ANALYSIS OF RAYMOND TREE TOKEN BASED DISTRIBUTED MUTUAL EXCLUSION ALGORITHM

Lakshmi Nannapaneni
Computer Science Department
Kent State University, Ohio
e-mail: lnannapa@kent.edu

***Abstract*—Raymond Tree Algorithm is a token based Distributed Mutual exclusion Algorithm in which a process (a node) in a distributed system can enter the critical section only if it is in the possession of a token. The token is obtained by a node using message passing mechanism. The experiment results shows that the Raymond's Tree Algorithm requires O (Log N) message under low load and reduced number of messages exchanged per critical section to approximately 4 messages under high load. In this experiment Raymond Tree algorithm is applied on Star, Chain and an arbitrary Tree topology**

***Keywords*---** Critical Section, PRIVILEGE, REQUEST, Star, Chain, Tree

I. INTRODUCTION

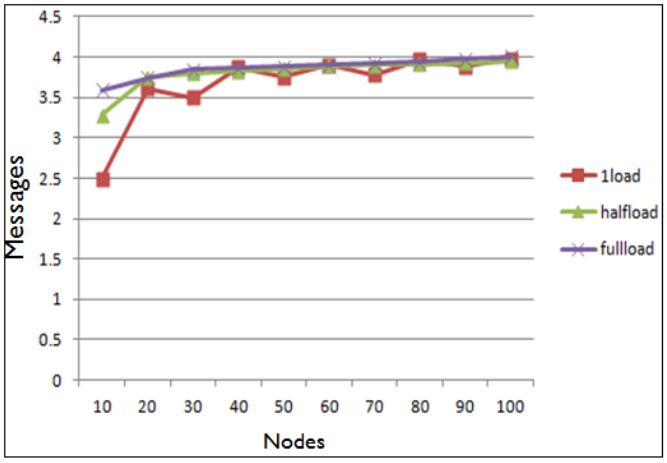
Raymond Tree Algorithm uses a spanning tree to reduce the number of messages exchanged per critical section execution. The network is viewed as a graph; a spanning tree of a network is a tree that contains all the N nodes. The algorithm assumes that the underlying network guarantees message delivery. All nodes of the network are 'completely reliable. Sites are logically arranged as a directed tree. Edges of the tree are assigned directions toward the token holder (root of the tree) Root of the tree is the node with the token. Messages between nodes traverse along the directed edges of the tree. A node needs to hold information about and communicate only to its immediate-neighboring nodes. Similar to the concept of tokens used in token-based algorithms, this algorithm uses a concept of privilege. Only one node can be in possession of the privilege (called the privileged node) at any time, except when the privilege is in transit from one node to another in the form of a PRIVILEGE message. When there are no nodes requesting for the privilege, it remains in possession of the node that last used it. Each node maintains a HOLDER variable that provides information about the placement of the privilege in relation to the node itself. A node stores in its HOLDER variable the identity of a node that it thinks has the privilege or leads to the node having the privilege. Each node maintains a request Queue. Each node requesting for entry into the critical section sends REQUEST message to its HOLDER and the HOLDER in turn forwards the REQUEST to its HOLDER and this continues until the REQUEST reaches the actual holder of the token. The token holder sends the PRIVILEGE message to the requesting node at the head of the queue. On receiving the PRIVILEGE if the nodes own id is top of the queue, it executes critical section else sends the PRIVILEGE to the node pointed by the id, and set its holder to point to that node and if there are any requesting nodes in its queue sends the REQUEST message behind the PRIVILEGE message. The number of messages required to execute critical section can be 0 or typically 2D, where D is the diameter of the tree on which the algorithm is running. At high load, approximately 4 messages are required for each node to gain entry into the critical section. At low loads, number of messages required depends on the topology of the network. For each REQUEST message, there will be a corresponding PRIVILEGE message. Hence the number of PRIVILEGE messages will be equal to number of REQUEST messages.

II. EXPERIMENTAL SETUP

Experimental setup in the process of Analysis, data was collected by executing the algorithm for 10 times and taking the average of total number of messages which were sent for each critical section execution. Messages considered were of two categories - REQUEST and PRIVILEGE. Results were taken by varying critical section execution for 1load, half load, and full load on different number of nodes from 10 to 100 in the increments of 10 by fixing the topologies Star, Tree, Chain.

