[illegible]

Pearson Education

Distributed Operating Systems

Andrew S. Tanenbaum

3125995

This edition is manufactured in India and is authoriz
for sale only in India, Bangladesh, Bhutan, Pakistan, N
Sri Lanka and the Maldives. Circulation of this edi
outside of these territories is UNAUTHORIZED.

Distributed Operating Systems

Andrew S. Tanenbaum

Vrije Universiteit
Amsterdam, The Netherlands

2037
[illegible]
Department of Computer Science
UNIVERSITY OF KARACHI
1-7-08

PEARSON
Education

Contents

PREFACE

1 INTRODUCTION TO DISTRIBUTED SYSTEMS

1.1 WHAT IS A DISTRIBUTED SYSTEM? 20

1.2 GOALS 21

1.2.1 Advantages of Distributed Systems over Centralized System
1.2.2 Advantages of Distributed Systems over Independent PCs
1.2.3 Disadvantages of Distributed Systems 24

1.3 HARDWARE CONCEPTS 26

1.3.1 Bus-Based Multiprocessors 28
1.3.2 Switched Multiprocessors 30
1.3.3 Bus-Based Multicomputers 31
1.3.4 Switched Multicomputers 32

1.4 SOFTWARE CONCEPTS 33

1.4.1 Network Operating Systems 34
1.4.2 True Distributed Systems 36
1.4.3 Multiprocessor Timesharing Systems 38

1.5 DESIGN ISSUES 40

1.5.1 Transparency 40
1.5.2 Flexibility 43
1.5.3 Reliability 45
1.5.4 Performance 46
1.5.5 Scalability 47

1.6 SUMMARY 49

2 COMMUNICATION IN DISTRIBUTED SYSTEMS 53

2.1 LAYERED PROTOCOLS 56
2.1.1 The Physical Layer 56
2.1.2 The Data Link Layer 58
2.1.3 The Network Layer 58
2.1.4 The Transport Layer 59
2.1.5 The Session Layer 59
2.1.6 The Presentation Layer 60
2.1.7 The Application Layer 60
2.2 ASYNCHRONOUS TRANSFER MODE NETWORKS 60
2.2.1 What Is Asynchronous Transfer Mode? 60
2.2.2 The ATM Physical Layer 62
2.2.3 The ATM Layer 63
2.2.4 The ATM Adaptation Layer 64
2.2.5 ATM Switching 65
2.2.6 Some Implications of ATM for Distributed Systems 67
2.3 THE CLIENT-SERVER MODEL 68
2.3.1 Clients and Servers 69
2.3.2 An Example Client and Server 70
2.3.3 Addressing 74
2.3.4 Blocking versus Nonblocking Primitives 76
2.3.5 Buffered versus Unbuffered Primitives 79
2.3.6 Reliable versus Unreliable Primitives 81
2.3.7 Implementing the Client-Server Model 83
2.4 REMOTE PROCEDURE CALL 86
2.4.1 Basic RPC Operation 86
2.4.2 Parameter Passing 90
2.4.3 Dynamic Binding 95
2.4.4 RPC Semantics in the Presence of Failures 98
2.4.5 Implementation Issues 102
2.4.6 Problem Areas 113
2.5 GROUP COMMUNICATION 117
2.5.1 Introduction to Group Communication 117
2.5.2 Design Issues 119
2.5.3 Group Communication in ISIS 128
2.6 SUMMARY 132

3 SYNCHRONIZATION IN DISTRIBUTED SYSTEMS 136

3.1 CLOCK SYNCHRONIZATION 137

3.1.1 Logical Clocks 138
3.1.2 Physical Clocks 142
3.1.3 Clock Synchronization Algorithms 145
3.1.4 Use of Synchronized Clocks 150

3.2 MUTUAL EXCLUSION 152

3.2.1 A Centralized Algorithm 152
3.2.2 A Distributed Algorithm 153
3.2.3 A Token Ring Algorithm 156
3.2.4 A Comparison of the Three Algorithms 157

