

Contents

Part I Distributed Graph Algorithms

1 Basic Definitions and Network Traversal Algorithms	3
1.1 Distributed Algorithms	3
1.1.1 Definition	3
1.1.2 An Introductory Example: Learning the Communication Graph	6
1.2 Parallel Traversal: Broadcast and Convergecast	9
1.2.1 Broadcast and Convergecast	9
1.2.2 A Flooding Algorithm	10
1.2.3 Broadcast/Convergecast Based on a Rooted Spanning Tree	10
1.2.4 Building a Spanning Tree	12
1.3 Breadth-First Spanning Tree	16
1.3.1 Breadth-First Spanning Tree Built Without Centralized Control	17
1.3.2 Breadth-First Spanning Tree Built with Centralized Control	20
1.4 Depth-First Traversal	24
1.4.1 A Simple Algorithm	24
1.4.2 Application: Construction of a Logical Ring	27
1.5 Summary	32
1.6 Bibliographic Notes	32
1.7 Exercises and Problems	33
2 Distributed Graph Algorithms	35
2.1 Distributed Shortest Path Algorithms	35
2.1.1 A Distributed Adaptation of Bellman–Ford’s Shortest Path Algorithm	35
2.1.2 A Distributed Adaptation of Floyd–Warshall’s Shortest Paths Algorithm	38
2.2 Vertex Coloring and Maximal Independent Set	42
2.2.1 On Sequential Vertex Coloring	42

2.2.2	Distributed $(\Delta + 1)$ -Coloring of Processes	43
2.2.3	Computing a Maximal Independent Set	46
2.3	Knot and Cycle Detection	50
2.3.1	Directed Graph, Knot, and Cycle	50
2.3.2	Communication Graph, Logical Directed Graph, and Reachability	51
2.3.3	Specification of the Knot Detection Problem	51
2.3.4	Principle of the Knot/Cycle Detection Algorithm	52
2.3.5	Local Variables	53
2.3.6	Behavior of a Process	54
2.4	Summary	57
2.5	Bibliographic Notes	58
2.6	Exercises and Problems	58
3	An Algorithmic Framework	
	to Compute Global Functions on a Process Graph	59
3.1	Distributed Computation of Global Functions	59
3.1.1	Type of Global Functions	59
3.1.2	Constraints on the Computation	60
3.2	An Algorithmic Framework	61
3.2.1	A Round-Based Framework	61
3.2.2	When the Diameter Is Not Known	64
3.3	Distributed Determination of Cut Vertices	66
3.3.1	Cut Vertices	66
3.3.2	An Algorithm Determining Cut Vertices	67
3.4	Improving the Framework	69
3.4.1	Two Types of Filtering	69
3.4.2	An Improved Algorithm	70
3.5	The Case of Regular Communication Graphs	72
3.5.1	Tradeoff Between Graph Topology and Number of Rounds	72
3.5.2	De Bruijn Graphs	73
3.6	Summary	75
3.7	Bibliographic Notes	76
3.8	Problem	76
4	Leader Election Algorithms	77
4.1	The Leader Election Problem	77
4.1.1	Problem Definition	77
4.1.2	Anonymous Systems: An Impossibility Result	78
4.1.3	Basic Assumptions and Principles of the Election Algorithms	79
4.2	A Simple $O(n^2)$ Leader Election Algorithm for Unidirectional Rings	79
4.2.1	Context and Principle	79
4.2.2	The Algorithm	80
4.2.3	Time Cost of the Algorithm	80

