Contents

Preface xvii
Acknowledgments xxv

CHAPTER 1 Introduction 1
1.1 Rationale 2
1.1.1 The Traditional Approach to Teaching Computer Science 2
1.1.2 The Systems Approach Taken in This Book 4
1.2 The Significance of Networking and Distributed Systems in Modern Computing—A Brief Historical Perspective 6
1.3 Introduction to Distributed Systems 9
1.3.1 Benefits and Challenges of Distributed Systems 9
1.3.2 The Nature of Distribution 10
1.3.3 Software Architectures for Distributed Applications 11
1.3.4 Metrics for Measuring the Quality of Distributed Systems and Applications 11
1.3.5 Introduction to Transparency 12
1.4 Introduction to the Case Studies 14
1.4.1 The Main (Distributed Game) Case Study 14
1.4.2 The Additional Case Studies 16
1.5 Introduction to Supplementary Material and Exercises 16
1.5.1 In-Text Activities 17
1.6 The Workbenches Suite of Interactive Teaching and Learning Tools 18
1.6.1 Operating Systems Workbench 3.1 "Systems Programming Edition" 19
1.6.2 The Networking Workbench 3.1 "Systems Programming Edition" 19
1.6.3 Distributed Systems Workbench 3.1 "Systems Programming Edition" 19
1.7 Sample Code and Related Exercises 19
1.7.1 Source Code, in C++, C#, and Java 19
1.7.2 Application Development Exercises 20

CHAPTER 2 The Process View 21
2.1 Rationale and Overview 22
2.2 Processes 22
2.2.1 Basic Concepts 22
2.2.2 Creating a Process 23
2.3 Process Scheduling 28
2.3.1 Scheduling Concepts 32
2.4 Scheduling for Real-Time Systems 58
2.4.1 Limitations of General-Purpose Schedulers for Real-Time Systems 59

2.5 Specific Scheduling Algorithms and Variants, Used in Modern Operating Systems 70
2.6 Interprocess Communication 70
2.6.1 Introduction to Sockets 70
2.7 Threads: An Introduction 75
2.7.1 General Concepts 75
2.7.2 Thread Implementations 76
2.7.3 Thread Scheduling Approaches 77
2.7.4 Synchronous (Sequential) Versus Asynchronous (Concurrent) Thread Operation 78
2.7.5 Additional Complexity Accompanying Threading 83
2.7.6 A Multithreaded IPC Example 84
2.8 Other Roles of the Operating System 91
2.9 Use of Timers Within Programs 91
2.9.1 Use of Timers to Simulate Threadlike Behavior 93
2.10 Transparency from the Process Viewpoint 94
2.10.1 Concurrency Transparency 94
2.10.2 Migration Transparency 94
2.11 The Case Study from the Process Perspective 95
2.11.1 Scheduling Requirements 95
2.11.2 Use of Timers 95
2.11.3 Need for Threads 96
2.11.4 IPC, Ports, and Sockets 96
2.12 End-of-Chapter Exercises 97
2.12.1 Questions 97
2.12.2 Exercises with the Workbenches 98
2.12.3 Programming Exercises 103
2.12.4 Answers to End-of-Chapter Questions 104
2.12.5 List of In-text Activities 105
2.12.6 List of Accompanying Resources 105

CHAPTER 3 The Communication View 107
3.1 Rationale and Overview 108
3.2 The Communication View 109
3.2.1 Communication Basics 109
3.3 Communication Techniques 110
3.3.1 One-Way Communication 110
3.3.2 Request-Reply Communication 115
3.3.3 Two-Way Data Transfer 119
3.3.4 Addressing Methodologies 120

