brief Outline

Here the Mutual Exclusion Algorithm is executed in different level of clusters. In each level a cluster participates. Inside a cluster a classic ME algorithm runs. Any algorithm, whether centralized or distributed, is applicable here. A simple Coordination Algorithm is proposed to communicate between the executions of ME algorithm at different levels. Message routers play the main role here. However, how the ME algorithm works using the coordination algorithm is written below:

- Requests are generated at Level l clusters. These requests are processed in different levels, Level l to Level 0 sequentially. The ME algorithm is run in a cluster at each level. The selected cluster at each level is the one in which the requesting node falls. However the root cluster, the only cluster at Level 0, plays for each request.
- The CS requesting node first executes ME algorithm in the cluster, a Level l cluster, in which it falls. If its request is granted in this cluster it sends a special request to the message router of the cluster if and only if there are upper levels of clusters. The request is passed to the Level $l - 1$ cluster, in which that Level l cluster falls, through the message router for further processing. The message router will process the request in Level $l - 1$ level by executing the ME algorithm.
- When the message router gets the request (a special request only to a message router) from one of the nodes of its cluster, Level k ($0 \leq k \leq l$) cluster, it runs ME algorithm for the request in the immediate upper level cluster, Level $k - 1$ ($k - 1 \geq 0$) cluster, to which it is a member. Note that, the message router may also be a member of that Level k cluster. However, in Level $k - 1$ this message router is now the requesting node. If the permission is granted here then the requesting node will place the request to the higher level cluster, Level $k - 2$ cluster if any ($k - 2 \geq 0$), through the message router of this cluster. This process continues until the Level 0 cluster is reached.
- When a node requests it needs permission from two sides in two sequential steps: first from the nodes of its cluster and next from upper level through message router. If there is no upper level then only the consensus from same

level is required. After getting total consensus a node of Level k sends a special reply downward to the requesting node of the Level $k + 1$ ($k + 1 \leq l$) cluster. In this way the reply reaches the requesting node at Level l , the actual requestor. Now the requesting node has the total consensus and so it executes the CS mutual exclusively.

- After execution of CS, the requesting node sends release messages to the nodes inside its cluster, Level l cluster. In the mean time it also sends a release message to its message router. When the message router, which is a member of a Level k cluster, gets release from a node of a Level $k + 1$ cluster, it sends release messages to the member nodes of its Level k cluster. If Level $k - 1$ cluster exists ($k - 1 \geq 0$), then it sends a release message to its message router to transmit the release to upper Level.

3.2 Messages and Node States

Message types are depends basically on the classic algorithm that run in a cluster. Naturally they are REQUEST, REPLY, RELEASE, INQUIRE and YIELD [6]. INQUIRE and YIELD both used for distributed algorithms only. However, as here the proposed algorithm run in different levels of clusters and a single node (especially message router) may play in several levels, an important data, *Level No*, is also added with the usual parameters for these messages. Level No is the level number of the cluster in which the messages are being used.

The communication between the member nodes of a cluster and the message router of this cluster is done through SREQUEST, SREPLY and SRELEASE messages. When a node x gets the CS consensus from its cluster, Level k cluster, then it sends SREQUEST message to its message router y to process the request in the immediate higher level, Level $k - 1$, cluster. Note that, the message router is a member of Level $k - 1$ cluster. Message router y sent back SREPLY to the requesting cluster node x if it has the total permission, permission from inside the cluster as well as the permission from the upper level if any. SRELEASE is sent to the message router from the cluster node to propagate the release message to upper Level.

All of the messages have almost the same format as follows:

MESSAGE (*Source, Level, Timestamp*)

According to message types different types of messages are used. The parameter Level is used for the no of Level of the cluster in which the sender node is acting. Source has the id of the node which sends the message to a destination node. And Timestamp contains the logical timestamp of the message [12]. It is important for REQUEST message for ordering the requests to avoid deadlock and starvation. For other messages, this parameter has no importance and so can be omitted.

