
Contents

PREFACE	xvi
1 INTRODUCTION TO DISTRIBUTED SYSTEMS	1
1.1 WHAT IS A DISTRIBUTED SYSTEM?	2
1.2 GOALS	3
1.2.1 Advantages of Distributed Systems over Centralized Systems	3
1.2.2 Advantages of Distributed Systems over Independent PCs	6
1.2.3 Disadvantages of Distributed Systems	6
1.3 HARDWARE CONCEPTS	8
1.3.1 Bus-Based Multiprocessors	10
1.3.2 Switched Multiprocessors	12
1.3.3 Bus-Based Multicomputers	13
1.3.4 Switched Multicomputers	14
1.4 SOFTWARE CONCEPTS	15
1.4.1 Network Operating Systems	16
1.4.2 True Distributed Systems	18
1.4.3 Multiprocessor Timesharing Systems	20
1.5 DESIGN ISSUES	22
1.5.1 Transparency	22
1.5.2 Flexibility	25
1.5.3 Reliability	27
1.5.4 Performance	28
1.5.5 Scalability	29
1.6 SUMMARY	31

2 COMMUNICATION IN DISTRIBUTED SYSTEMS

- 2.1 LAYERED PROTOCOLS 35
 - 2.1.1 The Physical Layer 38
 - 2.1.2 The Data Link Layer 38
 - 2.1.3 The Network Layer 40
 - 2.1.4 The Transport Layer 40
 - 2.1.5 The Session Layer 41
 - 2.1.6 The Presentation Layer 41
 - 2.1.7 The Application Layer 42
- 2.2 ASYNCHRONOUS TRANSFER MODE NETWORKS 42
 - 2.2.1 What Is Asynchronous Transfer Mode? 42
 - 2.2.2 The ATM Physical Layer 44
 - 2.2.3 The ATM Layer 45
 - 2.2.4 The ATM Adaptation Layer 46
 - 2.2.5 ATM Switching 47
 - 2.2.6 Some Implications of ATM for Distributed Systems 49
- 2.3 THE CLIENT-SERVER MODEL 50
 - 2.3.1 Clients and Servers 51
 - 2.3.2 An Example Client and Server 52
 - 2.3.3 Addressing 56
 - 2.3.4 Blocking versus Nonblocking Primitives 58
 - 2.3.5 Buffered versus Unbuffered Primitives 61
 - 2.3.6 Reliable versus Unreliable Primitives 63
 - 2.3.7 Implementing the Client-Server Model 65
- 2.4 REMOTE PROCEDURE CALL 68
 - 2.4.1 Basic RPC Operation 68
 - 2.4.2 Parameter Passing 72
 - 2.4.3 Dynamic Binding 77
 - 2.4.4 RPC Semantics in the Presence of Failures 80
 - 2.4.5 Implementation Issues 84
 - 2.4.6 Problem Areas 95
- 2.5 GROUP COMMUNICATION 99
 - 2.5.1 Introduction to Group Communication 99
 - 2.5.2 Design Issues 101
 - 2.5.3 Group Communication in ISIS 110
- 2.6 SUMMARY 114

3 SYNCHRONIZATION IN DISTRIBUTED SYSTEMS

118

- 3.1 CLOCK SYNCHRONIZATION 119
 - 3.1.1 Logical Clocks 120
 - 3.1.2 Physical Clocks 124
 - 3.1.3 Clock Synchronization Algorithms 127
 - 3.1.4 Use of Synchronized Clocks 132
- 3.2 MUTUAL EXCLUSION 134
 - 3.2.1 A Centralized Algorithm 134
 - 3.2.2 A Distributed Algorithm 135
 - 3.2.3 A Token Ring Algorithm 138
 - 3.2.4 A Comparison of the Three Algorithms 139
- 3.3 ELECTION ALGORITHMS 140
 - 3.3.1 The Bully Algorithm 141
 - 3.3.2 A Ring Algorithm 143
- 3.4 ATOMIC TRANSACTIONS 144
 - 3.4.1 Introduction to Atomic Transactions 144
 - 3.4.2 The Transaction Model 145
 - 3.4.3 Implementation 150
 - 3.4.4 Concurrency Control 154
- 3.5 DEADLOCKS IN DISTRIBUTED SYSTEMS 158
 - 3.5.1 Distributed Deadlock Detection 159
 - 3.5.2 Distributed Deadlock Prevention 163
- 3.6 SUMMARY 165