3.3 ELECTION ALGORITHMS 158

3.3.1 The Bully Algorithm 159
3.3.2 A Ring Algorithm 161

3.4 ATOMIC TRANSACTIONS 162

3.4.1 Introduction to Atomic Transactions 162
3.4.2 The Transaction Model 163
3.4.3 Implementation 168
3.4.4 Concurrency Control 172

3.5 DEADLOCKS IN DISTRIBUTED SYSTEMS 176

3.5.1 Distributed Deadlock Detection 177
3.5.2 Distributed Deadlock Prevention 181

3.6 SUMMARY 183

4 PROCESSES AND PROCESSORS IN DISTRIBUTED SYSTEMS 187

4.1 THREADS 187

4.1.1 Introduction to Threads 188
4.1.2 Thread Usage 189
4.1.3 Design Issues for Threads Packages 192
4.1.4 Implementing a Threads Package 196
4.1.5 Threads and RPC 202

4.2 SYSTEM MODELS 204

4.2.1 The Workstation Model 204
4.2.2 Using Idle Workstations 207
4.2.3 The Processor Pool Model 211
4.2.4 A Hybrid Model 215

4.3 PROCESSOR ALLOCATION 215

4.3.1 Allocation Models 215

4.3.2 Design Issues for Processor Allocation Algorithms 217
4.3.3 Implementation Issues for Processor Allocation Algorithms 219
4.3.4 Example Processor Allocation Algorithms 221
4.4 SCHEDULING IN DISTRIBUTED SYSTEMS 228
4.5 FAULT TOLERANCE 230
4.5.1 Component Faults 230
4.5.2 System Failures 231
4.5.3 Synchronous versus Asynchronous Systems 232
4.5.4 Use of Redundancy 232
4.5.5 Fault Tolerance Using Active Replication 233
4.5.6 Fault Tolerance Using Primary-Backup 235
4.5.7 Agreement in Faulty Systems 237
4.6 REAL-TIME DISTRIBUTED SYSTEMS 241
4.6.1 What Is a Real-Time System? 241
4.6.2 Design Issues 244
4.6.3 Real-Time Communication 248
4.6.4 Real-Time Scheduling 252
4.7 SUMMARY 258

5 DISTRIBUTED FILE SYSTEMS

5.1 DISTRIBUTED FILE SYSTEM DESIGN 264
5.1.1 The File Service Interface 264
5.1.2 The Directory Server Interface 266
5.1.3 Semantics of File Sharing 271
5.2 DISTRIBUTED FILE SYSTEM IMPLEMENTATION 274
5.2.1 File Usage 274
5.2.2 System Structure 276
5.2.3 Caching 280
5.2.4 Replication 286
5.2.5 An Example: Sun's Network File System 290
5.2.6 Lessons Learned 296
5.3 TRENDS IN DISTRIBUTED FILE SYSTEMS 297
5.3.1 New Hardware 298
5.3.2 Scalability 300
5.3.3 Wide Area Networking 301
5.3.4 Mobile Users 302
5.3.5 Fault Tolerance 302
5.3.6 Multimedia 303
5.4 SUMMARY 303

6 DISTRIBUTED SHARED MEMORY

6.1 INTRODUCTION 308

6.2 WHAT IS SHARED MEMORY? 310

6.2.1 On-Chip Memory 311
6.2.2 Bus-Based Multiprocessors 311
6.2.3 Ring-Based Multiprocessors 316
6.2.4 Switched Multiprocessors 319
6.2.5 NUMA Multiprocessors 326
6.2.6 Comparison of Shared Memory Systems 330

6.3 CONSISTENCY MODELS 333

6.3.1 Strict Consistency 333
6.3.2 Sequential Consistency 335
6.3.3 Causal Consistency 339
6.3.4 PRAM Consistency and Processor Consistency 340
6.3.5 Weak Consistency 343
6.3.6 Release Consistency 345
6.3.7 Entry Consistency 348
6.3.8 Summary of Consistency Models 349