Each node may be in the following states at different levels:

- REQUESTING [$0 \dots l$]: Here l is the Level of Clustering. This state at a level is set when a node sends REQUEST to other nodes in a cluster of that level. The node remains in this state until it gets necessary replies from a quorum.

- **LOCKED** $[0 \dots l]$: This state at a level is set when a node sends **REPLY** against a request of the same level to a node and remains in this state until it gets a **RELEASE** message from that node.
- **BUSY** $[0 \dots l]$: When a requesting node at a level gets consensus in both ways, those are, from its cluster of that level and from the upper level, this state is set.
- **CREQUESTING** $[0 \dots l]$: When a requesting node at a level gets necessary consensus from its cluster of that level, it sends **CREQUEST** to its message router for upper level consensus, if any, and sets this state for this level.
- **INQUIRING** $[0 \dots l]$: If a node x gets request from a node y at a level which has less timestamp than a earlier received request and that is already replied, then x node sends a **INQUIRE** message to y , as well as, sets the state for this level.

3.3 State Diagram

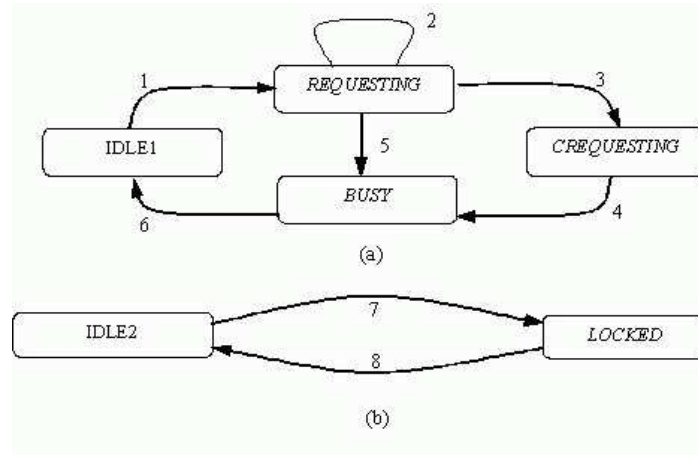


Fig. 2. State Diagrams, (a) and (b), for the proposed algorithm

State diagram of our proposed algorithm is very simple. Here a node faces two different state transitions while participating at any level. One, as in figure 2(a), happens at the time of requesting for CS for own self or on behalf of others as message router. Other state transition, as in figure 2(b), occurs at the time of giving consensus to a requesting node. Description of the different states is already given in above sections. But two new states: IDLE1 and IDLE2 are introduced here. When a node is not in REQUESTING or CREQUESTING or BUSY states, that node is in IDLE1 state. And when a node is not in LOCKED state the node is in IDLE2 state. The most important thing is that, a node

possesses a state of figure 2(a) and a state of figure 2(b) at the same time. And both sides' state transitions are occurred independently to each other.

All the state transitions, shown by number in figure2, are briefly described below by citing the conditions which make the state change and the actions which are executed at the time of state change.