4 PROCESSES AND PROCESSORS IN DISTRIBUTED SYSTEMS

169

- 4.1 THREADS 169
 - 4.1.1 Introduction to Threads 170
 - 4.1.2 Thread Usage 171
 - 4.1.3 Design Issues for Threads Packages 174
 - 4.1.4 Implementing a Threads Package 178
 - 4.1.5 Threads and RPC 184
- 4.2 SYSTEM MODELS 186
 - 4.2.1 The Workstation Model 186
 - 4.2.2 Using Idle Workstations 189
 - 4.2.3 The Processor Pool Model 193
 - 4.2.4 A Hybrid Model 197
- 4.3 PROCESSOR ALLOCATION 197
 - 4.3.1 Allocation Models 197

CONTENTS

- 4.3.2 Design Issues for Processor Allocation Algorithms 199
- 4.3.3 Implementation Issues for Processor Allocation Algorithms 201
- 4.3.4 Example Processor Allocation Algorithms 203
- 4.4 SCHEDULING IN DISTRIBUTED SYSTEMS 210
- 4.5 FAULT TOLERANCE 212
 - 4.5.1 Component Faults 212
 - 4.5.2 System Failures 213
 - 4.5.3 Synchronous versus Asynchronous Systems 214
 - 4.5.4 Use of Redundancy 214
 - 4.5.5 Fault Tolerance Using Active Replication 215
 - 4.5.6 Fault Tolerance Using Primary-Backup 217
 - 4.5.7 Agreement in Faulty Systems 219
- 4.6 REAL-TIME DISTRIBUTED SYSTEMS 223
 - 4.6.1 What Is a Real-Time System? 223
 - 4.6.2 Design Issues 226
 - 4.6.3 Real-Time Communication 230
 - 4.6.4 Real-Time Scheduling 234
- 4.7 SUMMARY 240

DISTRIBUTED FILE SYSTEMS

245

- 5.1 DISTRIBUTED FILE SYSTEM DESIGN 246
 - 5.1.1 The File Service Interface 246
 - 5.1.2 The Directory Server Interface 248
 - 5.1.3 Semantics of File Sharing 253
- 5.2 DISTRIBUTED FILE SYSTEM IMPLEMENTATION 256
 - 5.2.1 File Usage 256
 - 5.2.2 System Structure 258
 - 5.2.3 Caching 262
 - 5.2.4 Replication 268
 - 5.2.5 An Example: Sun's Network File System 272
 - 5.2.6 Lessons Learned 278
- 5.3 TRENDS IN DISTRIBUTED FILE SYSTEMS 279
 - 5.3.1 New Hardware 280
 - 5.3.2 Scalability 282
 - 5.3.3 Wide Area Networking 283
 - 5.3.4 Mobile Users 284
 - 5.3.5 Fault Tolerance 284
 - 5.3.6 Multimedia 285
- 5.4 SUMMARY 285

