Mutual Exclusion and Its Algorithms

Jaya Jain
Student, Dept. of Computer Applications, National Institute of Technology, Kurukshetra , (India)

Neha Mishra
Student, Dept. of Computer Applications, National Institute of Technology, Kurukshetra , (India)

Dr.Bharti Sharma
Assistant professor, National Institute of Technology, Kurukshetra, (India)

Abstract – A distributed system is a gathering of self-regulating node appears to its user as a single coherent system. Resource division is the chief advantage of distributed computing. Though, a distributed computing system may perhaps have some physical or virtual resource that may possibly be reachable by an only process at a time. The mutual exclusion concern is to ensure that no further process at a time is permissible to access some collective resource .In this paper the various protocols will be discussed for mutual exclusion process.

Index Terms – Mutual Exclusion, Distributed System, Protocols.

1. INTRODUCTION

Generally in every computer system there are resources that are shared among two or numerous tasks. That means these resources need to be protected from having more than one task using them at the same time. This protection mechanism is called mutual exclusion .In other words mutual exclusion is a property of concurrency control, which is introduce for the purpose of preventing race conditions; it is the requirement that one thread of execution never enter its critical section at the same time that another concurrent thread of execution enters its own critical section. Resource sharing and cost effective are the major advantage of distributed computing system.

2. RELATED WORK

- Types of mutual exclusion algorithms:-

  Centralized Algorithm

  Distributed Algorithm

Centralized Algorithm :-As its name implies, there is single coordinator which handles all the requests to access the shared data. Every process takes permission to the coordinator, if coordinator agree and will give permission then a particular process can enter into CS. Coordinator will maintain a queue and keep all the requests in order.

Benefits

- It is Fair algorithm; it follow FIFO algorithm concept for giving permission.
- It is very simple algorithm in context of implementation.
- We can use this scheme for general resource allocation.

Shortcomings

- No multiple point failures, No fault tolerant.
- Uncertainty between No-reply and permission denied.
- Performance bottleneck because of single coordinator in a large system.

Distributed algorithm [2]:-In distributed algorithm , there is no coordinator. For permission to enter into CS, Every process communicate to other process. These algorithms are sectioned into two parts -

- Non-Token based algorithms
- Token based algorithms

Contention (Non-Token) based algorithms [2,4,9]:-In this algorithm, process converse with a group of other process to select who should execute the critical section first. These algorithms also divided into 2 parts:

1. Timestamp based
2. Voting scheme

Two basic Timestamp-based Contention algorithms are:

LAMPORT'S ALGORITHM :Lamport designed a distributed MUTEX algorithm on the basis of his concept of logical clock . This is a algorithm which is a non-token based scheme. Non-token based protocols use timestamps to order requests for CS. Request message Contain following:

- Identifier (Machine id, process id)
- Name of resource
- Timestamp

Timestamp is a distinctive integer value which is given by the operating system to the process when they produce requests for CS. Timestamp is rapidly increased every time when a request

is reached. Lesser timestamp requests have higher precedence rather than big timestamps requests. In lamport's algorithm, for a process Pi, request set Ri= {P1, P2, P3…….Pn}. It contains of all those process from which Pi must require permission before entering the CS. Every process maintains a queue of awaiting requests for entering CS in the ascending order of timestamps. This algorithm assumes that channels which are using follows FIFO method.

3. ALGORITHM

Requesting the critical section:

When a process wants to enter into CS, it follows the following steps:

1. Enters its request in its own queue (ordered by time stamps).

2. Sends a request to all other nodes.

3. Wait for replies from all other nodes.

4. When another process receives this request message, it sends a timestamp reply message to the requesting process and keeps this request in its own request queue.

Executing the critical section:

A process can enter into CS when these two conditions are satisfied:

- Pi has not received a message with timestamp larger than its from all other process.

Pi's request is at the top of request_ queue.

Releasing the critical section:

1. For exiting the critical section, it removes its request from the queue and sends a release message to every process.

2. Upon receiving release message form the process, then other process removes the related entry from its own request queue.