3.3.5 Remote Procedure Call 124
3.3.6 Remote Method Invocation 126
3.4 Layered Models of Communication 128
3.4.1 The OSI Model 131
3.4.2 The TCP/IP Model 132
3.5 The TCP/IP Suite 133
3.5.1 The IP 134
3.5.2 The TCP 136
3.5.3 The TCP Connection 138
3.5.4 The UDP 140
3.5.5 TCP and UDP Compared 141
3.5.6 Choosing Between TCP and UDP 141
3.6 Addresses 144
3.6.1 Flat Versus Hierarchical Addressing 144
3.6.2 Addresses in the Link Layer 145
3.6.3 Addresses in the Network Layer 145
3.6.4 Addresses in the Transport Layer (Ports) 148
3.6.5 Well-Known Ports 149
3.7 Sockets 151
3.7.1 The Socket API: An Overview 152
3.7.2 The Socket API: UDP Primitive Sequence 152
3.7.3 The Socket API: TCP Primitives Sequence 153
3.7.4 Binding (Process to Port) 158
3.8 Blocking and Nonblocking Socket Behaviors 161
3.8.1 Handling Nonblocking Socket Behavior 164
3.8.2 Communication Deadlock 166
3.9 Error Detection and Error Correction 168
3.9.1 A Brief Introduction to Error Detection and Error Correction Codes 168
3.10 Application-Specific Protocols 170
3.11 Integrating Communication with Business Logic 170
3.12 Techniques to Facilitate Components Locating Each Other 172
3.13 Transparency Requirements from the Communication Viewpoint 173
3.13.1 Logical and Physical Views of Systems 174
3.14 The Case Study from the Communication Perspective 174
3.15 End-of-Chapter Exercises 181
3.15.1 Questions 181
3.15.2 Exercises with the Workbenches 184
3.15.3 Programming Exercises 187
3.15.4 Answers to End-of-Chapter Questions 187

3.15.5 Answers/Outcomes of the Workbench Exercises 189
3.15.6 List of in-Text Activities 190
3.15.7 List of Accompanying Resources 190
Appendix Socket API Reference 191
A1. Socket 191
A2. Socket Options 193
A3. Socket Address Formats 194
A4. Setting a Socket to Operate in Blocking or Nonblocking IO Mode 195
A5. Bind 196
A6. Listen 196
A7. Connect 197
A8. Accept 197
A9. Send (Over a TCP Connection) 198
A10. Recv (Over a TCP Connection) 198
A11. SendTo (Send a UDP Datagram) 199
A12. Recvfrom (Receive a UDP Datagram) 200
A13. Shutdown 200
A14. Closesocket 201

CHAPTER 4 The Resource View 203

4.1 Rationale and Overview 204
4.2 The CPU as a Resource 204
4.3 Memory as a Resource for Communication 205
4.3.1 Memory Hierarchy 210
4.4 Memory Management 212
4.4.1 Virtual Memory 217
4.5 Resource Management 225
4.5.1 Static Versus Dynamic Allocation of Private Memory Resources 225
4.5.2 Shared Resources 227
4.5.3 Transactions 228
4.5.4 Locks 229
4.5.5 Deadlock 232
4.5.6 Replication of Resources 236
4.6 The Network as a Resource 238
4.6.1 Network Bandwidth 238
4.6.2 Data Compression Techniques 243
4.6.3 Message Format 246
4.6.4 Serialization 248
4.6.5 The Network as a Series of Links 251
4.6.6 Routers and Routing 254
4.6.7 Overheads of Communication 258
4.6.8 Recovery Mechanisms and Their Interplay with Network Congestion 259

4.7 Virtual Resources 262
4.7.1 Sockets 262
4.7.2 Ports 262
4.7.3 Network Addresses 263
4.7.4 Resource Names 263
4.8 Distributed Application Design Influences on Network Efficiency 264
4.9 Transparency from the Resource Viewpoint 265
4.9.1 Access Transparency 265
4.9.2 Location Transparency 265
4.9.3 Replication Transparency 265
4.9.4 Concurrency Transparency 265
4.9.5 Scaling Transparency and Performance Transparency 266
4.10 The Case Study from the Resource Perspective 266
4.11 End-of-Chapter Exercises 268
4.11.1 Questions 268
4.11.2 Exercises with the Workbenches 269
4.11.3 Programming Exercises 272
4.11.4 Answers to End-of-Chapter Questions 273
4.11.5 Answers/Outcomes of the Workbenches Exercises 274
4.11.6 List of In-Text Activities 276
4.11.7 List of Accompanying Resources 276

