Contents

Preface xvi

Chapter 1

Fundamentals 1

1.1 Requirements for Computer Networking 2
- **1.1.1** Connectivity: Node, Link, Path 2
- **Historical Evolution: Link Standards** 4
- **Historical Evolution: ATM Faded** 5
- **1.1.2** Scalability: Number of Nodes 6
- **1.1.3** Resource Sharing 7
- **Principle in Action: Datacom vs. Telecom** 10

1.2 Underlying Principles 10
- **1.2.1** Performance Measures 10
- **Principle in Action: Little's Result** 13
- **1.2.2** Operations at Control Plane 14
- **1.2.3** Operations at Data Plane 16
- **1.2.4** Interoperability 20

1.3 The Internet Architecture 21
- **1.3.1** Solutions to Connectivity 22
- **Principle in Action: Constantly Challenged Statelessness** 23
- **1.3.2** Solutions to Scalability 25
- **1.3.3** Solutions to Resource Sharing 27
- **1.3.4** Control-Plane and Data-Plane Operations 29
- **Principle in Action: Flavors of the Internet Architecture** 31

1.4 Open Source Implementations 32
- **1.4.1** Open vs. Closed 32
- **1.4.2** Software Architecture in Linux Systems 33
- **1.4.3** Linux Kernel 36
- **1.4.4** Clients and Daemon Servers 36
- **1.4.5** Interface Drivers 37
- **1.4.6** Device Controllers 38

1.5 Book Roadmap: A Packet's Life 39
- **1.5.1** Packet Data Structure: `sk_buff` 39
- **1.5.2** A Packet's Life in a Web Server 40
- **1.5.3** A Packet's Life in a Gateway 41
- **Performance Matters: From Socket to Driver within a Server** 42
- **Performance Matters: From Input Port to Output Port within a Router** 44
- **Principle in Action: A Packet's Life in the Internet** 45

1.6 Summary 46
- **The Evolving Hourglass** 47
- **Common Pitfalls** 47
- **Further Readings** 49
- **Frequently Asked Questions** 50
- **Exercises** 51

Chapter 2

Physical Layer 54

2.1 General Issues 55
- **2.1.1** Data and Signal: Analog or Digital 55
- **Principle in Action: Nyquist Theorem vs. Shannon Theorem** 57
- **2.1.2** Transmission and Reception Flows 59
- **2.1.3** Transmission: Line Coding and Digital Modulation 61
- **2.1.4** Transmission Impairments 62
- **Historical Evolution: Software Defined Radio** 63