3. If own request is at the top of its queue and all replies have been received, enter into critical section.

Performance

 Message complexity:

(N-1) number of messages required for requesting CS. (N-1) number of messages required for reply. (N-1) number of messages necessary for release. Total 3 (N-1) numbers of messages required in heavy load as well as in case of lightly loaded.

 Synchronization delay:

Average message delay for sending a message from one process to another process. T time is required in synchronization.

RICART-AGRAWALA ALGORITHM:

Ricart-agrawala algorithm is an expansion and optimization of Lamport's protocol. This algorithm is also for MUTEX and it is a non-token based algorithm. This algorithm combines the RELEASE and REPLY message of lamport's algorithm and decreases the complexity of the algorithm by (N-1). In this algorithm, there is a request set Pi= (P1, P2, P3……Pn). It comprises of all the sites from which a site needs to acquire authorization before entering to CS. The Algorithm proceeds as follows.

Requesting the critical section

1. When a site desires to execute into CS, it sends a request along with its timestamp to all sites. This message includes the site's name, and the current timestamp of the system according to its logical clock.

2. Upon reception of a request message, another site will immediately sends a time stamped reply message if and only if:

 The receiving process is not currently interested in the critical section.

 The receiving process desires to enter into CS but its own timestamp value is higher than requesting site.

 Otherwise, the receiving process will suspend the reply message. This means that a reply will be sent only after the receiving process has completed using the CS itself.

Executing the critical section

 Requesting site enters its CS only after receiving all reply messages.

Releasing the critical section

1. Upon exiting the critical section, the site sends all deferred reply messages.

2. In this algorithm, all the CS requests are executed in their timestamp order.

Performance

 Message complexity:

(N-1) number of messages required for requesting CS. (N-1) number of reply messages merges with release. Total 2 (N-1) numbers of messages required in heavy load as well as in case of lightly loaded.

 Synchronization delay:

Average message delay for sending a message from one process to another process. T time takes place in synchronization.

 Reply messages are combined with release messages because reply messages are send to only those sites whose timestamp is greater than executing site.

Disadvantage:

Failure of a node – May result in starvation.

Controlled (TOKEN) BASED ALGORITHMS :-

Token-based algorithms are the algorithm in which a site is allowed to enter its CS if it possesses the token. This token is unique among the processes . Instead of timestamps Token-based algorithms use sequence numbers to differentiate between old and existing requests. Usually FIFO message delivery is not adopted. Also their Proof of correctness is trivial.

Problems: The question is how to get a token. This makes distinction between different algorithms. On the basis of structure in which process are connected, these algorithms are divided into 3 parts.

Ring Structure : According to this structure all processes are allotted and are linked in the form of a ring . The positions of ring may be given in numerical sequence of network addresses . The important thing is that each process knows who is next in line after itself. The way of ordering is not that important.

Advantages:

Simple, deadlock-free, fair.

Disadvantages:

 Even inthe absence of any request (unnecessary traffic), the token gets distributed

 Long path (O(N)) – the wait for token may be high.

Broadcast Structure (Suzuki-Kasami Algorithm) :

According to Suzuki-kasami algorithm, if a process do not possess the token and still wants to enter the CS , the process sends a REQUEST message for the token to all other processs. A process which possesses the token sends it to the requesting site upon the receipt of its REQUEST message. If a process receives a REQUEST message when it is executing the CS, it sends the token only after it has completed the execution of its CS.

This algorithm must efficiently address the following two design issues:

(1) How to distinguish an outdated REQUEST message from a current REQUEST message: A site may receive a token request message after the corresponding request has been satisfied and this is because of variable delays in messages. If the request corresponding to a token request has been satisfied site cannot determined, it might dispatch the token to a process that does not need it .this will not disturb the accuracy, nevertheless, this may seriously reduce the performance.