6.4 PAGE-BASED DISTRIBUTED SHARED MEMORY 351

6.4.1 Basic Design 352
6.4.2 Replication 352
6.4.3 Granularity 354
6.4.4 Achieving Sequential Consistency 355
6.4.5 Finding the Owner 357
6.4.6 Finding the Copies 360
6.4.7 Page Replacement 361
6.4.8 Synchronization 362

6.5 SHARED-VARIABLE DISTRIBUTED SHARED MEMORY 36

6.5.1 Munin 364
6.5.2 Midway 371

6.6 OBJECT-BASED DISTRIBUTED SHARED MEMORY 374

6.6.1 Objects 374
6.6.2 Linda 376
6.6.3 Orca 383

6.7 COMPARISON 389

7 CASE STUDY 1: AMOEBA

7.1 INTRODUCTION TO AMOEBA 394
7.1.1 History of Amoeba 394
7.1.2 Research Goals 395
7.1.3 The Amoeba System Architecture 396
7.1.4 The Amoeba Microkernel 398
7.1.5 The Amoeba Servers 400
7.2 OBJECTS AND CAPABILITIES IN AMOEBA 402
7.2.1 Capabilities 402
7.2.2 Object Protection 403
7.2.3 Standard Operations 405
7.3 PROCESS MANAGEMENT IN AMOEBA 406
7.3.1 Processes 406
7.3.2 Threads 409
7.4 MEMORY MANAGEMENT IN AMOEBA 410
7.4.1 Segments 410
7.4.2 Mapped Segments 411
7.5 COMMUNICATION IN AMOEBA 411
7.5.1 Remote Procedure Call 412
7.5.2 Group Communication in Amoeba 416
7.5.3 The Fast Local Internet Protocol 425
7.6 THE AMOEBA SERVERS 433
7.6.1 The Bullet Server 433
7.6.2 The Directory Server 438
7.6.3 The Replication Server 443
7.6.4 The Run Server 443
7.6.5 The Boot Server 445
7.6.6 The TCP/IP Server 445
7.6.7 Other Servers 446
7.7 SUMMARY 446

8 CASE STUDY 2: MACH

8.1 INTRODUCTION TO MACH 449
8.1.1 History of Mach 449
8.1.2 Goals of Mach 451
8.1.3 The Mach Microkernel 451
8.1.4 The Mach BSD UNIX S

8.2 PROCESS MANAGEMENT IN MACH 454
- 8.2.1 Processes 454
- 8.2.2 Threads 457
- 8.2.3 Scheduling 460

8.3 MEMORY MANAGEMENT IN MACH 463
- 8.3.1 Virtual Memory 464
- 8.3.2 Memory Sharing 467
- 8.3.3 External Memory Managers 470
- 8.3.4 Distributed Shared Memory in Mach 474

8.4 COMMUNICATION IN MACH 475
- 8.4.1 Ports 475
- 8.4.2 Sending and Receiving Messages 482
- 8.4.3 The Network Message Server 487

8.5 UNIX EMULATION IN MACH 489

8.6 SUMMARY 490

9 CASE STUDY 3: CHORUS 49

9.1 INTRODUCTION TO CHORUS 493
- 9.1.1 History of Chorus 494
- 9.1.2 Goals of Chorus 495
- 9.1.3 System Structure 496
- 9.1.4 Kernel Abstractions 497
- 9.1.5 Kernel Structure 499
- 9.1.6 The UNIX Subsystem 501
- 9.1.7 The Object-Oriented Subsystem 501

9.2 PROCESS MANAGEMENT IN CHORUS 501
- 9.2.1 Processes 502
- 9.2.2 Threads 503
- 9.2.3 Scheduling 504
- 9.2.4 Traps, Exceptions, and Interrupts 505
- 9.2.5 Kernel Calls for Process Management 506