6 DISTRIBUTED SHARED MEMORY

289

- 6.1 INTRODUCTION 290
- 6.2 WHAT IS SHARED MEMORY? 292
 - 6.2.1 On-Chip Memory 293
 - 6.2.2 Bus-Based Multiprocessors 293
 - 6.2.3 Ring-Based Multiprocessors 298
 - 6.2.4 Switched Multiprocessors 301
 - 6.2.5 NUMA Multiprocessors 308
 - 6.2.6 Comparison of Shared Memory Systems 312
- 6.3 CONSISTENCY MODELS 315
 - 6.3.1 Strict Consistency 315
 - 6.3.2 Sequential Consistency 317
 - 6.3.3 Causal Consistency 321
 - 6.3.4 PRAM Consistency and Processor Consistency 322
 - 6.3.5 Weak Consistency 325
 - 6.3.6 Release Consistency 327
 - 6.3.7 Entry Consistency 330
 - 6.3.8 Summary of Consistency Models 331
- 6.4 PAGE-BASED DISTRIBUTED SHARED MEMORY 333
 - 6.4.1 Basic Design 334
 - 6.4.2 Replication 334
 - 6.4.3 Granularity 336
 - 6.4.4 Achieving Sequential Consistency 337
 - 6.4.5 Finding the Owner 339
 - 6.4.6 Finding the Copies 342
 - 6.4.7 Page Replacement 343
 - 6.4.8 Synchronization 344
- 6.5 SHARED-VARIABLE DISTRIBUTED SHARED MEMORY 345
 - 6.5.1 Munin 346
 - 6.5.2 Midway 353
- 6.6 OBJECT-BASED DISTRIBUTED SHARED MEMORY 356
 - 6.6.1 Objects 356
 - 6.6.2 Linda 358
 - 6.6.3 Orca 365
- 6.7 COMPARISON 371
- 6.8 SUMMARY 372

7 CASE STUDY 1: AMOEBEA

- 7.1 INTRODUCTION TO AMOEBEA 376
 - 7.1.1 History of Amoeba 376
 - 7.1.2 Research Goals 377
 - 7.1.3 The Amoeba System Architecture 378
 - 7.1.4 The Amoeba Microkernel 380
 - 7.1.5 The Amoeba Servers 382
- 7.2 OBJECTS AND CAPABILITIES IN AMOEBEA 384
 - 7.2.1 Capabilities 384
 - 7.2.2 Object Protection 385
 - 7.2.3 Standard Operations 387
- 7.3 PROCESS MANAGEMENT IN AMOEBEA 388
 - 7.3.1 Processes 388
 - 7.3.2 Threads 391
- 7.4 MEMORY MANAGEMENT IN AMOEBEA 392
 - 7.4.1 Segments 392
 - 7.4.2 Mapped Segments 393
- 7.5 COMMUNICATION IN AMOEBEA 393
 - 7.5.1 Remote Procedure Call 394
 - 7.5.2 Group Communication in Amoeba 398
 - 7.5.3 The Fast Local Internet Protocol 407
- 7.6 THE AMOEBEA SERVERS 415
 - 7.6.1 The Bullet Server 415
 - 7.6.2 The Directory Server 420
 - 7.6.3 The Replication Server 425
 - 7.6.4 The Run Server 425
 - 7.6.5 The Boot Server 427
 - 7.6.6 The TCP/IP Server 427
 - 7.6.7 Other Servers 428
- 7.7 SUMMARY 428

8 CASE STUDY 2: MACH

- 8.1 INTRODUCTION TO MACH 431
 - 8.1.1 History of Mach 431
 - 8.1.2 Goals of Mach 433
 - 8.1.3 The Mach Microkernel 433
 - 8.1.4 The Mach BSD UNIX Server 435

- 8.2 PROCESS MANAGEMENT IN MACH 436
 - 8.2.1 Processes 436
 - 8.2.2 Threads 439
 - 8.2.3 Scheduling 442
- 8.3 MEMORY MANAGEMENT IN MACH 445
 - 8.3.1 Virtual Memory 446
 - 8.3.2 Memory Sharing 449
 - 8.3.3 External Memory Managers 452
 - 8.3.4 Distributed Shared Memory in Mach 456
- 8.4 COMMUNICATION IN MACH 457
 - 8.4.1 Ports 457
 - 8.4.2 Sending and Receiving Messages 464
 - 8.4.3 The Network Message Server 469
- 8.5 UNIX EMULATION IN MACH 471
- 8.6 SUMMARY 472