CHAPTER 5 The Architecture View 277
5.1 Rationale and Overview 279
5.2 The Architecture View 279
5.2.1 Separation of Concerns 280
5.2.2 Networking and Distribution 281
5.2.3 Complexity in Distributed Systems 282
5.2.4 Layered Architectures 283
5.2.5 Hierarchical Architectures 285
5.3 Heterogeneity 287
5.3.1 Definitions and Sources of Heterogeneity 287
5.3.2 Performance Heterogeneity 288
5.3.3 Platform Heterogeneity 289
5.3.4 Operating System Heterogeneity 290
5.3.5 Impact of Heterogeneity 290
5.3.6 Porting of Software 292
5.4 Hardware and System-Level Architectures 293
5.4.1 Tightly Coupled (Hardware) Systems 293
5.4.2 Loosely Coupled (Hardware) Systems 294
5.4.3 Parallel Processing 295

5.5 Software Architectures 296
5.5.1 Coupling Between Software Components 297
5.6 Taxonomy of Software Architecture Classes 302
5.6.1 Single-Tier Applications 302
5.6.2 Two-Tier Applications 302
5.6.3 Three-Tier Applications 304
5.6.4 Multitier Applications 304
5.7 Client-Server 305
5.7.1 Lifetimes of Client and Server 305
5.7.2 Active and Passive Sides of the Connection 306
5.7.3 The CS Architectural Model 306
5.7.4 Variants of the CS Model 308
5.7.5 Stateful Versus Stateless Services 309
5.7.6 Modular and Hierarchical CS Systems 310
5.8 Three-Tier and Multitier Architectures 312
5.9 Peer-to-Peer 323
5.9.1 Characteristics of Peer-to-Peer Applications 323
5.9.2 Complexity of Peer-to-Peer Connectivity 324
5.9.3 Exploring Peer-to-Peer Behavior 325
5.10 Distributed Objects 329
5.11 Middleware: Support for Software Architectures 331
5.11.1 Middleware Operation, an Overview 331
5.12 System Models of Collective Resources and Computation Resource Provision 333
5.12.1 Clusters 334
5.12.2 Grids 334
5.12.3 Data Centers 335
5.12.4 Clouds 335
5.13 Software Libraries 335
5.13.1 Software Library Case Example 338
5.13.2 Static Linking and Dynamic Linking 344
5.13.3 Language-Dependent Feature: The C/C++ Header File 346
5.14 Hardware Virtualization 348
5.14.1 Virtual Machines 349
5.14.2 Java Virtual Machine 350
5.15 Static and Dynamic Configurations 351
5.15.1 Static Configuration 351
5.15.2 Dynamic Configuration 352
5.15.3 Context Awareness 352
5.16 Nonfunctional Requirements of Distributed Applications 353

5.16.1 Replication 355
5.16.2 Semantics of Replication 358
5.16.3 An Implementation of Replication 359
5.17 The Relationship Between Distributed Applications and Networks 369
5.18 Transparency from the Architecture Viewpoint 371
5.19 The Case Study from the Architectural Perspective 372
5.19.1 Stateful Server Design 372
5.19.2 Separation of Concerns for the Game Components 373
5.19.3 Physical and Logical Architectures of the Game Application 374
5.19.4 Transparency Aspects of the Game 375
5.20 End-of-Chapter Exercises 376
5.20.1 Questions 376
5.20.2 Programming Exercises 377
5.20.3 Answers to end-of-Chapter Questions 378
5.20.4 List of In-Text Activities 380
5.20.5 List of Accompanying Resources 381
Appendix The Peer-to-Peer Application-Level Protocol Message Sequence 382

