International Journal of Research and Development in Applied Science and Engineering (IJRDASE)
ISSN: 2454-6844

Advanced Approach for Achieving Mutual Exclusion in Distributed Network

Om Awasthi, RKDF University, Bhopal, Madhya Pradesh, omawasthi.rishi@gmail.com
Dr. Santosh Kumar Shukla, BBDITM, Lucknow, Uttar Pradesh
Dr. Devendra Kumar, BNCET Engineering College Lucknow

Abstract- This paper describes a unique way for ensuring mutual exclusion in distributed systems that makes use of an n-node network. The suggested approach communicates between nodes via message passing. This technique divides the distributed system into smaller sub-systems, each with about √m nodes (m being the smallest perfect square bigger than or equal to n). If n is already a perfect square, no changes are done. The method employs a token-based strategy in which, in the best scenario, a node obtains the token with just two messages, while in the worst case, the token is received after n messages.

Keywords: Computer network, Distributed algorithm, Data forwarding, Message complexity.

1. INTRODUCTION

The solution to the problem of the mutual exclusion in distributed is to implement a protocol that is implemented by processes inside the distributed system and is solely based on message forwarding. This enables one or more processes to access shared resources while performing private activities. Implementing mutual exclusion on shared items in centrally managed systems is quite simple, and is frequently accomplished via semaphores or monitors. However, in a distributed system, the lack of a global or centralized controller considerably complicates the solution because nodes may only interact via message exchanges.
A distributed mutual exclusion method requires a system that notifies all other nodes that a process has already entered the crucial region, prohibiting them from doing so at the same time. There are several approaches available for solving this. There is a risk of congestion with a centralized system since a single administrator is in the position of controlling the whole network; yet, the implementation is straightforward.
In a system distributed with network, complexity of message is large since no node knows if the token is available. As a result, a node desiring to reach the crucial region must send request messages to all nodes in the system to locate the token. Despite the high system load, the suggested method stays well-balanced. The remainder of this article is organized as: Section 2nd covers relevant work, and Section 3rd describes the algorithm's basis, which is following a formal explanation. Finally, Section 4 discusses the conclusions.

2. LITERATURE REVIEW

Distributed mutual exclusion methods [1], [2] use a unique token, also defined by the PRIVILEGE message [2], that is sharing among the sites. Possession of this token authorizes a site for entering and executing its crucial section (CS). The token's sole existence ensures that mutual exclusion is maintained across the distributed system.
The Suzuki-Kasami method, introduced in 1985, requires 0 to N messages for a node for entering the critical section (CS) [2]. A node with the token can access the CS. Because only one node at a time has the privilege, when a node asks access to the CS, it sends a message to all other nodes. If the token is idle at a certain site, that site delivers it to the requesting node. The site holding the token can enter the CS several times until another site do not acquires it.
The message that is in request state is formatted as REQUEST(j,n), indicating that j^{th} site is seeking the n^{th} CS. Each node keeps a size N array as RN to store the most recent sequence number received from each other node. The PRIVILEGE message is formatted as PRIVILEGE (Q, LN), with Q representing a nodes queue seeking access to the CS and LN representing an array of size N. LN[j] denotes the most recent CS processed by j^{th} node. If RN[j] = LN[j]+1, it indicates that j^{th} node issued a request for a new sequence of CSs, and the node with the permission to add this to the queue sends the PRIVILEGE(LN,Q) to the node that requested the CS. The messages number under one CS entrance is (N-1) REQUEST messages + 1 PRIVILEGE message, for a total of N messages, or 0 if the node with the token wishes for entering in respective CS.
Kerry Raymond created an algorithm in 1989 that organizes nodes in an unrooted tree structure [3] and transmits large amount of messages through the undirected edges of tree. Every node is aware of its near neighbours. To reach the CS, a node must first receive the PRIVILEGE message, much like with other token-based algorithms. At each node, a HOLDER named variable

referring to a node on the PRIVILEGE. If a node has the PRIVILEGE, the HOLDER is referring to itself. If a non-privileged node is asking access to the CS, it is creating a request and storing it under the local queue known as REQUESTQ. If it has already not sent a message toward the node indicated by the HOLDER, it prefers of sending message through a path by the token holder. Upon instance of request message receiving, nodes perform forwarding it through a path by token holder, but prior to this, the request is added to local REQUESTQ. On reaching of request to the node that possess PRIVILEGE, and if that node is currently not performing its critical section, it is transmitting the PRIVILEGE to the node that has sent request. When a node receives the PRIVILEGE, it inspects its queue. If its own ID is at the top, it is executing the CS; otherwise, it is sending the PRIVILEGE towards the node with the ID at the head of the queue and modifies respective HOLDER variable to pointing towards that node. The messages size in number necessary to access the CS might range from 0 to 2D, where D representing tree's diameter. However, under full load, this may be reducing to a maximum of 4 messages per CS executing in a correctly constructed tree, or to just two messages if the system is arranged as a chain.

