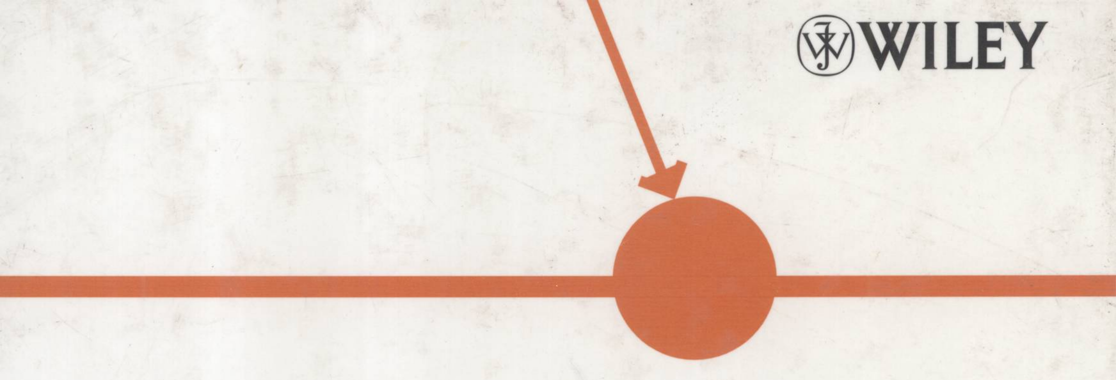
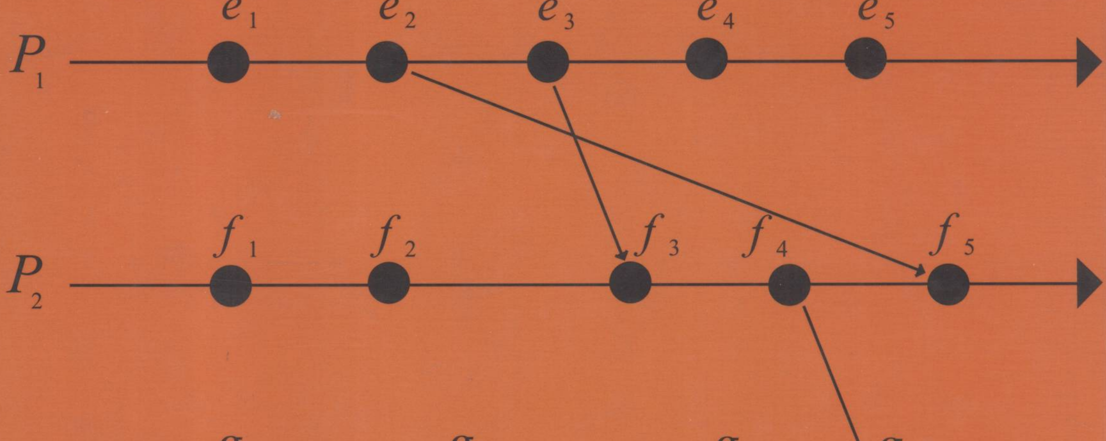
WILEY



ELEMENTS OF DISTRIBUTED COMPUTING



VIJAY K. GARG

ELEMENTS OF DISTRIBUTED COMPUTING

VIJAY K. GARG
Department of Electrical and Computer Engineering
University of Texas at Austin

江苏工业学院图书馆
藏书章

WILEY-
INTERSCIENCE

A John Wiley & Sons, Inc., Publication

This text is printed on acid-free paper. ♾

Copyright © 2002 by John Wiley & Sons, Inc., New York. All rights reserved.

Published simultaneously in Canada.

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Section 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 750-4744. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 605 Third Avenue, New York, NY 10158-0012, (212) 850-6011, fax (212) 850-6008, E-Mail: PERMREQ @ WILEY.COM.

For ordering and customer service, call 1-800-CALL WILEY.

Library of Congress Cataloging-in-Publication Data is available.