2.2 Medium 65
- **2.2.1** Wired Medium 65
- **2.2.2** Wireless Medium 68

2.3 Information Coding and Baseband Transmission 70
- **2.3.1** Source and Channel Coding 71
- **2.3.2** Line Coding 72

Open Source Implementation 2.1: 8B/10B Encoder 82

2.4 Digital Modulation and Multiplexing 84

2.4.1 Passband Modulation 84

2.4.2 Multiplexing 92

2.5 Advanced Topics 96

2.5.1 Spread Spectrum 96

2.5.2 Single-Carrier vs. Multiple-Carrier 106

2.5.3 Multiple Inputs, Multiple Outputs (MIMO) 109

Open Source Implementation 2.2: IEEE 802.11a Transmitter with OFDM 112

Historical Evolution: Cellular Standards 116

Historical Evolution: LTE-Advanced vs. IEEE 802.16m 117

2.6 Summary 118

Common Pitfalls 119

Further Readings 120

Frequently Asked Questions 122

Exercises 123

Chapter 3

Link Layer 125

3.1 General Issues 126

3.1.1 Framing 127

3.1.2 Addressing 129

3.1.3 Error Control and Reliability 130

Principle in Action: CRC or Checksum? 133

Principle in Action: Error Correction Code 133

Open Source Implementation 3.1: Checksum 134

Open Source Implementation 3.2: Hardware CRC-32 135

3.1.4 Flow Control 137

3.1.5 Medium Access Control 138

3.1.6 Bridging 139

3.1.7 Link-Layer Packet Flows 139

Open Source Implementation 3.3: Link-Layer Packet Flows in Call Graphs 139

3.2 Point-to-Point Protocol 142

3.2.1 High-Level Data Link Control (HDLC) 143

3.2.2 Point-to-Point Protocol (PPP) 145

3.2.3 Internet Protocol Control Protocol (IPCP) 147

Open Source Implementation 3.4: PPP Drivers 148

3.2.4 PPP over Ethernet (PPPoE) 149

3.3 Ethernet (IEEE 802.3) 150

3.3.1 Ethernet Evolution: A Big Picture 150

Historical Evolution: Competitors to Ethernet 153

3.3.2 The Ethernet MAC 153

Open Source Implementation 3.5: CSMA/CD 161

Historical Evolution: Power-Line Networking: HomePlug 166

3.3.3 Selected Topics in Ethernet 167

Historical Evolution: Backbone Networking: SONET/SDH and MPLS 169

Historical Evolution: First-Mile Networking: xDSL and Cable Modem 171

3.4 Wireless Links 171

3.4.1 IEEE 802.11 Wireless LAN 172

Principle in Action: Why Not CSMA/CD in WLAN? 175

Open Source Implementation 3.6: IEEE 802.11 MAC Simulation with NS-2 177

3.4.2 Bluetooth Technology 182

3.4.3 WiMAX Technology 186

Historical Evolution: Comparing Bluetooth and IEEE 802.11 187

Historical Evolution: Comparing 3G, LTE, and WiMAX 190

3.5 Bridging 191

3.5.1 Self-Learning 191

Historical Evolution: Cut-Through vs. Store-and-Forward 193

Open Source Implementation 3.7: Self-Learning Bridging 194

3.5.2 Spanning Tree Protocol 196

Open Source Implementation 3.8: Spanning Tree 198

3.5.3 Virtual LAN 200

Principle in Action: VLAN vs. Subnet 201

3.6 Device Drivers of a Network Interface 204

3.6.1 Concepts of Device Drivers 204

3.6.2 Communicating with Hardware in a Linux Device Driver 205

Open Source Implementation 3.9: Probing I/O Ports, Interrupt Handling, and DMA 207

Open Source Implementation 3.10: The Network Device Driver in Linux 211

Performance Matters: Interrupt and DMA Handling within a Driver 214

Historical Evolution: Standard Interfaces for Drivers 215

3.7 Summary 215

Common Pitfalls 216

Further Readings 218

Frequently Asked Questions 220

Exercises 221

Chapter 4

Internet Protocol Layer 223

4.1 General Issues 224

4.1.1 Connectivity Issues 224

4.1.2 Scalability Issues 225

Principle in Action: Bridging vs. Routing 226

4.1.3 Resource Sharing Issues 227

4.1.4 Overview of IP-Layer Protocols and Packet Flows 228

Open Source Implementation 4.1: IP-Layer Packet Flows in Call Graphs 229

Performance Matters: Latency within the IP Layer 230

4.2 Data-Plane Protocols: Internet Protocol 231

4.2.1 Internet Protocol Version 4 232

Open Source Implementation 4.2: IPv4 Packet Forwarding 238

Performance Matters: Lookup Time at Routing Cache and Table 241

Open Source Implementation 4.3: IPv4 Checksum in Assembly 244

Open Source Implementation 4.4: IPv4 Fragmentation 246

4.2.2 Network Address Translation (NAT) 248

Principle in Action: Different Types of NAT 250

Principle in Action: Messy ALG in NAT 253

Open Source Implementation 4.5: NAT 253

Performance Matters: CPU Time of NAT Execution and Others 258

4.3 Internet Protocol Version 6 259

Historical Evolution: NAT vs. IPv6 259

4.3.1 IPv6 Header Format 260

4.3.2 IPv6 Extension Header 261

4.3.3 Fragmentation in IPv6 262

4.3.4 IPv6 Address Notation 263

4.3.5 IPv6 Address Space Assignment 264

4.3.6 Autoconfiguration 266

4.3.7 Transition from IPv4 to IPv6 266

4.4 Control-Plane Protocols: Address Management 267

4.4.1 Address Resolution Protocol 268

Open Source Implementation 4.6: ARP 269

4.4.2 Dynamic Host Configuration 271

Open Source Implementation 4.7: DHCP 275

4.5 Control Plane Protocols: Error Reporting 277

4.5.1 ICMP Protocol 277

Open Source Implementation 4.8: ICMP 280

4.6 Control Plane Protocols: Routing 283

4.6.1 Routing Principles 283

Principle in Action: Optimal Routing 285

4.6.2 Intra-Domain Routing 294

Open Source Implementation 4.9: RIP 297

4.6.3 Inter-Domain Routing 305

Open Source Implementation 4.10: OSPF 307

Performance Matters: Computation Overhead of Routing Daemons 309

Open Source Implementation 4.11: BGP 312

4.7 Multicast Routing 313

4.7.1 Shifting Complexity to Routers 313

4.7.2 Group Membership Management 315

4.7.3 Multicast Routing Protocols 316

Principle in Action: When the Steiner Tree Differs from the Least-Cost-Path Tree 318

