TCP/IP Network Administration

TCP/IP Network Administration, Third Edition, is a complete guide to setting up and running a TCP/IP network, and is geared toward system administrators as well as users of home systems that access the Internet. It starts with the fundamentals: what protocols do and how they work, how addresses and routing are used to move data through the network, and how to set up your network connection.

Beyond basic setup, this book discusses advanced routing protocols (RIPv2, OSPF, and BGP) and the gated software package that implements them. It provides a tutorial on configuring important network services, including DNS, Apache, sendmail, Samba, PPP, and DHCP. There are chapters on troubleshooting and security. In addition, this book contains a command and syntax reference for important packages such as gated, pppd, named, dhcpd, and sendmail.

This new edition includes a section on configuring Samba to provide file and print sharing on networks that integrate Unix and Windows, and a new chapter dedicated to the important task of configuring the Apache web server. Network security coverage is expanded to include details on OpenSSH, stunnel, gpg, iptables, and the access control mechanism in xinetd. This book also contains updated information about DNS, including details on BIND 8 and BIND 9, the role of classless IP addressing and network prefixes, and the changing role of registrars.


Praise for previous editions:

“The book you reach for first…”
—Marshall Rose, ConneXions

“...the definitive volume on the subject.”
—Tom Yager, BYTE

“...probably the best single Unix TCP/IP system administrator's handbook in print...”
—Anthony M. Rutkowski, SprintLink

“The second edition of Hunt's superb book is even more useful and informative than the original edition...an extraordinary and outstanding revision of a classic and indispensable reference.”
—Elizabeth Zinkann, Sys Admin
TCP/IP Network Administration
TCP/IP Network Administration

Craig Hunt
—To Alana, the beginning of a new life.
# Table of Contents

**Preface**  .......................................................................................................................... xi

1. **Overview of TCP/IP**  ........................................................................................................ 1
   - TCP/IP and the Internet  .................................................................................................. 2
   - A Data Communications Model  ..................................................................................... 6
   - TCP/IP Protocol Architecture  ...................................................................................... 9
   - Network Access Layer  .................................................................................................. 11
   - Internet Layer  .............................................................................................................. 12
   - Transport Layer  ........................................................................................................... 18
   - Application Layer  ........................................................................................................ 22
   - Summary  ..................................................................................................................... 23

2. **Delivering the Data**  ..................................................................................................... 24
   - Addressing, Routing, and Multiplexing  .................................................................... 24
   - The IP Address  ............................................................................................................ 25
   - Internet Routing Architecture  .................................................................................... 35
   - The Routing Table  ....................................................................................................... 37
   - Address Resolution  ..................................................................................................... 43
   - Protocols, Ports, and Sockets  ..................................................................................... 44
   - Summary  ..................................................................................................................... 50