- State transition 1: There are two cases to initiate the state transition.
 - A node needs to use CS by itself
 - It gets CREQUEST from a node because of being message router
 Action: The node sends REQUESTs to the nodes of a quorum, selected among the nodes of its cluster and so gets into REQUESTING state.
- State transition 2: A REQUESTING node gets REPLY message from the nodes of the requesting quorum.
- State transition 3: A REQUESTING node gets the REPLY messages from all of the nodes of the quorum. Again, the acting cluster level of the node is not the topmost level, that is, there is at least a level up of this level. Action: The REQUESTING node sends CREQUEST to the message router of its cluster and so gets into CREQUESTING state.
- State transition 4: A CREQUESTING node gets the CREPLY message from the message router. Action: The node is now in the BUSY state. Two different cases effect the other input actions.
 - Case 1: If the level of the requesting node is equal to level of clustering, then the node starts to use CS.
 - Case 2: The node is acting as message router and so sends CREPLY downward to the CREQUESTING node.
- State transition 5: A REQUESTING node at a level gets the REPLY messages from all of the nodes of the quorum. Again, the acting cluster level of the node is the topmost level, that is, the level is 0. Action: The node gets into the BUSY state. There are two cases here which effect the input action.
 - Case 1: If the level of clustering is also 0, then the node starts to use CS.
 - Case 2: The node is acting as message router and so sends CREPLY downward to the CREQUESTING node.
- State transition 6: There are two cases to initiate this state transition from BUSY state to IDLE1 state.
 - Case 1: The node at bottommost level finishes its CS use
 - Case 2: The node, at upper levels than bottommost level, gets CRELEASE from a node because of being message router.
 Action: The node is now IDLE1 state. Input actions depend on two different cases.
 - Case 1: The node is at the topmost level. So, it only sends RELEASE messages to the nodes of the quorum.
 - Case 2: The node is at levels other than the topmost level. So, it sends RELEASE messages to the nodes of the quorum as well as sends CRELEASE message to the message router of its cluster.
- State transition 7: When a node has a REQUEST of a node, the state transition from state IDLE2 to LOCKED is happened. Action: The node sends a REPLY to the REQUESTING node.

- State transition 8: There are two cases to initiate this state transition from LOCKED state to IDLE2 state.
 - Case 1: When the node receives a RELEASE or YIELD message from the locking node.
 - Case 2: When the node receives a YIELD message in reply of a INQUIRE message from the locking node.

3.4 Correctness Proof

Here we first prove the correctness of the algorithm for single level of clustering and using quorum based algorithm, tree-quorum algorithm, as a classic algorithm in any cluster. As the level is single that is there are two levels of clusters: Level 1 clusters and Level 0 cluster. In a Level 1 cluster, the member nodes of the cluster will participate in the ME algorithm while in Level 0 cluster, the members of this cluster, that is, the message routers of those Level 1 clusters will participate. In a simple view, we can say that, the ME algorithm will run in two level: inter a Level 1 cluster between its member nodes and inter the Level 0 cluster between the message routers of Level 1 clusters. Before proving correctness, we need to cite the following intersection property:

The Intersection Property: If g and h are quorums in C , then g and h must have a nonempty intersection, i.e., $g \cap h \neq \phi$.

In Agrawal and Abbadi [6], the intersection property is proved for tree-quorum. Now we will show how our proposed algorithm satisfies above correctness property of mutual exclusive access of a resource.

Theorem 1. *No two nodes are simultaneously in CS*

Proof. In a Level 1 cluster tree-quorum algorithm is used. So here the quorums satisfy the intersection property. Similarly in the Level 0 cluster, that is, between the message routers, tree-quorum algorithm is used and so the intersection property is also maintained here.

The proposed algorithm runs level wise, bottom to upward, and sequentially. Here a node needs to have consensus from two different quorums sequentially, first one is from the members of a Level 1 cluster and later one is from the members of the Level 0 cluster. So, a quorum according to this proposed algorithm is the combination of these two sub-quorums. We name this quorum combination as total-quorum.

Now, all the total-quorums are formed similarly here and each of them must have a quorum generated from the members of the Level 0 cluster. Therefore, each of any two total-quorums must have a quorum from the Level 0 cluster and according to the intersection properties of the tree-quorum algorithm there should be an intersection between these two quorums of Level 0 and so these two total-quorums have an intersection. So, the intersection property is satisfied by the proposed algorithm for single level of clustering.