III. RESULTS FOR RAYMOND TREE AGORITHM

A. *Raymond Tree Algorithm for Star Topology*

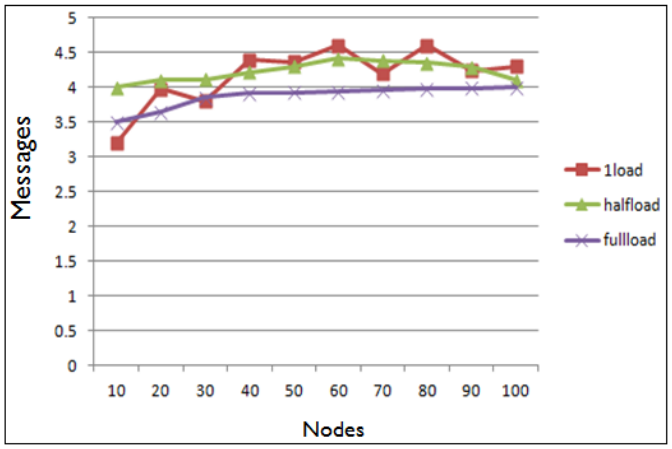


Raymond Tree- Number of Messages per CS entry in Star topology

The above graph indicates that at full load, the algorithm requires exchange of only four messages per CS execution.

B. *Raymond Tree Algorithm for Tree Topology*

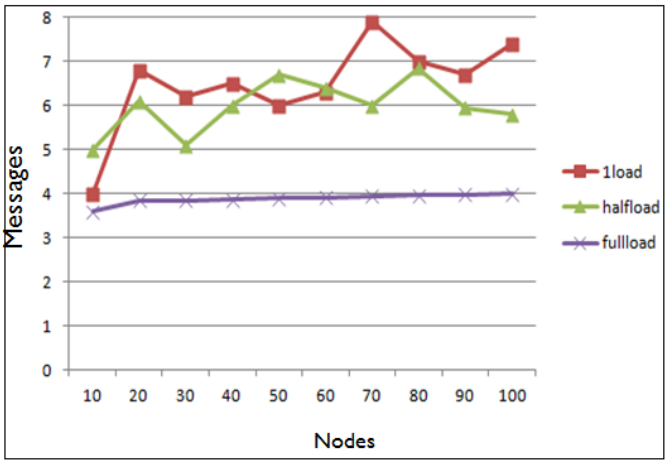
The Experiment was conducted on an arbitrary topology with random number of depths and random number of children for each node is considered.



Raymond Tree- Number of Messages per CS entry in Tree topology

The above graph, at full load, the algorithm requires exchange of only four messages per CS execution. For 1load and half load, more number of messages is required when compared to star topology. Because there would be many depths in a tree topology as compared to only 2 in star topology. Hence, more the number of depths, at low loads higher number of messages are required.

C. *Raymond Tree Algorithm for Chain Topology*



Raymond Tree- Number of Messages per CS entry in Chain topology

The above graph indicates that at full load, the algorithm requires exchange of only four messages per CS execution. But at the 1load and half load the number of messages required is more when we compared with the tree and star topology. Because in chain topology the nodes are arranged in a straight line as a chain. If suppose the requesting node is at one end of chain and the token holder is at the other end, then there will be N-1 REQUEST and N-1 PRIVILEGE messages for one critical section entry. Hence maximum number of messages per critical section entry can be 2(N-1) messages. So more number of messages are required if the request has to travel from one end to the other.

IV. CONCLUSION

An experiment was done on Raymond Tree Algorithm by varying parameters for fixed topologies. The result obtained signifies us that the worst case occurs at low loads and in chain topology which increases the diameter of the network used. Under heavy load, the algorithm exhibits an interesting property: "As the number of nodes requesting for the privilege increases, the number of messages exchanged per critical section entry decreases."

V. FUTURE WORK

Future work includes testing the performance of the algorithm on chain of large number of nodes in thousands range to get the accurate result. The implementation can be enhanced to make Raymond Tree work for completely connected graphs in which a spanning tree can be formed and algorithm can be applied on that.

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to Dr. Mikhail Nesterenko. In this course of Advanced Operating System, Dr. Mikhail Nesterenko has offered me lots of help and hint to me to carry the tasks to accomplish the project works.

REFERENCES

[1]Raymond, Kerry "A Tree-Based Algorithm for Distributed Mutual Exclusion", *ADM Transactions on Computer system, Vol. 7, No.1,* February 1989, Pages 61-77.

[2]R. Satyanarayanan and D. R. Muthukrishnan. A Note on Raymond's Tree Based Algorithm for Distributed Mutual Exclusion. *Information Processing Letters*, 43(5):249–255, 1992.