3. **Network Services**  ......................................................................................................... 51
   - Names and Addresses  .................................................................................................. 51
   - The Host Table  ............................................................................................................ 52
   - DNS  ............................................................................................................................ 54
   - Mail Services  ............................................................................................................... 62
   - File and Print Servers  .................................................................................................. 75
   - Configuration Services  ............................................................................................... 76
   - Summary  ..................................................................................................................... 82
<table>
<thead>
<tr>
<th></th>
<th>Chapter Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Local Network Services</td>
<td>233</td>
</tr>
<tr>
<td></td>
<td>The Network File System</td>
<td>233</td>
</tr>
<tr>
<td></td>
<td>Sharing Unix Printers</td>
<td>252</td>
</tr>
<tr>
<td></td>
<td>Using Samba to Share Resources with Windows</td>
<td>259</td>
</tr>
<tr>
<td></td>
<td>Network Information Service</td>
<td>268</td>
</tr>
<tr>
<td></td>
<td>DHCP</td>
<td>272</td>
</tr>
<tr>
<td></td>
<td>Managing Distributed Servers</td>
<td>277</td>
</tr>
<tr>
<td></td>
<td>Post Office Servers</td>
<td>280</td>
</tr>
<tr>
<td></td>
<td>Summary</td>
<td>283</td>
</tr>
<tr>
<td>10</td>
<td>sendmail</td>
<td>285</td>
</tr>
<tr>
<td></td>
<td>sendmail’s Function</td>
<td>285</td>
</tr>
<tr>
<td></td>
<td>Running sendmail as a Daemon</td>
<td>286</td>
</tr>
<tr>
<td></td>
<td>sendmail Aliases</td>
<td>288</td>
</tr>
<tr>
<td></td>
<td>The sendmail.cf File</td>
<td>290</td>
</tr>
<tr>
<td></td>
<td>sendmail.cf Configuration Language</td>
<td>297</td>
</tr>
<tr>
<td></td>
<td>Rewriting the Mail Address</td>
<td>309</td>
</tr>
<tr>
<td></td>
<td>Modifying a sendmail.cf File</td>
<td>319</td>
</tr>
<tr>
<td></td>
<td>Testing sendmail.cf</td>
<td>323</td>
</tr>
<tr>
<td></td>
<td>Summary</td>
<td>332</td>
</tr>
<tr>
<td>11</td>
<td>Configuring Apache</td>
<td>333</td>
</tr>
<tr>
<td></td>
<td>Installing Apache Software</td>
<td>334</td>
</tr>
<tr>
<td></td>
<td>Configuring the Apache Server</td>
<td>338</td>
</tr>
<tr>
<td></td>
<td>Understanding an httpd.conf File</td>
<td>341</td>
</tr>
<tr>
<td></td>
<td>Web Server Security</td>
<td>361</td>
</tr>
<tr>
<td></td>
<td>Managing Your Web Server</td>
<td>378</td>
</tr>
<tr>
<td></td>
<td>Summary</td>
<td>380</td>
</tr>
<tr>
<td>12</td>
<td>Network Security</td>
<td>381</td>
</tr>
<tr>
<td></td>
<td>Security Planning</td>
<td>382</td>
</tr>
<tr>
<td></td>
<td>User Authentication</td>
<td>387</td>
</tr>
<tr>
<td></td>
<td>Application Security</td>
<td>402</td>
</tr>
<tr>
<td></td>
<td>Security Monitoring</td>
<td>404</td>
</tr>
<tr>
<td></td>
<td>Access Control</td>
<td>409</td>
</tr>
<tr>
<td></td>
<td>Encryption</td>
<td>418</td>
</tr>
<tr>
<td></td>
<td>Firewalls</td>
<td>425</td>
</tr>
<tr>
<td></td>
<td>Words to the Wise</td>
<td>433</td>
</tr>
<tr>
<td></td>
<td>Summary</td>
<td>434</td>
</tr>
</tbody>
</table>
The first edition of *TCP/IP Network Administration* was written in 1992. In the
decade since, many things have changed, yet some things remain the same. TCP/IP is
still the preeminent communications protocol for linking together diverse computer
systems. It remains the basis of interoperable data communications and global com-
puter networking. The underlying Internet Protocol (IP), Transmission Control Pro-
tocol, and User Datagram Protocol (UDP) are remarkably unchanged. But change
has come in the way TCP/IP is used and how it is managed.

A clear symbol of this change is the fact that my mother-in-law has a TCP/IP net-
work connection in her home that she uses to exchange electronic mail, compressed
graphics, and hypertext documents with other senior citizens. She thinks of this as
“just being on the Internet,” but the truth is that her small system contains a func-
tioning TCP/IP protocol stack, manages a dynamically assigned IP address, and han-
dles data types that did not even exist a decade ago.

In 1991, TCP/IP was a tool of sophisticated users. Network administrators managed
a limited number of systems and could count on the users for a certain level of tech-
nical knowledge. No more. In 2002, the need for highly trained network administra-
tors is greater than ever because the user base is larger, more diverse, and less
capable of handling technical problems on its own. This book provides the informa-
tion needed to become an effective TCP/IP network administrator.