4.2.4	Message Cost of the Algorithm	81
4.2.5	A Simple Variant	82
4.3	An $O(n \log n)$ Leader Election Algorithm for Bidirectional Rings .	83
4.3.1	Context and Principle	83
4.3.2	The Algorithm	84
4.3.3	Time and Message Complexities	85
4.4	An $O(n \log n)$ Election Algorithm for Unidirectional Rings	86
4.4.1	Context and Principles	86
4.4.2	The Algorithm	88
4.4.3	Discussion: Message Complexity and FIFO Channels	89
4.5	Two Particular Cases	89
4.6	Summary	90
4.7	Bibliographic Notes	90
4.8	Exercises and Problems	91
5	Mobile Objects Navigating a Network	93
5.1	Mobile Object in a Process Graph	93
5.1.1	Problem Definition	93
5.1.2	Mobile Object Versus Mutual Exclusion	94
5.1.3	A Centralized (Home-Based) Algorithm	94
5.1.4	The Algorithms Presented in This Chapter	95
5.2	A Navigation Algorithm for a Complete Network	96
5.2.1	Underlying Principles	96
5.2.2	The Algorithm	97
5.3	A Navigation Algorithm Based on a Spanning Tree	100
5.3.1	Principles of the Algorithm: Tree Invariant and Proxy Behavior	101
5.3.2	The Algorithm	102
5.3.3	Discussion and Properties	104
5.3.4	Proof of the Algorithm	106
5.4	An Adaptive Navigation Algorithm	108
5.4.1	The Adaptivity Property	109
5.4.2	Principle of the Implementation	109
5.4.3	An Adaptive Algorithm Based on a Distributed Queue	111
5.4.4	Properties	113
5.4.5	Example of an Execution	114
5.5	Summary	115
5.6	Bibliographic Notes	115
5.7	Exercises and Problems	116

Part II Logical Time and Global States in Distributed Systems

6	Nature of Distributed Computations and the Concept of a Global State	121
6.1	A Distributed Execution Is a Partial Order on Local Events	122
6.1.1	Basic Definitions	122

6.1.2	A Distributed Execution Is a Partial Order on Local Events	122
6.1.3	Causal Past, Causal Future, Concurrency, Cut	123
6.1.4	Asynchronous Distributed Execution with Respect to Physical Time	125
6.2	A Distributed Execution Is a Partial Order on Local States	127
6.3	Global State and Lattice of Global States	129
6.3.1	The Concept of a Global State	129
6.3.2	Lattice of Global States	129
6.3.3	Sequential Observations	131
6.4	Global States Including Process States and Channel States	132
6.4.1	Global State Including Channel States	132
6.4.2	Consistent Global State Including Channel States	133
6.4.3	Consistent Global State Versus Consistent Cut	134
6.5	On-the-Fly Computation of Global States	135
6.5.1	Global State Computation Is an Observation Problem	135
6.5.2	Problem Definition	136
6.5.3	On the Meaning of the Computed Global State	136
6.5.4	Principles of Algorithms Computing a Global State	137
6.6	A Global State Algorithm Suited to FIFO Channels	138
6.6.1	Principle of the Algorithm	138
6.6.2	The Algorithm	140
6.6.3	Example of an Execution	141
6.7	A Global State Algorithm Suited to Non-FIFO Channels	143
6.7.1	The Algorithm and Its Principles	144
6.7.2	How to Compute the State of the Channels	144
6.8	Summary	146
6.9	Bibliographic Notes	146
6.10	Exercises and Problems	147
7	Logical Time in Asynchronous Distributed Systems	149
7.1	Linear Time	149
7.1.1	Scalar (or Linear) Time	150
7.1.2	From Partial Order to Total Order: The Notion of a Timestamp	151
7.1.3	Relating Logical Time and Timestamps with Observations	152
7.1.4	Timestamps in Action: Total Order Broadcast	153
7.2	Vector Time	159
7.2.1	Vector Time and Vector Clocks	159
7.2.2	Vector Clock Properties	162
7.2.3	On the Development of Vector Time	163
7.2.4	Relating Vector Time and Global States	165
7.2.5	Vector Clocks in Action: On-the-Fly Determination of a Global State Property	166
7.2.6	Vector Clocks in Action: On-the-Fly Determination of the Immediate Predecessors	170
7.3	On the Size of Vector Clocks	173

