Journal of Advanced Computing Research Vol. 5, Issue 1 (2023) 9-11

Parallel Computing: A Case Study on Mutual Exclusion

Hajar Hanisah Ayob #1, Mohamed Faidz Mohamed Said #2

Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA
70300 Seremban, Negeri Sembilan, MALAYSIA

[1] hajar.hanisah@gmail.com

[2] mohdfaidz@uitm.edu.my

***Abstract*—Through this paper, the discussion is about the certain type of the approach used and the solution to the problem occurred that different mutual exclusion algorithm can do. One thread of execution never enters its critical section (CS) at the same time that another concurrent thread of execution enters its own critical section. Mutual exclusion described a process for concurrent access to a shared resource or data with the mutually exclusive execution. Only one process is allowed to execute critical section (CS) anytime. Distributed mutual exclusion algorithms are deals with unpredictable delayed message and incomplete knowledge of the system. These are the basic approaches for distributed mutual exclusion such as Token-Based approach and Non-Token-Based approach. In this paper, a discussion of different approaches will be described further with the distinct algorithms that were used from the research that have been made throughout this paper. Based on the algorithm used that differed with one another, difference results and performances of each method proposed will be shown.**

***Keywords*: distributed mutual exclusion; critical section**

I. INTRODUCTION

Mutual Exclusion (ME) is a fundamental problem of a distributed system where a program object wants to avoid simultaneous access to a shared resource. This concept is used in concurrent programming with a critical section, a piece of code in which processes or threads access a shared resource. Back in 1965, the requirement of mutual exclusion was first identified and solved by *Edsger W. Dijkstra* in his paper titled *Solution of a problem in concurrent programming control,* which the first topic in the study of concurrent algorithms have been credited.

In Distributing System (DS), compilations of different processes were involved which the separation of space occurred interconnect to one another by switching messages to provide proficient and conducive surroundings for sharing resources. The algorithms designed are called Distributed Mutual Exclusion algorithms. The problems of DME need to be solved to prohibit race condition and avoid generating false results. This paper discussed some problems occurred in mutual exclusion in distributing system. The main issues is when there is one thread of execution never entered its critical section (CS) at the same time that another concurrent thread of execution enters its own critical section.

II. LITERATURE REVIEW

This section will be discussed further on the methods and techniques to solve the problem in distributing mutual exclusion with distinct algorithm. There are two different approaches which are Token-Based approach and Non-token-based approach. For token-based approach, a unique token was shared among the sites where the site is allowed to enter its critical section (CS) if it holds the token. Thus, mutual exclusion is ensured because the token is unique. Meanwhile, non-token based approach describes as two or more successive rounds of messages are exchanged among the sites to determine which site will enter the next CS.

Mutual exclusion problem is one of the important problems in concurrent programming. One of algorithms in token based approach that being proposed to solve the problem are called Peterson's algorithm as it has been widely studied for its elegance and simplicity that useful for ME. Alagarsamy [1] proposed a simple modification of this algorithm that optimized the bound on the number of possible bypasses to optimal n-1. A study of Peterson's algorithm has been conducted before came out with another improvised algorithm to be used in solving. The algorithm assures improved fairness by optimally bounding the bypasses number.

Process i:

for $j = 1$ **to** $n-1$ **do**
$Q[i] := j;$
$TURN[j] := i;$
wait until $(TURN[j] \neq i) \vee (\forall k \neq i, Q[k] < j)$
enddo;
CS;
$Q[i] := 0;$

Fig. 1. Peterson's algorithm

Token-based approach includes a token-based fully distributed algorithm where many nodes composed in a computer network that communicates by message exchanges. Neamatollahi and Taheri [2] proposed improvised token-based algorithm to solve DME problems using logic structure in the form of two-dimensional (2D) array which imposed in the interconnecting network. While, for a group of ME problems, most algorithm have been proposed treat all groups equally if a process of making request for CS, a group being selects uniformly. Thus, Mittal and Mohan [3] proposed an efficient algorithm that is derived from Suzuki and Kasami's token-based algorithm to solve this kind of problem when there are non-uniformly distributed group selections happen and the results obtained indicates that the algorithm has a better performances than any existing group ME algorithm.

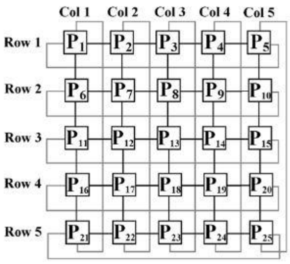

Fig. 2. A proposed logical topology of 25 nodes

Meanwhile, other ME problem is about the real-time applications ones or those where priority is related to a quality of requirement of service. This priority based strategies could be the causes for starvation problems as the low priority ones need to prevent by the high priority requests. Therefore, Lejeune and Arantes [4] solved it by added some heuristics in Kanrar-Chaki priority-based token-oriented algorithm to reduce the priority violations quantity without starvation thus can delay the frequency of priority pending requests. This is another token-based approach used by proposing an extension algorithm from the original algorithm using tree labelling heuristics for the effectiveness performances for the starvation-free priority-based ME algorithm. Furthermore, Grid application in distributed ME problem using hierarchical approach because majority of current ME algorithm are not appropriate for that application. Thus, Bertier and Arantes [5] proposed two ME algorithm based on Naimi-Trehel token-based algorithm that involved hierarchical network topology of Grids using logical tree structure.