*TCP/IP Network Administration* was the first book of practical information for the
professional TCP/IP network administrator, and it is still the best. Since the first edi-
tion was published there has been an explosion of books about TCP/IP and the Inter-
net. Still, too few books concentrate on what a system administrator really needs to
know about TCP/IP administration. Most books are either scholarly texts written
from the point of view of the protocol designer, or instructions on how to use TCP/IP
applications. All of those books lack the practical, detailed network information
needed by the Unix system administrator. This book strives to focus on TCP/IP and
Unix and to find the right balance of theory and practice.
I am proud of the earlier editions of *TCP/IP Network Administration*. In this edition, I have done everything I can to maintain the essential character of the book while making it better. Dynamic address assignment based on Dynamic Host Configuration Protocol (DHCP) is covered. The Domain Name System material has been updated to cover BIND 8 and, to a lesser extent, BIND 9. The email configuration is based on current version of sendmail 8, and the operating system examples are from the current versions of Solaris and Linux. The routing protocol coverage includes Routing Information Protocol version 2 (RIPv2), Open Shortest Path First (OSPF), and Border Gateway Protocol (BGP). I have also added a chapter on Apache web server configuration, new material on `xinetd`, and information about building a firewall with `iptables`. Despite the additional topics, the book has been kept to a reasonable length.

TCP/IP is a set of communications protocols that define how different types of computers talk to each other. *TCP/IP Network Administration* is a book about building your own network based on TCP/IP. It is both a tutorial covering the “why” and “how” of TCP/IP networking, and a reference manual for the details about specific network programs.

**Audience**

This book is intended for everyone who has a Unix computer connected to a TCP/IP network.* This obviously includes the network managers and the system administrators who are responsible for setting up and running computers and networks, but it also includes any user who wants to understand how his or her computer communicates with other systems. The distinction between a “system administrator” and an “end user” is a fuzzy one. You may think of yourself as an end user, but if you have a Unix workstation on your desk, you’re probably also involved in system administration tasks.

Over the last several years there has been a rash of books for “dummies” and “idiots.” If you really think of yourself as an “idiot” when it comes to Unix, this book is not for you. Likewise, if you are a network administration “genius,” this book is probably not suitable either. If you fall anywhere between these two extremes, however, you’ll find this book has a lot to offer.

This book assumes that you have a good understanding of computers and their operation and that you’re generally familiar with Unix system administration. If you’re not, the Nutshell Handbook *Essential System Administration* by Æleen Frisch (published by O’Reilly & Associates) will fill you in on the basics.

* Much of this text also applies to non-Unix systems. Some of the file formats and commands and all of the protocol descriptions apply equally well to Windows, Windows NT/2000, and other operating systems. If you’re an NT administrator, you should read Windows NT TCP/IP Network Administration (O’Reilly).
Organization

Conceptually, this book is divided into three parts: fundamental concepts, tutorial, and reference. The first three chapters are a basic discussion of the TCP/IP protocols and services. This discussion provides the fundamental concepts necessary to understand the rest of the book. The remaining chapters provide a “how-to” tutorial. Chapters 4–7 discuss how to plan a network installation and configure the basic software necessary to get a network running. Chapters 8–11 discuss how to set up various important network services. Chapters 12 and 13 cover how to perform the ongoing tasks that are essential for a reliable network: security and troubleshooting. The book concludes with a series of appendixes that are technical references for important commands and programs.

This book contains the following chapters:

Chapter 1, Overview of TCP/IP, gives the history of TCP/IP, a description of the protocol architecture, and a basic explanation of how the protocols function.

Chapter 2, Delivering the Data, describes addressing and how data passes through a network to reach the proper destination.

Chapter 3, Network Services, discusses the relationship between clients and server systems and the various services that are central to the function of a modern internet.