7.3.1	A Lower Bound on the Size of Vector Clocks	174
7.3.2	An Efficient Implementation of Vector Clocks	176
7.3.3	k -Restricted Vector Clock	181
7.4	Matrix Time	182
7.4.1	Matrix Clock: Definition and Algorithm	182
7.4.2	A Variant of Matrix Time in Action: Discard Old Data . . .	184
7.5	Summary	186
7.6	Bibliographic Notes	186
7.7	Exercises and Problems	187
8	Asynchronous Distributed Checkpointing	189
8.1	Definitions and Main Theorem	189
8.1.1	Local and Global Checkpoints	189
8.1.2	Z-Dependency, Zigzag Paths, and Z-Cycles	190
8.1.3	The Main Theorem	192
8.2	Consistent Checkpointing Abstractions	196
8.2.1	Z-Cycle-Freedom	196
8.2.2	Rollback-Dependency Trackability	197
8.2.3	On Distributed Checkpointing Algorithms	198
8.3	Checkpointing Algorithms Ensuring Z-Cycle Prevention	199
8.3.1	An Operational Characterization of Z-Cycle-Freedom . . .	199
8.3.2	A Property of a Particular Dating System	199
8.3.3	Two Simple Algorithms Ensuring Z-Cycle Prevention . . .	201
8.3.4	On the Notion of an Optimal Algorithm for Z-Cycle Prevention	203
8.4	Checkpointing Algorithms Ensuring Rollback-Dependency Trackability	203
8.4.1	Rollback-Dependency Trackability (RDT)	203
8.4.2	A Simple Brute Force RDT Checkpointing Algorithm . . .	205
8.4.3	The Fixed Dependency After Send (FDAS) RDT Checkpointing Algorithm	206
8.4.4	Still Reducing the Number of Forced Local Checkpoints . .	207
8.5	Message Logging for Uncoordinated Checkpointing	211
8.5.1	Uncoordinated Checkpointing	211
8.5.2	To Log or Not to Log Messages on Stable Storage	211
8.5.3	A Recovery Algorithm	214
8.5.4	A Few Improvements	215
8.6	Summary	216
8.7	Bibliographic Notes	216
8.8	Exercises and Problems	217
9	Simulating Synchrony on Top of Asynchronous Systems	219
9.1	Synchronous Systems, Asynchronous Systems, and Synchronizers .	219
9.1.1	Synchronous Systems	219
9.1.2	Asynchronous Systems and Synchronizers	221
9.1.3	On the Efficiency Side	222

9.2	Basic Principle for a Synchronizer	223
9.2.1	The Main Problem to Solve	223
9.2.2	Principle of the Solutions	224
9.3	Basic Synchronizers: α and β	224
9.3.1	Synchronizer α	224
9.3.2	Synchronizer β	227
9.4	Advanced Synchronizers: γ and δ	230
9.4.1	Synchronizer γ	230
9.4.2	Synchronizer δ	234
9.5	The Case of Networks with Bounded Delays	236
9.5.1	Context and Hypotheses	236
9.5.2	The Problem to Solve	237
9.5.3	Synchronizer λ	238
9.5.4	Synchronizer μ	239
9.5.5	When the Local Physical Clocks Drift	240
9.6	Summary	242
9.7	Bibliographic Notes	243
9.8	Exercises and Problems	244

Part III Mutual Exclusion and Resource Allocation

10	Permission-Based Mutual Exclusion Algorithms	247
10.1	The Mutual Exclusion Problem	247
10.1.1	Definition	247
10.1.2	Classes of Distributed Mutex Algorithms	248
10.2	A Simple Algorithm Based on Individual Permissions	249
10.2.1	Principle of the Algorithm	249
10.2.2	The Algorithm	251
10.2.3	Proof of the Algorithm	252
10.2.4	From Simple Mutex to Mutex on Classes of Operations	255
10.3	Adaptive Mutex Algorithms Based on Individual Permissions	256
10.3.1	The Notion of an Adaptive Algorithm	256
10.3.2	A Timestamp-Based Adaptive Algorithm	257
10.3.3	A Bounded Adaptive Algorithm	259
10.3.4	Proof of the Bounded Adaptive Mutex Algorithm	262
10.4	An Algorithm Based on Arbiter Permissions	264
10.4.1	Permissions Managed by Arbiters	264
10.4.2	Permissions Versus Quorums	265
10.4.3	Quorum Construction	266
10.4.4	An Adaptive Mutex Algorithm Based on Arbiter Permissions	268
10.5	Summary	273
10.6	Bibliographic Notes	273
10.7	Exercises and Problems	274