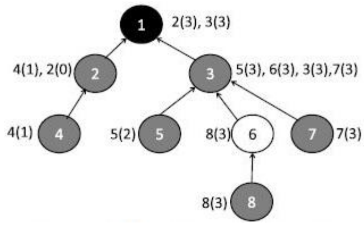

Fig. 3. Kanrar-Chaki algorithm

Moreover, token-based approach is also implemented for a mobile ad hoc network circulation problem which moving randomly depends on their positions and transmission ranges. Chen and Welch [6] proposed a self-stabilizing ME algorithm for mobile ad hoc networks using circulating tokens to solve the constraints on the reliability of delivered messages on mobility. Meanwhile, for a self-stabilizing protocol or *superstabilizing* protocol in ME problem in unidirectional rings were recovered from any legitimate configuration while satisfying some safety property during recovery. Katayama and Masuzawa [7] stated that superstabilizing protocol is attractive with small amount of processes when experience transient fault simultaneously that considered as potential optimization of protocols on unidirectional rings. Moreover, every site communicates with a few sites before entering the critical section. Fu, et al. [8] studied evaluated a modified Raymond algorithm which performed better than the Raymond algorithm as the algorithm behaviour is too complex and hardly analysed mathematically. Thus, the best algorithm of distributed memory system (DMS) for ME depends on those requests produced by the sites. For other distributed ME problems that involved an anonymous unidirectional rings, the algorithm called self-stabilizing fair ME algorithm was presented by Kakugawa and Yamashita [9] under unfair distributed daemon system. This self-stabilizing system was from an arbitrary system state where legitimate state being always resumes continuously. Therefore, it tolerates any finite amount of transient faults.

```
enqueue(Req_Q, Me);
if (HOLDER == Me) {
   HOLDER = dequeue(Req_Q);
   if (HOLDER == Me) {
      enter critical section;
   }
   else
      send TOKN msg is marked to HOLDER;
}
else if (no REQT msg or marked TOKN msg has been issued)
   send a REQT msg to HOLDER;
```

(a) Request critical section.

```
enqueue(Req_Q, Y);
if (HOLDER == Me) {
   if (not in critical section) {
      HOLDER = dequeue(Req_Q);
      if (Req_Q is not empty)
         send TOKN msg is marked to HOLDER;
      else
         send a TOKN msg to HOLDER;
   }
}
else if (no REQT msg or marked TOKN msg has been issued)
   send a REQT msg to HOLDER;
```

(b) Receive a REQT msg from neighbor Y.

```
if (Req_Q is not empty) {
   HOLDER = dequeue(Req_Q);
   if (Req_Q is not empty)
      send TOKN msg is marked to HOLDER;
   else
      send a TOKN msg to HOLDER;
}
```

(c) Exit critical section.

```
HOLDER = Me;
if (TOKN msg is marked) enqueue(Req_Q, Y);
if (Req_Q is not empty) {
   HOLDER = dequeue(Req_Q);
   if (HOLDER == Me) {
      enter critical section;
   }
   else {
      if (Req_Q is not empty)
         send TOKN msg is marked to HOLDER;
      else
         send a TOKN msg to HOLDER;
   }
}
```

(d) Receive a (marked) TOKN msg from neighbor Y

Fig. 4. Modified Raymond algorithm

ME algorithm of non-token-based approach using First-Come-First-Serve (FCFS) property and FCFS algorithm that use only read-write operations. The practical significance of concurrent algorithm increasing thus raises the interest to its correctness where the algorithm unexpectedly misbehaves due to bugs or race condition. Hesselink [10] proposed a new method, MX algorithm with FCFS property for complexity that proves its output is less than a quadratic amount of threads. Next, simple and attractive properties of Lamport's Bakery algorithm had included FCFS but required bounded shared space. Aravind [11] addressed two basic questions of FCFS-ME algorithm that use only read-write operations in terms of lower bound shared space with fairness. The simple algorithms used are by atomic, nonatomic and highly-fair nonatomic algorithm.