Chapter 4, Getting Started, begins the discussion of network setup and configuration. This chapter discusses the preliminary configuration planning needed before you configure the systems on your network.

Chapter 5, Basic Configuration, describes how to configure TCP/IP in the Unix kernel, and how to configure the system to start the network services.

Chapter 6, Configuring the Interface, tells you how to identify a network interface to the network software. This chapter provides examples of Ethernet and PPP interface configurations.

Chapter 7, Configuring Routing, describes how to set up routing so that systems on your network can communicate properly with other networks. It covers the static routing table, commonly used routing protocols, and gated, a package that provides the latest implementations of several routing protocols.

Chapter 8, Configuring DNS, describes how to administer the name server program that converts system names to Internet addresses.

Chapter 9, Local Network Services, describes how to configure many common network servers. The chapter discusses the DHCP configuration server, the LPD print server, the POP and IMAP mail servers, the Network File System (NFS), the Samba file and print server, and the Network Information System (NIS).
Chapter 10, sendmail, discusses how to configure sendmail, which is the daemon responsible for delivering electronic mail.

Chapter 11, Configuring Apache, describes how the Apache web server software is configured.

Chapter 12, Network Security, discusses how to live on the Internet without excessive risk. This chapter covers the security threats introduced by the network, and describes the plans and preparations you can make to meet those threats.

Chapter 13, Troubleshooting TCP/IP, tells you what to do when something goes wrong. It describes the techniques and tools used to troubleshoot TCP/IP problems and gives examples of actual problems and their solutions.

Appendix A, PPP Tools, is a reference guide to the various programs used to configure a serial port for TCP/IP. The reference covers dip, pppd, and chat.

Appendix B, A gated Reference, is a reference guide to the configuration language of the gated routing package.

Appendix C, A named Reference, is a reference guide to the Berkeley Internet Name Domain (BIND) name server software.

Appendix D, A dhcpd Reference, is a reference guide to the Dynamic Host Configuration Protocol Daemon (dhcpd).

Appendix E, A sendmail Reference, is a reference guide to sendmail syntax, options, and flags.

Appendix F, Solaris httpd.conf File, lists the contents of the Apache configuration file discussed in Chapter 11.

Appendix G, RFC Excerpts, contains detailed protocol references taken directly from the RFCs that support the protocol troubleshooting examples in Chapter 13. This appendix explains how to obtain your own copies of the RFCs.

Unix Versions

Most of the examples in this book are taken from Red Hat Linux, currently the most popular Linux distribution, and from Solaris 8, the Sun operating system based on System V Unix. Fortunately, TCP/IP software is remarkably standard from system to system, and because of this uniformity, the examples should be applicable to any Linux, System V, or BSD-based Unix system. There are small variations in command output or command-line options, but these should not present a problem.

Some of the ancillary networking software is identified separately from the Unix operating system by its own release number. Many such packages are discussed, and when appropriate are identified by their own release numbers. The most important of these packages are:
BIND

Our discussion of the BIND software is based on version 8 running on a Solaris 8 system. BIND 8 is the version of the BIND software delivered with Solaris, and supports all of the standard resource records. There are relatively few administrative differences between BIND 8 and the newer BIND 9 release for basic configurations.

sendmail

Our discussion of sendmail is based on release 8.11.3. This version should be compatible with other releases of sendmail v8.

Conventions

This book uses the following typographical conventions:

Italic

is used for the names of files, directories, hostnames, domain names, and to emphasize new terms when they are introduced.

Constant width

is used to show the contents of files or the output from commands. It is also used to represent commands, options, and keywords in text.

Constant width bold

is used in examples to show commands typed on the command line.

Constant width italic

is used in examples and text to show variables for which a context-specific substitution should be made. (The variable filename, for example, would be replaced by some actual filename.)

%, #

Commands that you would give interactively are shown using the default C shell prompt (%). If the command must be executed as root, it is shown using the default superuser prompt (#). Because the examples may include multiple systems on a network, the prompt may be preceded by the name of the system on which the command was given.