(2) How to determine which site has an outstanding request for the CS:, The process must determine how many sites have an excellent request for the CS after a process has finished the execution of the CS so that the token can be dispatched to one of them. The first issue is addressed in the following manner: A REQUEST message of site Pj has the form REQUEST (j, n) where n (n=1, 2 ...) is a sequence number which indicates that site Pj is requesting its nth CS execution. A site Pi keeps an array of integers RNi[1..N].where RNi[j] denotes the largest sequence number received in a REQUEST message so far received from site Pj .When site Pi receives a REQUEST(j, n) message, it sets RNi[j]:=max(RNi[j], n).When a site Pi receives a REQUEST(j, n) message, the request is outdated if RNi[j]>n. The second issue is addressed in the following manner: The token consists of a queue of requesting sites Q, and an array of integers LN [1...N]; where LN[j] is the sequence number of the request which site Pj executed most recently. After executing its CS, a site Pi updates LN[i]:=RNi[i] to indicate that its request corresponding to sequence number RNi[i] has been executed .At site Pi if RNi[j]=LN[j]+1, then site Pj is currently requesting token.

Requesting the CS:

If the requesting site Pi does not have the token, it increments its sequence number RNi[i], and sends a REQUEST (i,sn) message to all other sites.

– When Pj receives the message, it sets RNj[i] to max (RNj[i], sn). If Pj has the idle token, it sends the token to Pi if RNj[i] =LN[i] +1.

Executing the CS:

Enter CS when token is found.

Releasing the CS:

When the execution of the CS is completed, process Pi takes the following activities: – Sets LN[i] to Rni[i]. – For each process Pj whose ID is absent in the token queue, it attaches its ID to the token queue if RNi[j] = LN[j] +1. After the above update if the token queue is nonempty, it vanishes the top process ID from the queue and sends the token to the process specified by the ID.

Performance:

 Message complexity :

Needs 0 messages if the requesting process holds the token not in use. N messages otherwise (N-1 broadcast and 1 to send the token).

 Synchronization delay:

0 or T based on if process grasp the token at the time of request.

 No Starvation:

Token request messages spread all other processes in limited time. Meanwhile one of these processes possess the token, the request will be placed to the token Q in finite time. As there are at maximum of N-1 more requests in front of this request, the request.

4. CONCLUSION

Requests for access to the critical section are satisfied in the order of their timestamps ,in Non-Token based approach , therefore fairness is certain. In their communications in comparison with the token-based algorithms More no. of messages require for Non-token based algorithms. No algorithm is perfect as everyone has their own pros and cons. No token is required to enter into CS as Non-Token based algorithms are called permission based algorithms . Multiple successive rounds of messages are exchanged between the processes to determine which process will enter the CS next. All nodes of the network are completely reliable. The algorithm delivers the resulting certainties: Mutual exclusion is guaranteed, Deadlock is not possible, Starvation is also impossible.

REFERENCES

[1] Rahul Garg, Vijay K Garg, Yogish sabharwal "Scalable algorithms for global snapshots in distributed systems " ACM 2006.
[2] "Distributed Mutual exclusion" ppt. by Rajnitha Shivarudraiah
[3] "Several-tokens Distributed Mutual Exclusion algorithm in a logical ring network" by Ousmane.
[4] M. Singhal and N. Shivaratri, Advanced Concepts in Operating Systems, New York, McGraw Hill, 1994.
[5] Randy Chow,Theodore Johnson "Distributed Operating system and algorithm analysis"
[6] A. Tanenbaum and M. Van Steen, Distributed Systems: Principles and Paradigms, Upper Saddle River, NJ, Prentice-Hall, 2003.
[7] Abhishek swaroop, Awadesh kumar singh "a STUDY BASED ALGORITHMS FOR Distributed mutual exclusion".
[8] K.Raymond,"A distributed algorithm for multiple entries to a critical section",Information processing letter, vol.30, pp.lg9- 193,1989.
[9] D.agarwal,A.El Abbadi,"A token baesd fault tolerant Distributed mutual exclusion algorithm,journal of parallel and distributed computing,24,pp.164-176,1995.
[10] I.Suzuki and T.Kasami, "A distributed Mutual exclusion algorithm". ACM Transaction on computer systems. Vol.3,no. 4,pp. 344-349,1985.