ISBN 0-471-03600-5

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ELEMENTS OF DISTRIBUTED COMPUTING

To
my parents (Shri Saran Lal and Shrimati Laxmi Devi),
my wife (Meenakshi),
and my children (Sonika and Rohan)

此为试读，需要完整PDF请访问：www.ertongbook.com

Foreword

Distributed computing arises as soon as one has to solve a problem in terms of processes that individually have only a partial knowledge of the several parameters associated with the problem. Thus, distributed computing appears everywhere (even in everyday life!) and is consequently at the heart of lots of applications. The difficulty in designing a distributed algorithm comes from the fact that each of the processes cooperating in the achievement of the common goal cannot have instantaneous knowledge of the current state of the other processes; it can only know their past states. Moreover, because of the possibility of process or communication failures, this knowledge can become very difficult or even impossible to obtain.

Whereas parallel computing is mainly concerned with efficiency, distributed computing addresses uncertainty created by the multiplicity of control flows, the absence of shared memory and global time, and the occurrence of failures. Although it is true that distributed algorithms are often made up of only a few lines, their behaviors 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 criteria.

In this book, Vijay K. Garg provides a comprehensive and timely treatment of all these challenges. In a clear and consistent way, he addresses the fundamental problems, issues, and solutions that define the core of distributed computing. The book is both versatile and complete. While each chapter (designed to correspond to one lecture) addresses a basic issue of distributed computing, the examination of chapters addressing several facets of the same topic provides the reader with a deep insight into the subject and a good understanding of the subtleties one has to cope with when solving distributed computing problems. Moreover, the exercises accompanying each chapter can help the reader test her acquaintance with the subject concerned.

The book is addressed to all those who are concerned with algorithms, distributed computing, and distributed systems. It will provide software engineers with provably correct distributed protocols, researchers with a large collection of up-to-date results, teachers with a solid foundation for a course on distributed computing, and students with an introduction to distributed algorithms for distributed computing.

Professor Michel Raynal
IRISA,
Université de Rennes 1
Campus de Beaulieu, Rennes
France

18 February 2002

Preface

The existing books on distributed computing have mainly been developed as reference books on distributed algorithms. They are more useful to researchers than to students. The present book was designed keeping students in mind. Simplicity has been chosen at the cost of completeness whenever that choice was required.

The topics have been selected very carefully. No book, short of a thick encyclopedia, can cover all topics in distributed systems. I have chosen topics that have practical usefulness as well as aesthetic beauty. Because usefulness and beauty are only in the eyes of the beholder, the selection was necessarily subjective. In my defense, I will only give a brief rationale behind the choice of topics. My rule for selecting topics was simple: I have chosen algorithms over arguments for lower bounds and general mechanisms over special case optimized algorithms—the former choice for usefulness and the latter for beauty. For the sake of usefulness or beauty, I violated my general rule whenever necessary; for example, the result on impossibility of consensus in presence of failures (a negative result) and the algorithm for termination detection (a special case of stable property detection) have been included.

Most proofs in the book are rigorous but not completely formal. I have avoided proofs by contradiction. To quote Royden from his book on Real Analysis: "All students are enjoined in the strongest possible terms to eschew proof by contradiction! There are two reasons for this prohibition: First, such proofs are very often fallacious, the contradiction on the final page arising from an erroneous deduction on an earlier page,... Second, even when correct, such a proof gives little insight into connection between A and B (antecedent and consequent)."

The book is organized as a large number of small chapters rather than a small number of large chapters. Besides the obvious advantage of providing more occasions for satisfaction gained on finishing a chapter, this organization provides more flexibility to the instructor. Assuming that there is little dependency across chapters, as is the case for the present book, the instructor can substitute his or her own handouts and papers for some chapters.

The book is designed for a graduate level course on distributed systems. Several examples and exercise problems are included in each chapter to facilitate classroom teaching. My treatment is based on the graduate courses given at The University of Texas at Austin.