11	Distributed Resource Allocation	277
11.1	A Single Resource with Several Instances	277
11.1.1	The k -out-of- M Problem	277
11.1.2	Mutual Exclusion with Multiple Entries:	
	The 1-out-of- M Mutex Problem	278
11.1.3	An Algorithm for the k -out-of- M Mutex Problem	280
11.1.4	Proof of the Algorithm	283
11.1.5	From Mutex Algorithms to k -out-of- M Algorithms	285
11.2	Several Resources with a Single Instance	285
11.2.1	Several Resources with a Single Instance	286
11.2.2	Incremental Requests for Single Instance Resources:	
	Using a Total Order	287
11.2.3	Incremental Requests for Single Instance Resources:	
	Reducing Process Waiting Chains	290
11.2.4	Simultaneous Requests for Single Instance Resources	
	and Static Sessions	292
11.2.5	Simultaneous Requests for Single Instance Resources	
	and Dynamic Sessions	293
11.3	Several Resources with Multiple Instances	295
11.4	Summary	297
11.5	Bibliographic Notes	298
11.6	Exercises and Problems	299

Part IV High-Level Communication Abstractions

12	Order Constraints on Message Delivery	303
12.1	The Causal Message Delivery Abstraction	303
12.1.1	Definition of Causal Message Delivery	304
12.1.2	A Causality-Based Characterization	
	of Causal Message Delivery	305
12.1.3	Causal Order	
	with Respect to Other Message Ordering Constraints	306
12.2	A Basic Algorithm for Point-to-Point Causal Message Delivery	306
12.2.1	A Simple Algorithm	306
12.2.2	Proof of the Algorithm	309
12.2.3	Reduce the Size of Control Information	
	Carried by Messages	310
12.3	Causal Broadcast	313
12.3.1	Definition and a Simple Algorithm	313
12.3.2	The Notion of a Causal Barrier	315
12.3.3	Causal Broadcast with Bounded Lifetime Messages	317
12.4	The Total Order Broadcast Abstraction	320
12.4.1	Strong Total Order Versus Weak Total Order	320
12.4.2	An Algorithm Based on a Coordinator Process	
	or a Circulating Token	322

12.4.3	An Inquiry-Based Algorithm	324
12.4.4	An Algorithm for Synchronous Systems	326
12.5	Playing with a Single Channel	328
12.5.1	Four Order Properties on a Channel	328
12.5.2	A General Algorithm Implementing These Properties . . .	329
12.6	Summary	332
12.7	Bibliographic Notes	332
12.8	Exercises and Problems	333
13	Rendezvous (Synchronous) Communication	335
13.1	The Synchronous Communication Abstraction	335
13.1.1	Definition	335
13.1.2	An Example of Use	337
13.1.3	A Message Pattern-Based Characterization	338
13.1.4	Types of Algorithms Implementing Synchronous Communications	341
13.2	Algorithms for Nondeterministic Planned Interactions	341
13.2.1	Deterministic and Nondeterministic Communication Contexts	341
13.2.2	An Asymmetric (Static) Client–Server Implementation . .	342
13.2.3	An Asymmetric Token-Based Implementation	345
13.3	An Algorithm for Nondeterministic Forced Interactions	350
13.3.1	Nondeterministic Forced Interactions	350
13.3.2	A Simple Algorithm	350
13.3.3	Proof of the Algorithm	352
13.4	Rendezvous with Deadlines in Synchronous Systems	354
13.4.1	Synchronous Systems and Rendezvous with Deadline . . .	354
13.4.2	Rendezvous with Deadline Between Two Processes	355
13.4.3	Introducing Nondeterministic Choice	358
13.4.4	n -Way Rendezvous with Deadline	360
13.5	Summary	361
13.6	Bibliographic Notes	361
13.7	Exercises and Problems	362
 Part V Detection of Properties on Distributed Executions		
14	Distributed Termination Detection	367
14.1	The Distributed Termination Detection Problem	367
14.1.1	Process and Channel States	367
14.1.2	Termination Predicate	368
14.1.3	The Termination Detection Problem	369
14.1.4	Types and Structure of Termination Detection Algorithms .	369
14.2	Termination Detection in the Asynchronous Atomic Model	370
14.2.1	The Atomic Model	370