9.3 MEMORY MANAGEMENT IN CHORUS 508
- 9.3.1 Regions and Segments 508
- 9.3.2 Mappers 509
- 9.3.3 Distributed Shared Memory 510
- 9.3.4 Kernel Calls for Memory Management 511

9.4 COMMUNICATON IN CHORUS 513

9.4.1 Messages 513
9.4.2 Ports 513
9.4.3 Communication Operations 514
9.4.4 Kernel Calls for Communication 516

9.5 UNIX EMULATION IN CHORUS 517

9.5.1 Structure of a UNIX Process 518
9.5.2 Extensions to UNIX 518
9.5.3 Implementation of UNIX on Chorus 519

9.6 COOL: AN OBJECT-ORIENTED SUBSYSTEM 525

9.6.1 The COOL Architecture 525
9.6.2 The COOL Base Layer 525
9.6.3 The COOL Generic Runtime System 527
9.6.4 The Language Runtime System 527
9.6.5 Implementation of COOL 528

9.7 COMPARISON OF AMOEBA, MACH, AND CHORUS 528

9.7.1 Philosophy 529
9.7.2 Objects 530
9.7.3 Processes 531
9.7.4 Memory Model 532
9.7.5 Communication 533
9.7.6 Servers 534

9.8 SUMMARY 535

10 CASE STUDY 22: DCE **538**

10.1 INTRODUCTION TO DCE 538

10.1.1 History of DCE 538
10.1.2 Goals of DCE 539
10.1.3 DCE Components 540
10.1.4 Cells 543

10.2 THREADS 545

10.2.1 Introduction to DCE Threads 545
10.2.2 Scheduling 547
10.2.3 Synchronization 548
10.2.4 Thread Calls 549

10.3 REMOTE PROCEDURE CALL 553

10.3.1 Goals of DCE RPC 553
10.3.2 Writing a Client and a Server 554
10.3.3 Binding a Client to a Server 556
10.3.4 Performing an RPC 557

10.4 TIME SERVICE 558
10.4.1 DTS Time Model 559
10.4.2 DTS Implementation 561
10.5 DIRECTORY SERVICE 562
10.5.1 Names 564
10.5.2 The Cell Directory Service 565
10.5.3 The Global Directory Service 567
10.6 SECURITY SERVICE 572
10.6.1 Security Model 573
10.6.2 Security Components 575
10.6.3 Tickets and Authenticators 576
10.6.4 Authenticated RPC 577
10.6.5 ACLs 580
10.7 DISTRIBUTED FILE SYSTEM 582
10.7.1 DFS Interface 583
10.7.2 DFS Components in the Server Kernel 584
10.7.3 DFS Components in the Client Kernel 587
10.7.4 DFS Components in User Space 589
10.8 SUMMARY 591

INDEX

Preface

With the publication of *Distributed Operating Systems* I have now completed my trilogy on operating systems. The three volumes of this trilogy are:

- *Operating Systems: Design and Implementation*
- *Distributed Operating Systems*
- *Modern Operating Systems*

The three volumes are not completely independent, however. For schools having a two-course sequence in operating systems (or an undergraduate course plus a graduate course), one possible choice is to use *Operating Systems: Design and Implementation* in the first course and *Distributed Operating Systems* in the second one.

The former book treats the standard principles of single-processor systems, including processes, synchronization, I/O, deadlocks, memory management, file systems, security, and so on. It also illustrates these principles in great detail through the use of MINIX, a UNIX-clone whose source listing is given in an appendix. MINIX is available on diskette from Prentice Hall for the IBM PC (8088 and up), Atari, Amiga, Macintosh, and SPARC processors.

The latter book (this one), covers distributed operating systems in detail, including communication, synchronization, processes, file systems, and memory management, but this time in the context of distributed systems. Four examples of distributed systems are given in great detail: Amoeba, Mach, Chorus, and DCE. Amoeba is available for free to universities for educational use. It runs