Preface

TRENDS IN NETWORKING COURSES

Technologies in computer networks have gone through many generations of evolution; many failed or faded away, some prevailed, and some are emerging today. The Internet technologies driven by TCP/IP currently dominate. Thus, a clear trend in organizing the content of courses in computer networks is to center around TCP/IP, adding *some* lower-layer link technologies and *many* upper-layer applications, while eliminating details about the faded technologies, and perhaps explaining why they faded away.

Textbooks on computer networking have also gone through several iterations of evolution, from traditional, and sometimes dry, protocol descriptions to the application-driven, top-down approach and the system-aspect approach. One trend is to explain more of the *why*, in addition to the *how,* for protocol behaviors so that readers can better appreciate various protocol designs. The evolution, however, shall continue.

GAP BETWEEN DESIGN AND IMPLEMENTATION

Another less clear trend is to add practical flavors to the protocol descriptions. Readers of other textbooks might not know *where* and *how* the protocol designs could be implemented. The net result is that when they do their research in the graduate schools they tend to simulate their designs for performance evaluation, instead of real implementation with real benchmarking. When they join the industry, they need to start from scratch to learn the implementation environment, skills, and issues. Apparently there is a *gap* between *knowledge* and *skills* for students trained by these textbooks. This gap could be bridged with *live running codes* easily accessible from the *open source* community.

AN OPEN SOURCE APPROACH

Almost all protocols in use today have implementations in the Linux operating system and in many open source packages. The Linux and open source communities have grown, and their applications predominate in the networking world. However, the abundant resources available there are *not yet leveraged* by the regular textbooks in computer science, and more specifically in computer networks. We envision a trend in textbooks for several courses that could leverage open source resources to narrow the gap between domain knowledge and hands-on skills. These courses include Operating Systems (with Linux kernel implementations as examples of process management, memory management, file system management, I/O management, etc.), Computer Organizations (with verilog codes in www.opencores.org as examples of processors, memory units, I/O device controllers, etc.), Algorithms (with GNU libraries as examples of classical algorithms), and Computer Networks (with open source codes as examples of protocol implementations). This text might prove to be an early example of this trend.

Our open source approach bridges the gap by *interleaving* the descriptions of protocol behaviors with vivid sample implementations extracted from open source packages. These examples are explicitly numbered with, say, Open Source Implementation 3.4. The source sites from which complete live examples can be downloaded are referred to in the text, so students can access them on the Internet easily. For example, immediately after explaining the concept of longest prefix matching in routing table lookup, we illustrate how the routing table is organized (as an ordered array of hash tables according to prefix lengths) and how this matching is implemented (as the *first* matching, since the matching process starts from the hash table with the longest prefixes) in the Linux kernel. This enables instructors to lecture on the design of routing table lookup and its implementation, and give sound hands-on projects to, for example, profile the bottleneck of routing table lookup or modify hash table implementation. We argue that this interleaving approach is better than a *separating* approach with a *second* course or text. It benefits the *average* students most because it ties together design and implementation, and the majority of students would not need a second course. With other textbooks, instructors, teaching assistants, and students have to make an extra effort to bridge this gap that has long been ignored, or in most cases, simply left untouched.

The protocol descriptions in this text are interleaved with 56 representative open source implementations, ranging from the Verilog or VHDL code of codec, modem, CRC32, CSMA/CD, and crypto, to the C code of adaptor driver, PPP daemon and driver, longest prefix matching, IP/TCP/UDP checksum, NAT, RIP/OSPF/BGP routing daemons, TCP slow-start and congestion avoidance, socket, popular packages supporting DNS, FTP, SMTP, POP3, SNMP, HTTP, SIP, streaming, P2P, to QoS features such as traffic shaper and scheduler, and security features such as firewall, VPN, and intrusion detection. This system-awareness is further fortified by *hands-on exercises* right at the end of each open source implementation and at the end of each chapter, where readers are asked to *run, search, trace, profile,* or *modify* the source codes of particular *kernel* code segments, *drivers,* or *daemons.* Students equipped with such system-awareness and hands-on skills, in addition to their protocol domain knowledge, can be expected to do more sound research works in academia and solid development works in industry.

WHY IS MORE IMPORTANT THAN *HOW*

This text was written with the idea that it is more important to understand *why* a protocol is designed a certain way than it is to know *how* it works. Many key concepts and underlying principles are illustrated before we explain how the mechanisms or protocols work. They include statelessness, control plane and data plane, routing and switching, collision and broadcast domains, scalability of bridging, classless and classful routing, address translation and configuration, forwarding versus routing, window flow control, RTT estimation, well-known ports and dynamic ports, iterative and concurrent servers, ASCII application protocol messages, variable-length versus fixed-field protocol messages, transparent proxy, and many others.

