

Preface

*La profusion des choses cachait la rareté des idées et l'usure des croyances.
[...] Retenir quelque chose du temps où l'on ne sera plus.*

In *Les années* (2008), Annie Ernaux

*Nel mezzo del cammin di nostra vita
Mi ritrovai per una selva oscura,
Ché la diritta via era smarrita.*

In *La divina commedia* (1307–1321), Dante Alighieri (1265–1321)

*Wir müssen nichts sein, sondern alles werden wollen.
Johann Wolfgang von Goethe (1749–1832)*

*Chaque génération, sans doute, se croit vouée à refaire le monde.
La mienne sait pourtant qu'elle ne le refera pas. Mais sa tâche est peut-être plus grande.
Elle consiste à empêcher que le monde ne se défasse.*
Speech at the Nobel Banquet, Stockholm, December 10, 1957, Albert Camus (1913–1960)

*Rien n'est précaire comme vivre
Rien comme être n'est passager
C'est un peu fondre pour le givre
Ou pour le vent être léger
J'arrive où je suis étranger.*

In *Le voyage de Hollande* (1965), Louis Aragon (1897–1982)

What Is Distributed Computing? Distributed computing was born in the late 1970s when researchers and practitioners started taking into account the intrinsic characteristic of physically distributed systems. The field then emerged as a specialized research area distinct from networking, operating systems, and parallel computing.

Distributed computing arises when one has to solve a problem in terms of distributed entities (usually called processors, nodes, processes, actors, agents, sensors, peers, etc.) such that each entity has only a partial knowledge of the many parameters involved in the problem that has to be solved. While parallel computing and real-time computing can be characterized, respectively, by the terms *efficiency* and *on-time computing*, distributed computing can be characterized by the term *uncertainty*. This uncertainty is created by asynchrony, multiplicity of control flows,

absence of shared memory and global time, failure, dynamicity, mobility, etc. Mastering one form or another of uncertainty is pervasive in all distributed computing problems. A main difficulty in designing distributed algorithms comes from the fact that each entity cooperating in the achievement of a common goal cannot have instantaneous knowledge of the current state of the other entities; it can only know their past local states.

Although distributed algorithms are often made up of a few lines, their behavior can be difficult to understand and their properties hard to state and prove. Hence, distributed computing is not only a fundamental topic but also a challenging topic where simplicity, elegance, and beauty are first-class citizens.

Why This Book? While there are a lot of books on sequential computing (both on basic data structures, or algorithms), this is not the case in distributed computing. Most books on distributed computing consider advanced topics where the uncertainty inherent to distributed computing is created by the net effect of asynchrony and failures. It follows that these books are more appropriate for graduate students than for undergraduate students.

The aim of this book is to present in a comprehensive way basic notions, concepts and algorithms of distributed computing when the distributed entities cooperate by sending and receiving messages on top of an underlying network. In this case, the main difficulty comes from the physical distribution of the entities and the asynchrony of the environment in which they evolve.

Audience This book has been written primarily for people who are not familiar with the topic and the concepts that are presented. These include mainly:

- Senior-level undergraduate students and graduate students in computer science or computer engineering, who are interested in the principles and foundations of distributed computing.
- Practitioners and engineers who want to be aware of the state-of-the-art concepts, basic principles, mechanisms, and techniques encountered in distributed computing.

Prerequisites for this book include undergraduate courses on algorithms, and basic knowledge on operating systems. Selections of chapters for undergraduate and graduate courses are suggested in the section titled “How to Use This Book” in the Afterword.

Content As already indicated, this book covers algorithms, basic principles, and foundations of message-passing programming, i.e., programs where the entities communicate by sending and receiving messages through a network. The world is distributed, and the algorithmic thinking suited to distributed applications and systems is not reducible to sequential computing. Knowledge of the bases of distributed computing is becoming more important than ever as more and more computer applications are now distributed. The book is composed of six parts.