CHAPTER 6 Distributed Systems 383
6.1 Rationale and Overview 385
6.2 Transparency 385
6.2.1 Access Transparency 387
6.2.2 Location Transparency 388
6.2.3 Replication Transparency 389
6.2.4 Concurrency Transparency 393
6.2.5 Migration Transparency 398
6.2.6 Failure Transparency 398
6.2.7 Scaling Transparency 400
6.2.8 Performance Transparency 403
6.2.9 Distribution Transparency 404
6.2.10 Implementation Transparency 404
6.3 Common Services 404
6.4 Name Services 405
6.4.1 Name Service Operation 406
6.4.2 Directory Service 407
6.4.3 Challenges of Name Service Design and Implementation 413
6.5 Domain Name System (DNS) 414
6.5.1 Domain Namespace 415
6.5.2 DNS Implementation 417
6.5.3 DNS Name Servers: Authority and Delegation 420

6.5.4 Replication 421
6.5.5 Name Resolution in More Detail 422
6.5.6 Caching in DNS 423
6.5.7 Exploring Address Resolution 423
6.5.8 Reverse DNS Lookup 426
6.6 Time Services 427
6.6.1 Time Services 427
6.6.2 Physical Clock Synchronization 428
6.6.3 Logical Clocks and Synchronization 433
6.7 Election Algorithms 436
6.7.1 Operation Overview 437
6.7.2 The Bully Election Algorithm 439
6.7.3 The Ring Election Algorithm 440
6.7.4 Leader Preelection 440
6.7.5 Exploration with an Election Algorithm 441
6.8 Group Communications 448
6.8.1 Transparency Aspects of Group Communication 450
6.9 Notification Services 450
6.9.1 Publish-Subscribe Services 451
6.10 Middleware: Mechanism and Operation 452
6.11 Middleware Examples and Support Technologies 454
6.11.1 The Common Object Request Broker Architecture (CORBA) 454
6.11.2 Interface Definition Language (IDL) 462
6.11.3 Extensible Markup Language 464
6.11.4 JavaScript Object Notation (JSON) 465
6.11.5 Web Services and REST 466
6.11.6 The Simple Object Access Protocol (SOAP) 467
6.12 Deterministic and Nondeterministic Aspects of Distributed Systems 468
6.13 End of Chapter Exercises 470
6.13.1 Questions 470
6.13.2 Programming Exercises 470
6.13.3 Answers to End of Chapter Questions 472
6.13.4 List of In-Text Activities 473
6.13.5 List of Accompanying Resources 473

CHAPTER 7 Case Studies: Putting It All Together 475
7.1 Rationale and Overview 476
7.2 Introduction to the Use Cases 476
7.3 Case Study #1: Time Service Client (with Library) 477
7.3.1 Learning Outcomes Associated with the Case Study 477
7.3.2 Requirements Analysis 477

7.3.3 Architecture and Code Structure 478
7.3.4 Separation of Concerns 483
7.3.5 Coupling and Binding Between Components 484
7.3.6 Communication Aspects of the Design 484
7.3.7 Implementation 488
7.3.8 Testing 488
7.3.9 Transparency Aspects of the Use Case 489
7.3.10 Case Study Resources 491
7.4 Case Study #2: Event Notification Service 492
7.4.1 Learning Outcomes Associated with the Case Study 492
7.4.2 Requirements Analysis 493
7.4.3 Architecture and Code Structure 494
7.4.4 Separation of Concerns 495
7.4.5 Coupling and Binding Between Components 495
7.4.6 Communication Aspects of the Design 496
7.4.7 An Application Use Scenario to Illustrate the Use of the ENS 501
7.4.8 Testing 503
7.4.9 Transparency Aspects of the ENS 503
7.4.10 Case Study Resources 508
7.5 Good Design Practice for Distributed Applications 508
7.5.1 Requirements Analysis 509
7.5.2 Architectural Aspects 509
7.5.3 Communication Aspects 510
7.5.4 Reuse Code When Opportunities Arise 510
7.5.5 Create Libraries of Tested and Trusted Code 511
7.5.6 Testing Aspects 511
7.6 End of Chapter Programming Exercises 512
7.7 List of Accompanying Resources 513

Index 515