Misunderstandings are as important as understandings, and they deserve special treatment to identify them. We arrange each chapter to start with general issues to raise fundamental questions. We have added sidebars about Principles in Action, Historical Evolution, and Performance Matters. We end with unnumbered sections on Common Pitfalls (for common misunderstandings in the reader community), Further Readings, FAQs on big questions for readers to preview and review, and a set of hands-on and written exercises.

PREPARING THE AUDIENCE WITH SKILLS

Whether the instructors or students are familiar with Linux systems should not play a critical factor in adopting this textbook. The Linux-related hands-on skills are covered in Appendices B, C, and D. Three appendices equip readers with enough hands-on skills, including Linux kernel overview (with a tutorial on source code tracing), development tools (`vim`, `gcc`, `make`, `gdb`, `ddd`, `kgdb`, `cscope`, `cvs/svn`, `gprof/kernprof`, `busybox`, `buildroot`), and network utilities (`host`, `arp`, `ifconfig`, `ping`, `traceroute`, `tcpdump`, `wireshark`, `netstat`, `ttcp`, `webbench`, `ns`, `nist-net`, `nessus`). Appendix A also has a section introducing readers to open source resources. There is also a section on "A Packet's Life" in Chapter 1 to vividly illustrate the book's roadmap.

Lowering the barrier of adopting open source implementations is considered. Instead of code listing and explanation, it is structured into Overview, Block Diagram when needed, Data Structures, Algorithm Implementation, and Exercises. This provides for ease of adoption for both students and instructors.

PEDAGOGICAL FEATURES AND SUPPLEMENTS

Textbooks usually have a rich set of features to help readers and class support materials to help instructors. We offer a set of features and a set of class support materials, summarized as follows:

1. Fifty-six explicitly numbered Open Source Implementations for key protocols and mechanisms.
2. Four appendices on Who's Who in Internet and open source communities, Linux kernel overview, development tools, and network utilities.
3. Logically reasoned *why, where,* and *how* of protocol designs and implementations.
4. Motivating general issues at the beginning of each chapter with big questions to answer.
5. "A Packet's Life" from the server and router perspectives to illustrate the book's roadmap and show how to trace packet flows in codes.

6. "Common Pitfalls" illustrated at the end of each chapter, identifying common misunderstandings.
7. Hands-on Linux-based exercises in addition to written exercises.
8. Sixty-nine sidebars about historical evolution, principles, in action, and performance matters.
9. End-of-chapter FAQs to help readers identify key questions to answer and review after reading each chapter.
10. Class support materials, including PowerPoint lecture slides, solutions manual, and the text images in PowerPoint are available at the textbook Web site: www.mhhe.com/lin.

AUDIENCE AND COURSE ROADMAP

The book is intended to be a textbook in Computer Networks for senior undergraduates or first-year graduate students in computer science or electrical engineering. It could also be used by professional engineers in the data communication industry. For the undergraduate course, we recommend instructors cover only Chapters 1 through 6. For the graduate course, all chapters should be covered. For instructors who lecture both undergraduate and graduate courses, two other possible differentiations are heavier hands-on assignments and additional reading assignments in the graduate course. In either undergraduate or graduate courses, instructors could assign students to study the appendices in the first few weeks to get familiar with Linux and its development and utility tools. That familiarity could be checked by either a hands-on test or a hands-on assignment. Throughout the course, both written and hands-on exercises can be assigned to reinforce knowledge and skills.

The chapters are organized as follows:

- Chapter 1 offers background on the requirements and principles of networking, and then presents the Internet solutions to meet the requirements given the underlying principles. Design philosophies of the Internet, such as statelessness, connectionlessness, and the end-to-end argument are illustrated. Throughout the process, we raise key concepts, including connectivity, scalability, resource sharing, data and control planes, packet and circuit switching, latency, throughput, bandwidth, load, loss, jitter, standards and interoperability, routing and switching. Next we take Linux as an implementation of the Internet solutions to illustrate where and how the Internet architecture and its protocols are implemented into chips, drivers, kernel, and daemons. The chapter ends with a book roadmap and the interesting description of "A Packet's Life."
- Chapter 2 gives a concise treatment of the physical layer. It first offers conceptual background on analog and digital signals, wired and wireless media, coding, modulation, and multiplexing. Then it covers classical techniques and standards on coding, modulation, and multiplexing. Two open source implementations illustrate the hardware implementation of Ethernet PHY using 8B/10B encoding and WLAN PHY using OFDM.