The list of known errors and the supplementary material for the book will be maintained on my home page:

`http://www.ece.utexas.edu/~garg`

I would like to thank the following people for working with me on various projects discussed in this book. These people include Craig Chase (weak predicates), Om Damani (message logging), Eddy Fromentin (predicate detection), Joydeep Ghosh (global computation), Richard Kilgore (channel predicates), Roger Mitchell (channel predicates), Neeraj Mittal (predicate detection and control, slicing, self-stabilization, distributed shared memory), Venkat Murty (synchronous ordering), Michel Raynal (control flow properties, distributed shared memory), Alper Sen (slicing), Chakarat Skawratonand (vector clocks), Ashis Tarafdar (message logging, predicate control) Alexander Tomlinson (global time, mutual exclusion, relational predicates, control flow properties) and Brian Waldecker (weak and strong predicates). I owe special thanks to Alper Sen for helping me with figures and his meticulous review of an early draft of this book. Ajay Kshemkalyani and Michel Raynal provided valuable feedback on the contents of the book.

I thank the Department of Electrical and Computer Engineering at The University of Texas at Austin, where I was given the opportunity to develop and teach courses on distributed systems. Students in my course gave me very useful feedback. I was supported in part by many grants from the National Science Foundation over the last twelve years. Many of the results reported in this book would not have been discovered by me and my research group without that support. I also thank John Wiley & Sons, Inc. for supporting the project.

Finally, I thank my wife Meenakshi, whose love, support, and understanding kept me inspired in accomplishing this goal.

Contents

Foreword **xvii**

Preface **xix**

1 Introduction **1**

1.1 Introduction 1
1.2 Distributed Systems Versus Parallel Systems 1
1.3 Characteristics of Distributed Systems 3
1.4 Scope of the Book 4
1.5 Overview of the Book 5
1.6 Notation 9
1.7 Problems 10
1.8 Bibliographic Remarks 10

2 Model of a Computation **11**

2.1 Introduction 11
2.2 Model of a Distributed System 12
2.3 Interleaving Model 14
2.4 Happened Before Model 16
2.5 Potential Causality Model 17
2.6 Appropriate Model 19
2.7 Models Based on States 20
2.7.1 Deposet 22
2.7.2 Global Sequence of States 23
2.8 Problems 25
2.9 Bibliographic Remarks 25

3 Logical Clocks **27**

3.1 Introduction 27
3.2 Logical Clocks 28

3.2.1 Definition 28
3.2.2 Implementation 32
3.3 Vector Clocks 34
3.3.1 Definition 34
3.3.2 Implementation 34
3.4 Problems 37
3.5 Bibliographic Remarks 38

4 Verifying Clock Algorithms 39
4.1 Introduction 39
4.2 Using Induction on $\rightarrow$ 40
4.3 Proof of the Vector Clock Algorithm 42
4.4 Problems 47
4.5 Bibliographic Remarks 47

5 Clocks of Different Dimensions 49
5.1 Introduction 49
5.2 Direct-Dependency Clocks 50
5.2.1 Algorithm 50
5.2.2 Properties 51
5.3 Higher-Dimensional Clocks 55
5.3.1 State Interval 56
5.3.2 Predecessor and Successor Functions 56
5.3.3 Algorithm for Matrix Clock 57
5.3.4 Application: Garbage Collection 60
5.4 Problems 61
5.5 Bibliographic Remarks 63

6 Mutual Exclusion: Using Timestamps 65
6.1 Introduction 65
6.2 Specifications of the Problem 67
6.3 Lamport's Algorithm 69
6.3.1 Proof of Correctness 71
6.4 Ricart and Agrawala's Algorithm 76
6.5 Problems 78
6.6 Bibliographic Remarks 79

7 Mutual Exclusion: Tokens and Quorums 81
7.1 Introduction 81
7.2 Centralized Algorithm 82

此为试读,需要完整PDF请访问: www.ertongbook.com

7.2.1 Proof of Correctness 85
7.3 An Exercise in Decentralization 85
7.4 Quorum-Based Algorithms 86
7.4.1 Voting Systems 89
7.4.2 Crumbling Walls 89
7.5 Problems 90
7.6 Bibliographic Remarks 91

8 Drinking Philosophers Problem **93**
8.1 Introduction 93
8.2 Dining Philosophers—Heavy Load 94
8.3 Dining Philosophers—Light Load 97
8.4 Drinking Philosophers 100
8.5 Problems 104
8.6 Bibliographic Remarks 104