Mukesh Singhal presented an algorithm in 1989 [4], which use state information to describe the collective mutual exclusion processes states throughout of the system. Status updates is tracked by each site for another sites and using that information for determination of subset of sites is most likely to have the token. As a result, the amount of exchanged messages to enter the CS is fluctuating at random value in between 0 and n (n here represents number of system sites). Sites employing sequence numbers for distinguishing in outdated and current requests of token. Every site keeping a counter, & when it has to run its CS, it increases the count and providing the value after modification of number of the sequence in its messages of token request. Each site additionally keeping the track of the largest value of sequence number and the state information that is updates most recently for all other sites. By comparison the number of sequence of a received request to the sender's most recent known sequence number, the token is provided to the site having sequence number with the lowest value.

Sebastian Cantarell et al. (2001) suggest a system with two layers: the application layer (as topmost layer) and the GME layer (as bottom most layer). There are two sorts of messages used to establish the interaction between these layers: Requesting & Granting Session [6]. When the needs to access to a session are rooted by application layer, named as 'Session X', the process sends a Session(X) Request message to the GME layer. The GME layer then authorizes access to Session(X) by returning a Grant-Session message to application layer. These messages have a maximum size of $2 \times \log(m + 1)$ bits. In the worst-case situation, each resource request may create $O(n^2)$ messages, whereas as per best-case scenario, it may need none of the messages.

Quazi Kabir Ehsanul & Nakazato Hidenori (2006) described a unique approach based on token for mutual exclusion in group for systems of distributed networks [5, 6]. This protocol is using a single token for allowing many processes for access under the CS area of a session under shared state. Concurrency is a major protocol component; both throughput and time of wait may be modified by changing the length of a session declaration. The least amount of messages numbers necessary for entering the CS is zero, and the highest is $(n + 2)$, where n is the total number of system processes.

3. PROPOSED APPROACH

3.1 Network Model

The network is believed to be completely linked, with no malfunctioning processors. The network's n nodes are separated into m sets logically, each of which has m nodes. These sets are known as local groups (LGs). The nodes in each LG are completely linked and may interact directly to reach the CS. Single node from every LG is chosen to serve role of local coordinator (LC). The LC under all LGs establishes a new group known as the global group (GG). One node in this GG is chosen to act as the Global Coordinator (GC). All the nodes in the GG has the ability to interact with all other nodes in the GG as well as all nodes in its own LG [7, 8].

3.2 Basis of the Algorithm

The suggested technique uses a token-based approach, with just one token available in the network. When a node obtains a token, it gains access to the CS.

1. Every node in the group is aware of the specified LC in each LG. A request message is sent to the LC by a node in the local group requesting access to the crucial section.
2. The LC awaits the token's availability inside the local group after receiving a token request. The LC obtains the token from the process that is presently holding it inside the group, and if it is accessible locally, it transmits it to the requesting node.
3. The node gives the token back to the local group's LC after finishing its CS operation.
4. The LC sends the request to the nodes in the GG if the token is not accessible locally.
5. By transmitting the token request, each LC in the GG verifies if the token is available inside their own LG. To verify its possession, the token's node replies to its LC.

6. The idle token that is holding by the LC sends it to LC that request for it, & LC that is requesting token then passes it to the requesting node.
7. The node gives the token back to the local group's LC after finishing the CS process.
8. To determine whether there are any pending requests, the LC with the idle token looks through its request queue.
9. The LC sends the token to the appropriate node if it detects a local request. The token is sent to the LC making the request if it comes from another LC.
10. Only the LC of the desired node receives the request if the requirement of a node is wanted by any other node.
11. Proving the equality higher for any homomorphism token f by showing that for elements a, b, c, $g(a,b,c)=g(a).g(b).g(c)$ and $g(a)^{-1}= g(a^{-1})$.
12. Searching the all probable homeomorphisms from F_2 to $Z_3 \times Z_3$ and from Z to Z.