4.7.4 Inter-Domain Multicast 325

Principle in Action: IP Multicast or Application Multicast? 326

Open Source Implementation 4.12: Mrouted 326

4.8 Summary 328

Common Pitfalls 329

Further Readings 330

Frequently Asked Questions 333

Exercises 335

Chapter 5

Transport Layer 339

5.1 General Issues 340

5.1.1 Node-to-Node vs. End-to-End 341

5.1.2 Error Control and Reliability 342

5.1.3 Rate Control: Flow Control and Congestion Control 343

5.1.4 Standard Programming Interfaces 344

5.1.5 Transport-Layer Packet Flows 344

Open Source Implementation 5.1: Transport-Layer Packet Flows in Call Graphs 344

5.2 Unreliable Connectionless Transfer: UDP 347

5.2.1 Header Format 347

5.2.2 Error Control: Per-Segment Checksum 348

Open Source Implementation 5.2: UDP and TCP Checksum 349

5.2.3 Carrying Unicast/Multicast Real-Time Traffic 350

5.3 Reliable Connection-Oriented Transfer: TCP 351

5.3.1 Connection Management 351

5.3.2 Reliability of Data Transfers 356

5.3.3 TCP Flow Control 358

Open Source Implementation 5.3: TCP Sliding-Window Flow Control 362

5.3.4 TCP Congestion Control 363

Historical Evolution: Statistics of TCP Versions 364

Open Source Implementation 5.4: TCP Slow Start and Congestion Avoidance 367

Principle in Action: TCP Congestion Control Behaviors 370

5.3.5 TCP Header Format 371

5.3.6 TCP Timer Management 374

Open Source Implementation 5.5: TCP Retransmission Timer 375

Open Source Implementation 5.6: TCP Persist Timer and Keepalive Timer 377

5.3.7 TCP Performance Problems and Enhancements 378

Historical Evolution: Multiple-Packet-Loss Recovery in NewReno, SACK, FACK, and Vegas 385

Principle in Action: TCP for the Networks with Large Bandwidth-Delay Product 390

5.4 Socket Programming Interfaces 391

5.4.1 Socket 391

5.4.2 Binding Applications through UDP and TCP 391

Principle in Action: SYN Flooding and Cookies 394

Open Source Implementation 5.7: Socket Read/Write Inside Out 394

Performance Matters: Interrupt and Memory Copy at Socket 397

5.4.3 Bypassing UDP and TCP 399

Open Source Implementation 5.8: Bypassing the Transport Layer 399

Open Source Implementation 5.9: Making Myself Promiscuous 401

Open Source Implementation 5.10: Linux Socket Filter 403

5.5 Transport Protocols for Real-Time Traffic 404

5.5.1 Real-Time Requirements 404

Principle in Action: Streaming: TCP or UDP? 406

5.5.2 Standard Data-Plane Protocol: RTP 407

5.5.3 Standard Control-Plane Protocol: RTCP 408

Historical Evolution: RTP Implementation Resources 409

5.6 Summary 410

Common Pitfalls 410

Further Readings 411

Frequently Asked Questions 412

Exercises 413

Chapter 6

Application Layer 417

Historical Evolution: Mobile Applications 419

6.1 General Issues 420
- **6.1.1** How Ports Work 420
- **6.1.2** How Servers Start 421
- **6.1.3** Classification of Servers 421

Historical Evolution: Cloud Computing 426
- **6.1.4** Characteristics of Application Layer Protocols 426

6.2 Domain Name System (DNS) 427
- **6.2.1** Introduction 427
- **6.2.2** Domain Name Space 428
- **6.2.3** Resource Records 430
- **6.2.4** Name Resolution 433

Historical Evolution: Root DNS Servers Worldwide 434

Open Source Implementation 6.1: BIND 437

6.3 Electronic Mail (E-Mail) 440
- **6.3.1** Introduction 440
- **6.3.2** Internet Message Standards 442
- **6.3.3** Internet Mail Protocols 447