- The aim of the first part, which is made up of six chapters, is to give a feel for the nature of distributed algorithms, i.e., what makes them different from sequential or parallel algorithms. To that end, it mainly considers distributed graph algorithms. In this context, each node of the graph is a process, which has to compute a result whose meaning depends on the whole graph.

Basic distributed algorithms such as network traversals, shortest-path algorithms, vertex coloring, knot detection, etc., are first presented. Then, a general framework for distributed graph algorithms is introduced. A chapter is devoted to leader election algorithms on a ring network, and another chapter focuses on the navigation of a network by mobile objects.

- The second part is on the nature of distributed executions. It is made up of four chapters. In some sense, this part is the core of the book. It explains what a distributed execution is, the fundamental notion of a consistent global state, and the impossibility—without freezing the computation—of knowing whether a computed consistent global state has been passed through by the execution or not.

Then, this part of the book addresses an important issue of distributed computations, namely the notion of logical time: scalar (linear) time, vector time, and matrix time. Each type of time is analyzed and examples of their uses are given. A chapter, which extends the notion of a global state, is then devoted to asynchronous distributed checkpointing. Finally, the last chapter of this part shows how to simulate a synchronous system on top of an asynchronous system (such simulators are called synchronizers).

- The third part of the book is made up of two chapters devoted to distributed mutual exclusion and distributed resource allocation. Different families of permission-based mutual exclusion algorithms are presented. The notion of an adaptive algorithm is also introduced. The notion of a critical section with multiple entries, and the case of resources with a single or several instances is also presented. Associated deadlock prevention techniques are introduced.
- The fourth part of the book is on the definition and the implementation of communication operations whose abstraction level is higher than the simple send/receive of messages. These communication abstractions impose order constraints on message deliveries. Causal message delivery and total order broadcast are first presented in one chapter. Then, another chapter considers synchronous communication (also called rendezvous or logically instantaneous communication).
- The fifth part of the book, which is made up of two chapters, is on the detection of stable properties encountered in distributed computing. A stable property is a property that, once true, remains true forever. The properties which are studied are the detection of the termination of a distributed computation, and the detection of distributed deadlock. This part of the book is strongly related to the second part (which is devoted to the notion of a global state).
- The sixth and last part of the book, which is also made up of two chapters, is devoted to the notion of a distributed shared memory. The aim is here to provide the entities (processes) with a set of objects that allow them to cooperate at

an abstraction level more appropriate than the use of messages. Two consistency conditions, which can be associated with these objects, are presented and investigated, namely, atomicity (also called linearizability) and sequential consistency. Several algorithms implementing these consistency conditions are described.

To have a more complete feeling of the spirit of this book, the reader is invited to consult the section “The Aim of This Book” in the Afterword, which describes what it is hoped has been learned from this book. Each chapter starts with a short presentation and a list of the main keywords, and terminates with a summary of its content. Each of the six parts of the book is also introduced by a brief description of its aim and its technical content.

Acknowledgments This book originates from lecture notes for undergraduate and graduate courses on distributed computing that I give at the University of Rennes (France) and, as an invited professor, at several universities all over the world. I would like to thank the students for their questions that, in one way or another, have contributed to this book. I want also to thank Ronan Nugent (Springer) for his support and his help in putting it all together.

Last but not least (and maybe most importantly), I also want to thank all the researchers whose results are presented in this book. Without their work, this book would not exist.

Michel Raynal

Professeur des Universités
Institut Universitaire de France
IRISA-ISTIC, Université de Rennes 1
Campus de Beaulieu, 35042, Rennes, France

March–October 2012

Rennes, Saint-Grégoire, Tokyo, Fukuoka (AINA’12), Arequipa (LATIN’12), Reykjavik (SIROCCO’12), Palermo (CISIS’12), Madeira (PODC’12), Lisbon, Douelle, Saint-Philibert, Rhodes Island (Europar’12), Salvador de Bahia (DISC’12), Mexico City (Turing Year at UNAM)



<http://www.springer.com/978-3-642-38122-5>

Distributed Algorithms for Message-Passing Systems

Raynal, M.

2013, XXXI, 500 p., Hardcover

ISBN: 978-3-642-38122-5