However, for all level of clustering there is a Level 0 cluster and so any total-quorum has a quorum from this cluster. And thereby the intersection property is satisfied by the proposed algorithm for l -level of clustering. *Q.E.D.*

4 Algorithm Analysis

Here the analysis of the proposed algorithm is presented taking the tree-quorum algorithm [6], as the classic algorithm, which runs in the clusters of any level. Message cost is proportional to quorum size. If n is the number of nodes of the network and f is the probability of a node to be available to form a quorum then according to tree-quorum algorithm the following power equation is found for cost (c).

$$c_h = \frac{(2-f)^h - f}{1-f} \quad \text{when } f \neq 0 \quad (1)$$

$$\text{or} \quad c_h = h + 1 \quad \text{when } f = 0 \quad (2)$$

Here, height of the tree (binary tree), $h = \log_2(n+1) - 1$. Approximately, $h = \log_2 n - 1$ for large n .

4.1 Optimal Cluster Size

Let the level of clustering is 1. So two levels of clusters: Level 1 cluster and Level 0 cluster, the cluster of Level 1 clusters. Now let the following parameters:

- The number of nodes in the network, n .
- Cluster size of each of the Level 1 clusters, C . So the number of Level 1 clusters is n/C . And this is the cluster size of the Level 0 cluster.
- Height of the tree made up with the whole network is $h = \log_2 n - 1$. And Level of the tree composed by the clusters' message routers in the Level 1 cluster is $h' = \log_2 C - 1$.

Taking the derivative of cost (according to the equation 1 and h') we get the optimal value of h' and so C , that is, $C = \sqrt{n}$. So the size of Level 0 cluster is also \sqrt{n} . In 1-Level of clustering, cluster sizes of the clusters of both levels are equal, square-root of the network size, for optimality.

Remarks: The cluster size is inversely proportional to the number of clusters, that is, the cluster size of Level 1 cluster is inversely proportional to that of Level 0 cluster. As the cost equation is exponential and same equation is used in both levels, for optimality both levels try to minimize its size and ultimately set to the point where both cluster sizes are same. The optimal value of C , \sqrt{n} , suggests the same thing, that is, the cluster sizes of both levels are same. This prove is done for 1-level of clustering. We can easily extend this concept for l level of clustering and so have a lemma.

Lemma 1. *For optimality, the cluster sizes of different Level of clusters are same for l level of clustering.*

4.2 Optimal Level of Clustering

For l level of clustering on n number of nodes, according to the Lemma 1 we take that $n = C^l + 1$. This means, at first the n is partitioned into C number of clusters. So each cluster contains n/C nodes in average. Then in each cluster, the n/C nodes are partitioned again into C clusters. So, at this level, the clusters will have n/C^2 nodes each. This clustering process will continue up to the l times of clustering and at last bottommost clusters will contain n/C^l nodes each. Now we will find the optimal value of l for the value n using the tree-quorum algorithm. However, in l level of clustering there are different clusters at different $l + 1$ levels each with cluster size C . The proposed algorithm runs the tree-quorum algorithm in a cluster at each level and so in total the tree-algorithm runs in $l + 1$ clusters.

Now, let h is height of the tree formed by whole network, that is, n nodes and $\frac{h+1}{l+1} - 1$ is the height of the tree formed in each cluster by C participating

nodes. According to the equation 1 the total cost is $c = \sum_1^{l+1} \frac{(2-f)^{\frac{h+1}{l+1}-1} - f}{1-f} +$ Layer to Layer communication cost

$$\text{or } c = \frac{(l+1)(2-f)^{\frac{h+1}{l+1}-1} - (l+1)f}{1-f} + l$$

Taking the derivative of the above equation we get the optimal value of l :

$$l = \frac{(h+1) \ln 2 - f}{1+k} - 1 \quad (3)$$

An equation solving tool, "DeadLine" is used to find out k . Approximate value of k is $0.46306 * (1 - f) - f$. However, this optimality of l is valid for $f < 1$. If $f = 1$, then optimal value for l is 0, that is, theoretically no clustering is needed.