- Chapter 3 introduces three dominant links: PPP, Ethernet, and WLAN. Bluetooth and WiMAX are also described. LAN interconnection through layer-2 bridging is then introduced. At the end, we detail the adaptor drivers that transmit and receive packets to and from the network interface card. Ten open source implementations, including hardware design of CRC32 and Ethernet MAC, are presented.
- Chapter 4 discusses the data plane and control plane of the IP layer. The data plane discussion includes IP forwarding process, routing table lookup, checksum, fragmentation, NAT, and the controversial IPv6, while the control plane discussion covers address management, error reporting, unicast routing, and multicast routing. Both routing protocols and algorithms are detailed. Twelve open source implementations are interleaved to illustrate how these designs are implemented.
- Chapter 5 moves up to the transport layer to cover the end-to-end, or host-to-host, issues. Both UDP and TCP are detailed, especially the design philosophies, behaviors, and versions of TCP. Then RTP for real-time multimedia traffic is introduced. A unique section follows to illustrate socket design and implementation where packets are copied between the kernel space and the user space. Ten open source implementations are presented.
- Chapter 6 covers both traditional applications, including DNS, Mail, FTP, Web, and SNMP, and new applications, including VoIP, streaming, and P2P applications. Eight open source packages that implement these eight applications are discussed.
- Chapter 7 touches on the advanced topic of QoS, where various traffic control modules such as policer, shaper, scheduler, dropper, and admission control are presented. Though the IntServ and DiffServ standard frameworks have not been widely deployed, many of these traffic control modules are embedded in products that are used every day. Hence they deserve a chapter. Six open source implementations are presented.
- Chapter 8 looks into network security issues ranging from access security (guarded by TCP/IP firewall and application firewall), data security (guarded by VPN), and system security (guarded by intrusion detection and antivirus). Both algorithms (table lookup, encryption, authentication, deep packet inspection) and standards (3DES, MD5, IPsec) are covered. Eight open source implementations are added.

ACKNOWLEDGMENTS

The draft of this text has gone through much evolution and revision. Throughout the process, many people have directly or indirectly contributed. First, many lab members and colleagues at National Chiao Tung University, National Chung Cheng University, and Cisco Systems, Inc., have contributed ideas, examples, and code explanations to this book. In particular, we would like to thank Po-Ching Lin, Shih-Chiang Weafon Tsao, Yi-Neng Lin, Huan-Yun Wei, Ben-Jye Chang, Shun-Lee Stanley Chang, Yuan-Cheng Lai, Jui-Tsun Jason Hung, Shau-Yu Jason Cheng, Chia-Yu Ku, Hsiao-Feng Francis Lu, and Frank Lin. Without their inputs, we would not have been able to embed many interesting and original ideas into this book. We also thank the National Science Council (NSC) in Taiwan, the Industrial Technology Research Institute (ITRI), D-Link Corporation, Realtek Semiconductor Corporation, ZyXEL Corporation, Cisco Systems, Inc., and Intel Corporation for supporting our networking research in the past few years.

Next, we wish to thank the following who reviewed drafts of all or parts of the manuscript: Emmanuel Agu, Worcester Polytechnic University; Tricha Anjali, Illinois Institute of Technology; Ladislau Boloni, University of Central Florida; Charles Colbourn, Arizona State University; XiaoJiang Du, Temple University; Jiang Guo, California State University, Los Angeles; Robert Kerbs, California State Polytechnic University, Pomona; Fang Liu, The University of Texas-Pan American; Oge Marques, Florida Atlantic University; Mitchell Neilsen, Kansas State University; Mahasweta Sarkar, San Diego State University; Edwin Sloan, Hillsborough Community College; Ioannis Viniotis, North Carolina State University; Bin Wang, Wright State University; Daniel Zappala, Brigham Young University. Thanks also to Chih-Chiang Wang, National Kaohsiung University of Applied Sciences, who polished the manuscript grammatically.

Finally, we would like to thank the folks at McGraw-Hill who coached us through the editorial and production phases. A special thanks should go to our Global Publisher, Raghu Srinivasan, our Developmental Editor, Lorraine Buczek, our production Project Manager, Jane Mohr, and Project Manager, Deepti Narwat. They have been very supportive coaches throughout this endeavor.