The technique of selecting the locations of a group's generators and then "extending" the homomorphism to the remainder of the group is frequently very helpful, but it is only applicable when the generators' images satisfy every relation between the generators [9, 10].
If Z_3 is generated by R_{120} than for defining a homomorphism $f:Z_3\rightarrow Z$ by letting $f(R_{120})=1$, to send a generator towards another generator. Does it is extending towards homomorphism? Which function does relate R_{120} for satisfying that $1\in Z$? There are many homeomorphisms from F_2 to $Z\times Z$. Take for instance $f(a)=(1,0)$ and $f(b)=(0,2)$.
To find the solution, all the homomorphisms from $Z\times Z$ to F_2 are checked to find that what do they have in common?

```
While (true)
        Do
      {  Selection of group g belongs to G;
         Requesting for g : - Entry protocol.
            Critical Section(CS)
            Releasing : Protocol Exit.
        }
```

Let $f:G\rightarrow H$ is showing a type of homomorphism in both groups, of an identity of G denoting e_G & identity of H denoting by e_H. Showing that $f(e_G)=e_H$, that is, identity is send towards identity using a homomorphism. That is obvious that the application of the fact that e=ee and the defining homomorphism's property [11].
On consideration of a mapping $f: Z_9\rightarrow Z_3$ described under $f(R_m)=R_{3m}$ (recall that if R_m is under rotation of direction in counterclockwise of degree m). Is this a homomorphism? Find the homomorphism from Z_6 to Z_3.
Is the mapping $f:Z_6\rightarrow Z_5$ given that $f(R_m)=R_0$ (the identity) a homomorphism? Find the homomorphism of F_2 to $Z\times Z$

4. CONCLUSION

The paper introduces a mutual exclusion scheme that uses a token-based algorithm and is useful in distributed systems applications. Performance improvement is demonstrated in comparison to existing mutual exclusion methods that use a token-based algorithm. The proposed approach successfully reduced message complexity to √n in the worst case and as low as 2 in the best case.

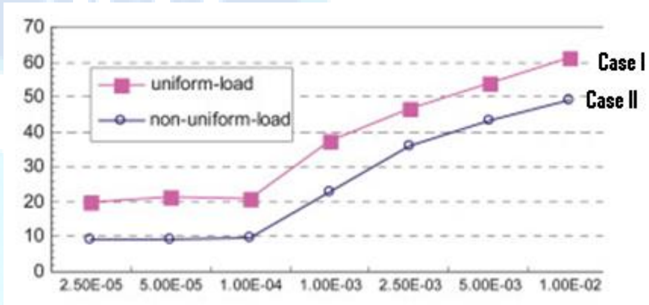

Figure 1: Comparison of complexity of Message of $n^{1/2}$ in between two case

5. REFERENCES

[1] Mohd Naimi, Michel Trehel, Andre Arnold. A log(n) distributed mutual exclusion algorithm based on path reversal. J. Parallel Distrib. Comput., 34(1):1-13, 1996.
[2] Ichiro Suzuki and Tadao Kasami. A distributed mutual exclusion algorithm. ACM Trans. Comput. Syst., 3(4):344-349, 1985.
[3] Kerry Raymond. A tree-based algorithm for distributed mutual exclusion. ACM Trans. Comput. Syst., 7(1):61-77, 1989.
[4] Mukesh Singhal. A heuristically-aided algorithm for mutual exclusion in distributed systems. IEEE Trans. Comput., 38(5):651-662, 1989.
[5] Quazi Ehsanul Kabir Mamun and Hidenori Nakazato. A new token based protocol for group mutual exclusion in distributed systems. In ISPDC, pages 34-41, 2006.
[6] Sebastien Cantarell, Ajoy Kumar Datta, Franck Petit, and Vincent Villain. Token based group mutual exclusion for asynchronous rings. In ICDCS, pages 691-694, 2001.
[7] Kumar, D., & Ahamad, F. (2023, December). Opinion extraction from big social data using machine learning techniques: A survey. In AIP Conference Proceedings (Vol. 2916, No. 1). AIP Publishing.
[8] Kumar, D., & Ahamad, F. (2024). Opinion Extraction using Hybrid Learning Algorithm with Feature Set Optimization Approach. Journal of Electrical Systems, 20(3), 1922-1932.
[9] Kumar, D., & Ahamad, F. (2024). Application of Machine Learning Algorithm for Optimal Model Design for Opinion Extraction. Telematique, 23(01), 215-227.
[10] Srivastava, A., & Banoudha, A. Techniques of Visualization of Web Navigation System.

[11] Sahay, S., Banoudha, A., & Sharma, R. (2013). Comparative Study of Soft Computing Techniques for Ground Water Level Forecasting in a Hard Rock Area. International Journal of Research and Development in Applied Science and Engineering, 4(1).