Historical Evolution: Web-Based Mail vs. Desktop Mail 453

Open Source Implementation 6.2: `qmail` 454

6.4 World Wide Web (WWW) 459
- **6.4.1** Introduction 459
- **6.4.2** Web Naming and Addressing 460
- **6.4.3** HTML and XML 463
- **6.4.4** HTTP 464

Principle in Action: Non-WWW Traffic Over Port 80 or HTTP 466

Historical Evolution: Google Applications 467
- **6.4.5** Web Caching and Proxying 468

Open Source Implementation 6.3: Apache 470

Performance Matters: Throughput and Latency of a Web Server 473

6.5 File Transfer Protocol (FTP) 475
- **6.5.1** Introduction 475
- **6.5.2** The Two-Connection Operation Model: Out-of-Band Signaling 477

Historical Evolution: Why Out-of-Band Signaling in FTP? 478
- **6.5.3** FTP Protocol Messages 479

Open Source Implementation 6.4: `wu-ftpd` 482

6.6 Simple Network Management Protocol (SNMP) 485
- **6.6.1** Introduction 485
- **6.6.2** Architectural Framework 486
- **6.6.3** Management Information Base (MIB) 487
- **6.6.4** Basic Operations in SNMP 491

Open Source Implementation 6.5: `Net-SNMP` 493

6.7 Voice over IP (VoIP) 496
- **6.7.1** Introduction 497

Historical Evolution: Proprietary VoIP Services—Skype and MSN 498
- **6.7.2** H.323 498
- **6.7.3** Session Initialization Protocol (SIP) 501

Historical Evolution: H.323 vs. SIP 504

Open Source Implementation 6.6: Asterisk 505

6.8 Streaming 510
- **6.8.1** Introduction 510
- **6.8.2** Compression Algorithms 511
- **6.8.3** Streaming Protocols 512

Historical Evolution: Streaming with Real Player, Media Player, QuickTime, and YouTube 514
- **6.8.4** QoS and Synchronization Mechanisms 515

Open Source Implementation 6.7: Darwin Streaming Server 516

6.9 Peer-to-Peer Applications (P2P) 520
- **6.9.1** Introduction 520

Historical Evolution: Popular P2P Applications 522

Historical Evolution: Web 2.0 Social Networking: Facebook, Plurk, and Twitter 523
- **6.9.2** P2P Architectures 524
- **6.9.3** Performance Issues of P2P Applications 529
- **6.9.4** Case Study: BitTorrent 531

Open Source Implementation 6.8: BitTorrent 533

6.10 Summary 539

Common Pitfalls 540

Further Readings **541**
Frequently Asked Questions **543**
Exercises **544**

Chapter 7

Internet QoS **546**

Historical Evolution: The QoS Hype around 2000s **547**
7.1 General Issues **548**
7.1.1 Signaling Protocol 549
7.1.2 QoS Routing 549
7.1.3 Admission Control 549
7.1.4 Packet Classification 549
7.1.5 Policing 550
7.1.6 Scheduling 550
Open Source Implementation 7.1: Traffic Control Elements in Linux **551**
7.2 QoS Architectures **553**
7.2.1 Integrated Services (IntServ) 553
7.2.2 Differentiated Services (DiffServ) 556
Principle in Action: Why Both DiffServ and IntServ Failed **563**
Principle in Action: QoS in Wireless Links **563**
7.3 Algorithms for QoS Components **564**
7.3.1 Admission Control 564
Open Source Implementation 7.2: Traffic Estimator **566**
7.3.2 Flow Identification 568
Open Source Implementation 7.3: Flow Identification **568**
7.3.3 Token Bucket 570
Open Source Implementation 7.4: Token Bucket **571**
7.3.4 Packet Scheduling 574
Open Source Implementation 7.5: Packet Scheduling **578**
7.3.5 Packet Discarding 581
Open Source Implementation 7.6: Random Early Detection (RED) **583**
Principle in Action: QoS Components in Daily Usage Today **585**
7.4 Summary **586**
Common Pitfalls **586**
Further Readings **586**
QoS Components **587**
Frequently Asked Questions **588**
Exercises **588**