9 Leader Election **107**
9.1 Introduction 107
9.2 Anonymous Rings 108
9.3 Chang-Roberts Algorithm 109
9.4 Hirschberg-Sinclair Algorithm 111
9.5 Problems 112
9.6 Bibliographic Remarks 113

10 Global State **115**
10.1 Introduction 115
10.2 Consistent Cuts 118
10.3 Consistent Subcuts 119
10.4 Global Snapshot Algorithm 121
10.5 Application: Detecting Stable Properties 125
10.6 Problems 127
10.7 Bibliographic Remarks 128

11 Observing Global Predicates **131**
11.1 Introduction 131
11.2 Modalities of Observation 132
11.3 Key Problems in Observation of Global Properties . . . 135
11.3.1 Lack of Shared Clock 135
11.3.2 Lack of Shared Memory 136
11.3.3 Combinatorial Explosion 137

11.4 Linear Predicates 140
11.4.1 The Least Cut Satisfying a Linear Predicate . . 141
11.5 Regular Predicates 144
11.6 Problems 145
11.7 Bibliographic Remarks 148

12 Observing Conjunctive Predicates 149
12.1 Introduction 149
12.2 Boolean Expressions of Local Predicates 150
12.3 A Vector Clock-Based Centralized Algorithm 152
12.3.1 Application Process 152
12.3.2 Checker Process 152
12.3.3 Proof of Correctness 154
12.3.4 Overhead Analysis 156
12.4 A Direct Dependency-Based Algorithm 158
12.4.1 Application Process 159
12.4.2 Checker Process 159
12.4.3 Proof of Correctness 161
12.4.4 Overhead Analysis 161
12.5 A Vector Clock-Based Distributed Algorithm 162
12.5.1 Overhead Analysis 163
12.6 Problems 164
12.7 Bibliographic Remarks 166

13 Channel Predicates 167
13.1 Introduction 167
13.2 Linear Channel Predicates 168
13.3 A Centralized Algorithm 170
13.3.1 Application Processes 170
13.3.2 Checker Process 170
13.3.3 Overhead Analysis 174
13.4 Problems 175
13.5 Bibliographic Remarks 175

14 Termination Detection 177
14.1 Introduction 177
14.2 Dijkstra and Scholten's Algorithm 178
14.3 An Optimization 180
14.4 Problems 180
14.5 Bibliographic Remarks 182

15 Control of a Distributed Computation **183**
15.1 Introduction 183
15.2 Hardness of the Control Problem 185
15.3 Mutual Exclusion 187
15.4 Disjunctive Predicates 189
15.5 Relationship Between Observation and Control 192
15.6 Problems 193
15.7 Bibliographic Remarks 194

16 Causal Message Ordering **195**
16.1 Introduction 195
16.2 Algorithm 197
16.3 Proof of Correctness 198
16.4 Problems 200
16.5 Bibliographic Remarks 202

17 Synchronous and Total Order **203**
17.1 Introduction 203
17.2 Synchronous Ordering 207
17.2.1 Relationship Among Message Orderings 208
17.2.2 Crowns in a Distributed Computation 208
17.3 Algorithm 210
17.3.1 Proof of Correctness 211
17.4 Total Order for Multicast Messages 213
17.4.1 Centralized Algorithm 214
17.4.2 Lamport's Algorithm for Total Order 214
17.4.3 Skeen's Algorithm 215
17.5 Application: Replicated State Machines 216
17.6 Problems 216
17.7 Bibliographic Remarks 217

18 Computation of a Global Function **219**
18.1 Introduction 219
18.2 Convergecast and Broadcast 220
18.3 Global Functions 223
18.4 Problems 229
18.5 Bibliographic Remarks 230

19 Repeated Global Computation **231**
19.1 Introduction 231
19.2 An Equitable, Revolving Hierarchy 232
19.3 Implementation Issues 239
19.4 Application: Branch-and-Bound Algorithm 241
19.5 Problems 242
19.6 Bibliographic Remarks 244

20 Synchronizers **245**
20.1 Introduction 245
20.2 A Simple Synchronizer 247
20.3 Synchronizer α 248
20.4 Synchronizer β 249
20.5 Synchronizer γ 249
20.6 Application: BFS Tree Algorithm 251
20.7 Limitations of Synchronizers 252
20.8 Problems 253
20.9 Bibliographic Remarks 253

