

# Preface

The broad area of research presented in this book is designing operating systems (OS) for distributed systems. The advantages of a distributed system over a traditional time-sharing mainframe system depend a lot on the underlying operating system. The expected benefits of a distributed computing platform include distribution transparency, improved price/performance ratio, better system response through load distributing, higher dependability, etc. This research area has always been fascinating to explore. Hence, operating systems, as a whole, forms the broad domain of the research presented in this book while the book focuses on process synchronization for a distributed system.

Existing mutual exclusion (ME) algorithms are often either symmetric or token-based. Token-based approaches offer solutions at relatively lower communication cost. One major limitation of the token-based mutual exclusion algorithms for distributed environment like Raymond's well-known work is inability in handling the processes with pre-assigned priority values. Besides, the natural fairness in terms of first-come-first-serve allocation among equal priority processes too is not guaranteed in Raymond's algorithm. This has been the motivation of the first work discussed in this book. In the book, we discussed a modification of Raymond's well-known algorithm to ensure fairness in terms of first-come-first-serve (FCFS) order among equal priority processes. Subsequently, it was improved and named *Modified Raymond's Algorithm with Priority* (MRA-P). The solution considers the priority of the requesting processes and allocates resource for the earliest request when no such request from a higher priority process is pending. In MRA-P, we introduced a pair of local queues whose union would resemble a single global queue.

However, MRA-P suffered from some major shortcomings like lack of liveness, high message complexity, etc. In the next step, we further improved on these and described a token-based *Fairness Algorithm for Priority Processes* (FAPP) that addresses both the issues. FAPP justifies properties like correctness of the algorithm, low message complexity, and fairness in token allocation. Appropriate simulation has been done to benchmark FAPP and MRA-P with other existing algorithms.

Subsequently, the book describes design of a novel mutual exclusion algorithm that would be compatible to a more flexible underlying topology than the inverted tree topology on which Raymond's original algorithm as well as both MRA-P and FAPP work. It is found that there exists ME algorithms that work on a directed, acyclic graph (DAG). However, the next algorithm discussed in the book named *Link-failure Resilient Token based ME Algorithm on Graphs* (LFRT) works on any directed graph topology, with or without cycles. Like MRA-P, the LFRT algorithm ensures liveness, safety, and fairness in allocating the token on a FCFS basis. The most significant advantage of the LFRT algorithm is its ability to handle the link failures. This is possible due to redundancy in path in the underlying graph topology. LFRT was further improved to *LFRT for Priority Processes* (LFRT-P) that also considers priority of participating processes.

Besides, token-based solution, the book delves in the area of permission-based mutual exclusion algorithms. There exist efficient approaches in the literature that selects one candidate process from many for allowing it to enter its critical section (CS) on the basis of the number of votes received by the processes. In voting-based approaches, a process that gets majority of the total number of votes is only to be allowed for CS. This ensures safety for such an algorithm as no two processes can earn majority of the total number of polls. However, this may lead to a live-lock situation where no single process reaches the magic number of majority votes. In this book, a voting-based algorithm has been discussed to select a process even when no single process has majority of votes.

Two voting-based algorithms are also described in this book. We have described a voting-based mutual exclusion algorithm (BMaV) that finds a candidate for CS when majority consensus is not achieved by any single process. Another algorithm *A New Hybrid Mutual Exclusion Algorithm in Absence of Majority Consensus* (NHME-AMC) has been discussed in the book. In this book, a two-phase, hybrid ME algorithm has been discussed that works even when majority consensus cannot be reached. The second phase of the algorithm, in spite of being symmetric, executes in constant time. These algorithms aim to find an effective solution when no single process achieves majority consensus. We have concluded the book by discussing on the significance of works in this book towards setting future directions of research for process synchronization in distributed systems.

We hope that researchers in the domain of distributed operating systems and faculty members teaching this subject will find the book a useful reference during treatment of distributed control algorithms. The language of the book is lucid and all algorithms are explained with suitable examples. In spite of our best efforts there may be shortcomings that might have escaped our notice. We shall be obliged if suggestions and criticisms are put forward to improve the content of the book. We also like to gratefully acknowledge Mr. Aninda Bose and Kamiya Khatter without whose continuous support the book could not be completed.

Howrah, India  
Kolkata, India  
Kolkata, India

Sukhendu Kanrar  
Nabendu Chaki  
Samiran Chattopadhyay

Concurrency Control in Distributed System Using Mutual  
Exclusion

Kanrar, S.; Chaki, N.; Chattopadhyay, S.

2018, X, 95 p. 43 illus., Hardcover

ISBN: 978-981-10-5558-4