Chapter 8

Network Security **590**

8.1 General Issues **591**
8.1.1 Data Security 591
8.1.2 Access Security 593
8.1.3 System Security 593
8.2 Data Security **594**
8.2.1 Principles of Cryptography 595
Open Source Implementation 8.1: Hardware 3DES **598**
Principle in Action: Secure Wireless Channels **604**
8.2.2 Digital Signature and Message Authentication 604
Open Source Implementation 8.2: MD5 **606**
8.2.3 Link Layer Tunneling 609
8.2.4 IP Security (IPSec) 609
Open Source Implementation 8.3: AH and ESP in IPSec **612**
8.2.5 Transport Layer Security 614
Historical Evolution: HTTP Secure (HTTPS) and Secure Shell (SSH) **616**
8.2.6 Comparison on VPNs 618
8.3 Access Security **618**
8.3.1 Introduction 619
8.3.2 Network/Transport Layer Firewall 619
Open Source Implementation 8.4: Netfilter and iptables **621**
8.3.3 Application Layer Firewall 623
Open Source Implementation 8.5: FireWall Toolkit (FWTK) **624**
Principle in Action: Wireless Access Control **627**
8.4 System Security **627**
8.4.1 Information Gathering 628
8.4.2 Vulnerability Exploiting 629
8.4.3 Malicious Code 632
Open Source Implementation 8.6: ClamAV **634**
8.4.4 Typical Defenses 637

Principle in Action: Bottleneck in IDS 639
Principle in Action: Wireless Intrusions 640
Open Source Implementation 8.7: Snort 640
Open Source Implementation 8.8: SpamAssassin 645
Performance Matters: Comparing Intrusion Detection, Antivirus, Anti-Spam, Content Filtering, and P2P Classification 647
8.5 Summary 649
Common Pitfalls 649
Further Readings 650
Frequently Asked Questions 652
Exercises 652

Appendices

A Who's Who 654
A.1 IETF: Defining RFCs 655
A.1.1 IETF History 655
Historical Evolution: Who's Who in IETF 656
A.1.2 The RFC Process 657
A.1.3 The RFC Statistics 658
A.2 Open Source Communities 660
A.2.1 Beginning and Rules of the Game 660
A.2.2 Open Source Resources 661
A.2.3 Websites for Open Source 663
A.2.4 Events and People 664
A.3 Research and Other Standards Communities 665
A.4 History 666
Further Readings 668

B Linux Kernel Overview 669
B.1 Kernel Source Tree 670
B.2 Source Code for Networking 674
B.3 Tools for Source Code Tracing 677
Example: Trace of Reassembly of IPv4 Fragments 677
Further Readings 682

C Development Tools 683
C.1 Programming 684
C.1.1 Text Editor – `vim` and `gedit` 684
C.1.2 Compiler – `gcc` 685
C.1.3 Auto-Compile – `make` 688
C.2 Debugging 689
C.2.1 Debugger – `gdb` 689
C.2.2 GUI Debugger – `ddd` 690
C.2.3 Kernel Debugger – kgdb 693
C.3 Maintaining 694
C.3.1 Source Code Browser – `cscope` 694
C.3.2 Version Control – `Git` 696
C.4 Profiling 699
C.4.1 Profiler – `gprof` 700
C.4.2 Kernel Profiler – `kernprof` 701
C.5 Embedding 702
C.5.1 Tiny Utilities – `busybox` 703
C.5.2 Embedding Development – `uClibc` and buildroot 704
Further Readings 705

D Network Utilities 707
D.1 Name-Addressing 707
D.1.1 Internet's Who-Is-Who – `host` 708
D.1.2 LAN's Who-Is-Who – `arp` 708
D.1.3 Who Am I – `ifconfig` 709
D.2 Perimeter-Probing 710
D.2.1 Ping for Living – `ping` 711
D.2.2 Find the Way – `tracepath` 711
D.3 Traffic-Monitoring 713
D.3.1 Dump Raw Data – `tcpdump` 713
D.3.2 GUI Sniffer – `Wireshark` 714
D.3.3 Collect Network Statistics – `netstat` 714
D.4 Benchmarking 716
D.4.1 Host-to-Host Throughput – `ttcp` 716
D.5 Simulation and Emulation 717
D.5.1 Simulate the Network – `ns` 717
D.5.2 Emulate the Network – `NIST Net` 718
D.6 Hacking 720
D.6.1 Exploit Scanning – `Nessus` 720
Further Readings 722

Index 723