4.3 Analytical Result and Associated Findings

Here we present two different graphs showing different level of clustering and the optimal level of clustering according to our optimal solution. Figure 3 shows the expected quorum size for different level of clustering taking a fixed f , the probability of a node being alive, and variable node sizes. And figure 4 also shows the same thing but taking a fixed node size, n , and variable values for f .

According to figure 3 we find that the optimal level of clustering, based on the equation 3, for different value of n is almost correct. Again, we see that the high level of clustering is good for large number of nodes while low level of clustering is suitable for small number of nodes. However, the cost reduces much when we use clustering for an n . But further increase in level of clustering does not show such change.

In figure 4 we see that the high level of clustering is good for lower values of f while low level of clustering is suitable for high values of f . However, the cost reduces much when we use clustering for an f less than 1. But if there is no unavailability of any node ($f=1$) to form a quorum, then no clustering gives optimal message cost.

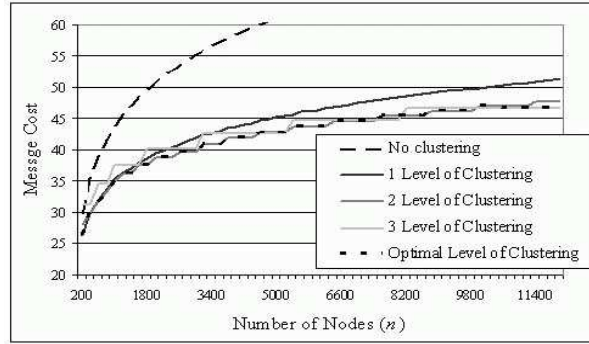


Fig. 3. Comparison between different levels of clustering changing n ($f=0.9$).

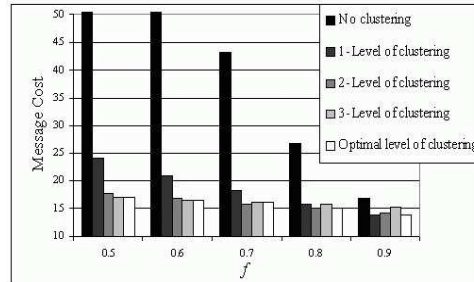


Fig. 4. Comparison between different levels of clustering changing f ($n=1000$).

5 Simulation Result

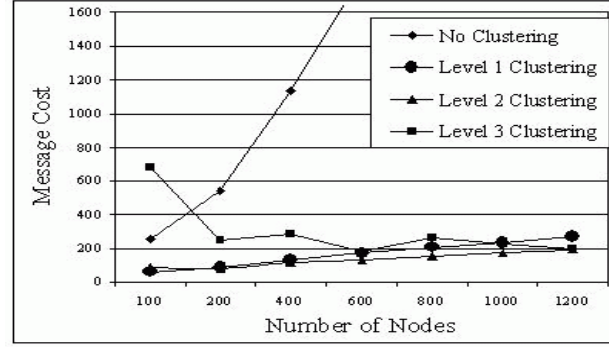


Fig. 5. Impact of various n on Optimal Point of Level of Clustering ($f=0.80$).

We have done the simulation using parsec. Here the result we have found is almost similar to our analytical findings. In figure 5 the impact of n on the level of clustering is shown. From here we find that, if n increases the higher level of clustering performs better.

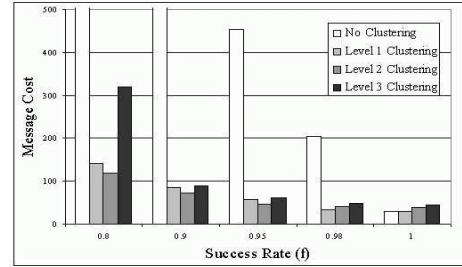


Fig. 6. Impact of different f on Optimal Point of Level of Clustering ($n=400$).