14.2.2	The Four-Counter Algorithm	371
14.2.3	The Counting Vector Algorithm	373
14.2.4	The Four-Counter Algorithm vs. the Counting Vector Algorithm	376
14.3	Termination Detection in Diffusing Computations	376
14.3.1	The Notion of a Diffusing Computation	376
14.3.2	A Detection Algorithm Suited to Diffusing Computations	377
14.4	A General Termination Detection Algorithm	378
14.4.1	Wave and Sequence of Waves	379
14.4.2	A Reasoned Construction	381
14.5	Termination Detection in a Very General Distributed Model	385
14.5.1	Model and Nondeterministic Atomic Receive Statement	385
14.5.2	The Predicate <i>fulfilled()</i>	387
14.5.3	Static vs. Dynamic Termination: Definition	388
14.5.4	Detection of Static Termination	390
14.5.5	Detection of Dynamic Termination	393
14.6	Summary	396
14.7	Bibliographic Notes	396
14.8	Exercises and Problems	397
15	Distributed Deadlock Detection	401
15.1	The Deadlock Detection Problem	401
15.1.1	Wait-For Graph (WFG)	401
15.1.2	AND and OR Models Associated with Deadlock	403
15.1.3	Deadlock in the AND Model	403
15.1.4	Deadlock in the OR Model	404
15.1.5	The Deadlock Detection Problem	404
15.1.6	Structure of Deadlock Detection Algorithms	405
15.2	Deadlock Detection in the One-at-a-Time Model	405
15.2.1	Principle and Local Variables	406
15.2.2	A Detection Algorithm	406
15.2.3	Proof of the Algorithm	407
15.3	Deadlock Detection in the AND Communication Model	408
15.3.1	Model and Principle of the Algorithm	409
15.3.2	A Detection Algorithm	409
15.3.3	Proof of the Algorithm	411
15.4	Deadlock Detection in the OR Communication Model	413
15.4.1	Principle	413
15.4.2	A Detection Algorithm	416
15.4.3	Proof of the Algorithm	419
15.5	Summary	421
15.6	Bibliographic Notes	421
15.7	Exercises and Problems	422

Part VI Distributed Shared Memory

16 Atomic Consistency (Linearizability)	427
16.1 The Concept of a Distributed Shared Memory	427
16.2 The Atomicity Consistency Condition	429
16.2.1 What Is the Issue?	429
16.2.2 An Execution Is a Partial Order on Operations	429
16.2.3 Atomicity: Formal Definition	430
16.3 Atomic Objects Compose for Free	432
16.4 Message-Passing Implementations of Atomicity	435
16.4.1 Atomicity Based on	
a Total Order Broadcast Abstraction	435
16.4.2 Atomicity of Read/Write Objects Based on	
Server Processes	437
16.4.3 Atomicity Based on	
a Server Process and Copy Invalidation	438
16.4.4 Introducing the Notion of an Owner Process	439
16.4.5 Atomicity Based on a Server Process and Copy Update	443
16.5 Summary	444
16.6 Bibliographic Notes	444
16.7 Exercises and Problems	445
17 Sequential Consistency	447
17.1 Sequential Consistency	447
17.1.1 Definition	447
17.1.2 Sequential Consistency Is Not a Local Property	449
17.1.3 Partial Order for Sequential Consistency	450
17.1.4 Two Theorems	
for Sequentially Consistent Read/Write Registers	451
17.1.5 From Theorems to Algorithms	453
17.2 Sequential Consistency from Total Order Broadcast	453
17.2.1 A Fast Read Algorithm for Read/Write Objects	453
17.2.2 A Fast Write Algorithm for Read/Write Objects	455
17.2.3 A Fast Enqueue Algorithm for Queue Objects	456
17.3 Sequential Consistency from a Single Server	456
17.3.1 The Single Server Is a Process	456
17.3.2 The Single Server Is a Navigating Token	459
17.4 Sequential Consistency with a Server per Object	460
17.4.1 Structural View	460
17.4.2 The Object Managers Must Cooperate	461
17.4.3 An Algorithm Based on the OO Constraint	462
17.5 A Weaker Consistency Condition: Causal Consistency	464
17.5.1 Definition	464
17.5.2 A Simple Algorithm	466
17.5.3 The Case of a Single Object	467
17.6 A Hierarchy of Consistency Conditions	468

17.7 Summary	468
17.8 Bibliographic Notes	469
17.9 Exercises and Problems	470
Afterword	471
The Aim of This Book	471
Most Important Concepts, Notions, and Mechanisms Presented in This Book	471
How to Use This Book	473
From Failure-Free Systems to Failure-Prone Systems	474
A Series of Books	474
References	477
Index	495



<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