III. METHODOLOGY

The study of ME algorithm is presented by identifying the priority relation by the algorithm construction itself. Moses and Patkin [12] provided an analysis that insight more into working on Bakery algorithm that achieved by proving correctness. The paper applied Lamport's Bakery algorithm using single-writer, multi-reader (SWMR) model safe register and provide an improved algorithm called Boulangerie algorithm. Two ways that proves the Bakery algorithm have unnecessary and potentially costly blocking were identified as the improvised Boulangerie algorithm are fixes the inefficiencies and prove not having unnecessary blocking. Another ME problems that involved an asynchronous system of passing message using a non-token-based approach are addressed by Delporte-Gallet [13] considered the fault tolerant ME problem in a distributed message-passing system as reliable channel and processes can be crashed. This FTME problem is about a single, indivisible, resource among n processes allocation. An algorithm is implemented as inspired by Lamport's Bakery algorithm to solve the problem.

```
Initialize: number[i] = 0;  choosing[i] = false;
while true do
    noncritical section;
    choosing[i] ← true;
    number[i] ← 1 + max{number[1], ..., number[N]};
    choosing[i] ← false;
    forall the j ≤ N s.t. j ≠ i do
        await choosing[j] = false ;
        await number[j] = 0 ∨ ⟨number[i], i⟩ <_L ⟨number[j], j⟩
    critical section;
    number[i] ← 0;
```

Fig. 5. Bakery algorithm

In a mutual exclusion problem, the other approach is a non-token based approach in distributed ME algorithm which considered as permission-based algorithm and quorum-based algorithm. Nesterenko and Mizuno [14] presented a self-stabilizing quorum based distributed ME algorithm based on Maekawa's algorithm that designed for an asynchronous message-passing model. The two main performance metrics MX algorithm is created to optimize the message complexity and synchronization delay where the algorithm well-scaled as it proportional to the square root of the system processes number.

IV. Conclusion

In this paper, those methods or approach that are being discussed have different techniques and model that being implemented to solve many problems involved in distributed system of mutual exclusion. Some of them are used for different issues and some are for same problems. From all the research that being made there will be a suitable approach that can obtain a better performance to the problem occurred. Token-based approaches have more benefits for the constraints in any ME problems while the non-token approach seems like having lack in terms of algorithm that being built. Therefore, token-based approaches are more likely recommended to build a much better performance for any mutual exclusion problem out there.

References

[1] K. Alagarsamy, "A Mutual Exclusion Algorithm with Optimally Bounded Bypasses," *Information Processing Letters*, vol. 96, pp. 36-40, 2005.

[2] P. Neamatollahi and H. Taheri, "Info-Based Approach in Distributed Mutual Exclusion Algorithms," *J. Parallel Distrib. Comput.*, vol. 72, pp. 650-665, 2012.

[3] N. Mittal and P. K. Mohan, "A Priority-Based Distributed Group Mutual Exclusion Algorithm When Group Access is Non-Uniform," *J. Parallel Distrib. Comput.*, vol. 67, pp. 797–815, 2007.

[4] J. Lejeune and L. Arantes, "A Fair Starvation-Free Prioritized Mutual Exclusion Algorithm for Distributed System," *J. Parallel Distrib. Comput.*, vol. 83, pp. 11-29, 2015.

[5] M. Bertier and L. Arantes, "Distributed Mutual Exclusion Algorithms for Grid Applications," *J. Parallel Distrib. Comput.*, vol. 66, pp. 128-144, 2006.

[6] Y. Chen and J. L. Welch, "Self-Stabilizing Dynamic Mutual Exclusion for Mobile Ad Hoc Networks," *J. Parallel Distrib. Comput.*, vol. 65, pp. 1072–1089, 2005.

[7] Y. Katayama and T. Masuzawa, "A Latency Optimal Superstabilizing Mutual Exclusion Protocol," *Journal of Parallel and Distributed Computing*, vol. 62, pp. 865–884, 2002.

[8] S. S. Fu, N.-F. Tzeng, and J.-Y. Chung, "Empirical Evaluation of Mutual Exclusion Algorithms for Distributed Systems," *Journal of Parallel and Distributed Computing*, vol. 60, pp. 785-806, 2000.

[9] H. Kakugawa and M. Yamashita, "Uniform and Self-Stabilizing Fair Mutual Exclusion on Unidirectional Rings under Unfair Distributed Daemon," *Journal of Parallel and Distributed Computing*, vol. 62, pp. 885–898, 2002.

[10] W. H. Hesselink, "Mutual Exclusion by Four Shared Bits with Not More Than Quadratic Complexity," *Science of Computer Programming*, vol. 102, pp. 57-75, 2015.

[11] A. A. Aravind, "Simple, Space-Efficient, and Fairness Improved FCFS Mutual Exclusion Algorithms," *J. Parallel Distrib. Comput.*, vol. 73, pp. 1029–1038, 2013.

[12] Y. Moses and K. Patkin, "Mutual Exclusion As a Matter of Priority," *Theoretical Computer Science*, pp. 1-15, 2016.

[13] C. Delporte-Gallet, "Mutual Exclusion in Asynchronous Systems with Failure Detectors," *J. Parallel Distrib. Comput.*, vol. 65, pp. 492–505, 2005.

[14] M. Nesterenko and M. Mizuno, "A Quorum-Based Self-Stabilizing Distributed Mutual Exclusion Algorithm," *Journal of Parallel and Distributed Computing*, vol. 62, pp. 284-305, 2002.

[15] H. H. Ayob. (2017). Parallel Computing – A Case Study on Mutual Exclusion. Available: https://www.youtube.com/watch?v=LuFJFa4FD_U&rel=0 [Accessed: 28-Nov-2017].