In figure 6 the impact of f is plotted against message cost. It shows that if f is high then lower level of clustering is preferable. However, in these figures we have seen that, the graph for higher level of clustering (such as 3-Level of clustering) performs poor for lower number of nodes at lower value of f although for lower value of f higher value for level of clustering is found to be suitable. This abrupt behavior is happened due to the failure of message routers. Because selecting another message router takes necessary cost.

6 Discussion

Here we have proposed a multi-level clustered architecture for the quorum based distributed solution of mutual exclusion problems. The proposed algorithm is general in this sense that any types of algorithm, centralized or distributed, quorum-based or token-based, can easily be executed in a cluster at any level. We have given some analytical results of the algorithm. As the number of the users and the applications in distributed system is increasing day by day and the growth rate is really high, algorithms using multi-layer network topology will become important very soon. Again algorithms, like this, are suitable for ad-hoc networks too.

References

1. Banerjee, S., K. Chrysanthos, P.: A New Token Passing Distributed Mutual Exclusion Algorithm. *Proceedings of the 16th ICDCS (1996)* 717–724
2. Raymond, K.: A Tree based Algorithm for Distributed Mutual Exclusion. *ACM Transactions on Computer Systems (1989)* Vol. 7 No. 1 61–77
3. Naimi, M., Trehel, M., Arnold, A.: A log (N) distributed mutual exclusion algorithm based on path reversal. *Journal of Parallel and Distributed Computing (1978)* Vol. 34 No. 1 1013–1015
4. Ricart, G., K. Agrawala, A.: An Optimal Algorithm for Mutual Exclusion in Computer Networks *Communications of the ACM (1981)* Vol. 24 No. 1 9–17
5. Maekawa, M.: A N Algorithm for Mutual Exclusion in Decentralized Systems. *ACM Transactions on Computer Systems (1985)* Vol. 3, No. 2 145–159
6. Agarwal, D., El Abbadi, A.: An Efficient and Fault-Tolerant Solution for Distributed Mutual Exclusion. *ACM Transactions on Computer Systems (1991)* Vol. 9 No. 1 1–20
7. C. Saxena, P., Rai, J.: A survey of permission-based distributed mutual exclusion algorithms. *Elsevier Science Publishers B. V. (2003)* Vol. 25 No. 2 159–181
8. E. K. Mamun, Q., Ali, M., M. Masum, S., A. R. Mustafa, M.: A Two-Layer Hybrid Algorithm for Achieving Mutual Exclusion In Distributed Systems. *WSEAS Transactions on Systems (2004)* Vol. 3 No. 3 1193–1198
9. Housni, A., Trelhel, M.: Distributed Mutual Exclusion Token-Permission based by Prioritized Groups. *Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications (2001)* 253–259
10. Erciyes, K., Ali, M., M. Masum, S., A. R. Mustafa, M.: Distributed Mutual Exclusion Algorithms on a Ring of Clusters. *ICCSA, SV-Lecture Notes in Computer Science (2004)*
11. Bertier, M., Arantes, L., Sens, P.: Distributed Mutual Exclusion Algorithms for Grid Applications: a Hierarchical Approach. *Journal of Parallel and Distributed Computing (JPDC), Elsevier (2006)* Vol. 66 128–144
12. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM (1978)* Vol. 21 No. 7 558–564
13. H. Thomas, R.: A majority consensus approach to concurrency control. *ACM Transaction on Database System (1979)* Vol. 4 No. 2 180–209
14. Garcia-Molina, H., Barbara, D.: How to Assign votes in a Distributed System. *Journal of the Association for Computer Machinery (1985)* Vol. 32 No. 4 841–860

- [15] Schlichting, R.D. and Schneider, F.B.: "Fail-stop processors: an approach to designing fault-tolerant computing systems," ACM Trans. on Computing Systems, Vol. 1, No. 3, pp. 222-238, 1983.
15. D. Schlichting, R., B. Schneider, F.: Fail-stop processors: an approach to designing fault-tolerant computing systems. ACM Transactions on Computing Systems (1983) Vol. 1 No. 3 222-238