9 CASE STUDY 3: CHORUS

475

- 9.1 INTRODUCTION TO CHORUS 475
 - 9.1.1 History of Chorus 476
 - 9.1.2 Goals of Chorus 477
 - 9.1.3 System Structure 478
 - 9.1.4 Kernel Abstractions 479
 - 9.1.5 Kernel Structure 481
 - 9.1.6 The UNIX Subsystem 483
 - 9.1.7 The Object-Oriented Subsystem 483
- 9.2 PROCESS MANAGEMENT IN CHORUS 483
 - 9.2.1 Processes 484
 - 9.2.2 Threads 485
 - 9.2.3 Scheduling 486
 - 9.2.4 Traps, Exceptions, and Interrupts 487
 - 9.2.5 Kernel Calls for Process Management 488
- 9.3 MEMORY MANAGEMENT IN CHORUS 490
 - 9.3.1 Regions and Segments 490
 - 9.3.2 Mappers 491
 - 9.3.3 Distributed Shared Memory 492
 - 9.3.4 Kernel Calls for Memory Management 493

- 9.4 COMMUNICATON IN CHORUS 495
 - 9.4.1 Messages 495
 - 9.4.2 Ports 495
 - 9.4.3 Communication Operations 496
 - 9.4.4 Kernel Calls for Communication 498
- 9.5 UNIX EMULATION IN CHORUS 499
 - 9.5.1 Structure of a UNIX Process 500
 - 9.5.2 Extensions to UNIX 500
 - 9.5.3 Implementation of UNIX on Chorus 501
- 9.6 COOL: AN OBJECT-ORIENTED SUBSYSTEM 507
 - 9.6.1 The COOL Architecture 507
 - 9.6.2 The COOL Base Layer 507
 - 9.6.3 The COOL Generic Runtime System 509
 - 9.6.4 The Language Runtime System 509
 - 9.6.5 Implementation of COOL 510
- 9.7 COMPARISON OF AMOEBA, MACH, AND CHORUS 510
 - 9.7.1 Philosophy 511
 - 9.7.2 Objects 512
 - 9.7.3 Processes 513
 - 9.7.4 Memory Model 514
 - 9.7.5 Communication 515
 - 9.7.6 Servers 516
- 9.8 SUMMARY 517

10 CASE STUDY 4: DCE

520

- 10.1 INTRODUCTION TO DCE 520
 - 10.1.1 History of DCE 520
 - 10.1.2 Goals of DCE 521
 - 10.1.3 DCE Components 522
 - 10.1.4 Cells 525
- 10.2 THREADS 527
 - 10.2.1 Introduction to DCE Threads 527
 - 10.2.2 Scheduling 529
 - 10.2.3 Synchronization 530
 - 10.2.4 Thread Calls 531
- 10.3 REMOTE PROCEDURE CALL 535
 - 10.3.1 Goals of DCE RPC 535
 - 10.3.2 Writing a Client and a Server 536
 - 10.3.3 Binding a Client to a Server 538
 - 10.3.4 Performing an RPC 539

- 10.4 TIME SERVICE 540
 - 10.4.1 DTS Time Model 541
 - 10.4.2 DTS Implementation 543
- 10.5 DIRECTORY SERVICE 544
 - 10.5.1 Names 546
 - 10.5.2 The Cell Directory Service 547
 - 10.5.3 The Global Directory Service 549
- 10.6 SECURITY SERVICE 554
 - 10.6.1 Security Model 555
 - 10.6.2 Security Components 557
 - 10.6.3 Tickets and Authenticators 558
 - 10.6.4 Authenticated RPC 559
 - 10.6.5 ACLs 562
- 10.7 DISTRIBUTED FILE SYSTEM 564
 - 10.7.1 DFS Interface 565
 - 10.7.2 DFS Components in the Server Kernel 566
 - 10.7.3 DFS Components in the Client Kernel 569
 - 10.7.4 DFS Components in User Space 571
- 10.8 SUMMARY 573

11 BIBLIOGRAPHY AND SUGGESTED READINGS

577

- 11.1 SUGGESTED READINGS 577
- 11.2 ALPHABETICAL BIBLIOGRAPHY 584

INDEX

603