21 Slicers **255**
21.1 Introduction 255
21.2 Characterization of an Execution Lattice 257
21.3 Slicing a Distributed Computation 259
21.4 Algorithm for Slicing Regular Predicates 263
21.4.1 Step 1: Computing the Least Cut 263
21.4.2 Step 2: Computing the Greatest Cut 265
21.4.3 Step 3: Computing Join-Irreducible Elements . . 265
21.4.4 Step 4: Forming Equivalence Classes 266
21.5 Application: Predicate Detection and Control 268
21.6 Problems 269
21.7 Bibliographic Remarks 270

22 Distributed Shared Memory **271**
22.1 Introduction 271
22.2 System Model 273
22.3 Sequential Consistency 274
22.3.1 Local Read Algorithm 277
22.3.2 Local Write Algorithm 278
22.4 Linearizability 280
22.5 Other Consistency Conditions 282

22.6 Problems 284
22.7 Bibliographic Remarks 285

23 Self-Stabilization 287
23.1 Introduction 287
23.2 Mutual Exclusion with K-State Machines 288
23.2.1 Proof of Correctness 289
23.3 Mutual Exclusion with Three-State Machines 291
23.3.1 Proof of Correctness 291
23.4 Problems 295
23.5 Bibliographic Remarks 296

24 Knowledge and Common Knowledge 297
24.1 Introduction 297
24.2 Knowledge and Common Knowledge 298
24.3 Application: Two-Generals Problem 300
24.4 Defining Knowledge Based on Isomorphism 302
24.5 Problems 303
24.6 Bibliographic Remarks 304

25 Consensus Under Asynchrony 307
25.1 Introduction 307
25.2 Requirements and Assumptions 308
25.3 An Informal Proof of the Impossibility Result 311
25.4 A Formal Proof of the Impossibility Result 314
25.4.1 Requirements 315
25.4.2 Classification of Global States 315
25.4.3 Impossibility Result 317
25.5 Terminating Reliable Broadcast 322
25.6 Problems 322
25.7 Bibliographic Remarks 323

26 Consensus Under Synchrony 325
26.1 Introduction 325
26.2 Consensus Under Crash Failures 326
26.2.1 An Algorithm for Consensus 327
26.2.2 An Early Stopping Algorithm 329
26.3 Consensus Under Byzantine Faults 331
26.4 Problems 334
26.5 Bibliographic Remarks 334

27 Failure Detectors **337**
27.1 Introduction 337
27.2 Completeness and Accuracy Properties 338
27.3 Relationship Among Various Failure Detectors 340
27.4 Algorithm for Consensus Using $\diamond\mathcal{S}$ 341
27.4.1 Proof of Correctness 342
27.5 Problems 344
27.6 Bibliographic Remarks 345

28 Solvable Problems in Asynchronous Systems **347**
28.1 Introduction 347
28.2 Failure Detection 348
28.2.1 Infinitely Often Accurate Detectors 349
28.2.2 Application: Maintaining a Leader 353
28.3 k-Set Consensus Problem 359
28.4 Reliable Broadcast Problem 360
28.5 Problems 361
28.6 Bibliographic Remarks 361

29 Checkpointing for Recovery **363**
29.1 Introduction 363
29.2 Zig-Zag Relation 365
29.3 R-Graphs 368
29.4 Recoverable Global States: Using Slices 369
29.5 Rollback Dependency Trackability (RDT) 371
29.6 Communication-Induced Checkpointing 372
29.7 Problems 373
29.8 Bibliographic Remarks 374

30 Message Logging for Recovery **375**
30.1 Introduction 375
30.2 Model 377
30.2.1 Happened Before With Process Failures 378
30.2.2 Orphan States 379
30.3 Fault-Tolerant Vector Clock 379
30.3.1 Properties of FTVC 381
30.4 Version End Table 384
30.5 The Protocol 384
30.5.1 An Example 388
30.5.2 Proof of Correctness 389

此为试读,需要完整PDF请访问: www.ertongbook.com