[ option ]

When showing command syntax, optional parts of the command are placed within brackets. For example, ls [ -l ] means that the -l option is not required.

We’d Like to Hear from You

We have tested and verified all of the information in this book to the best of our ability, but you may find that features have changed (or even that we have made
mistakes!). Please let us know about any errors you find, as well as your suggestions for future editions, by writing:

O’Reilly & Associates, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
(800) 998-9938 (in the United States or Canada)
(707) 829-0515 (international or local)
(707) 829-0104 (fax)

There is a web page for this book, where we list errata, examples, or any additional information. You can access this page at:

http://www.oreilly.com/catalog/tcp3

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about books, conferences, Resource Centers, and the O’Reilly Network, see our web site at:

http://www.oreilly.com

To find out what else Craig is doing, visit his web site, http://www.wrotethebook.com.

Acknowledgments

I would like to thank the many people who helped in the preparation of this book. All of the people who contributed to the first and second editions deserve thanks because so much of their input lives on in this edition. For the first edition that’s John Wack, Matt Bishop, Wietse Venema, Eric Allman, Jeff Honig, Scott Brim, and John Dorgan. For the second edition that’s Eric Allman again, Bryan Costales, Cricket Liu, Paul Albitz, Ted Lemon, Elizabeth Zwicky, Brent Chapman, Simson Garfinkel, Jeff Sedayao, and Æleen Frisch.

The third edition has also benefited from many contributors—a surprising number of whom are authors in their own right. They set me straight about the technical details and improved my prose. Three authors are due special thanks. Cricket Liu, one of the authors of the best book ever written about DNS, provided many comments that improved the sections on Domain Name System. David Collier-Brown, one of the authors of Using Samba, did a complete technical review of the Samba material. Charles Aulds, author of a best-selling book on Apache administration, provided insights into Apache configuration. All of these people helped me make this book better than earlier editions. Thanks!

All the people at O’Reilly & Associates have been very helpful. Deb Cameron, my editor, deserves a special thanks. Deb kept the whole thing moving forward while balancing the demands of a beautifully wonderful daughter, Bethany Rose. Emily Quill was
the production editor and project manager. Jeff Holcomb and Jane Ellin performed quality control checks. Leanne Soylemez provided production assistance. Tom Dinse wrote the index. Edie Freedman designed the cover, and Melanie Wang designed the interior format of the book. Neil Walls converted the book from Microsoft Word to Framemaker. Chris Reilley and Robert Romano’s illustrations from the earlier editions have been updated by Robert Romano and Jessamyn Read.

Finally, I want to thank my family—Kathy, Sara, David, and Rebecca. They keep my feet on the ground when the pressure to meet deadlines is driving me into orbit. They are the best.
All of us who use a Unix desktop system—engineers, educators, scientists, and business people—have second careers as Unix system administrators. Networking these computers gives us new tasks as network administrators.

Network administration and system administration are two different jobs. System administration tasks such as adding users and doing backups are isolated to one independent computer system. Not so with network administration. Once you place your computer on a network, it interacts with many other systems. The way you do network administration tasks has effects, good and bad, not only on your system but on other systems on the network. A sound understanding of basic network administration benefits everyone.

Networking your computers dramatically enhances their ability to communicate—and most computers are used more for communication than computation. Many mainframes and supercomputers are busy crunching the numbers for business and science, but the number of these systems in use pales in comparison to the millions of systems busy moving mail to a remote colleague or retrieving information from a remote repository. Further, when you think of the hundreds of millions of desktop systems that are used primarily for preparing documents to communicate ideas from one person to another, it is easy to see why most computers can be viewed as communications devices.

The positive impact of computer communications increases with the number and type of computers that participate in the network. One of the great benefits of TCP/IP is that it provides interoperable communications between all types of hardware and all kinds of operating systems.

The name “TCP/IP” refers to an entire suite of data communications protocols. The suite gets its name from two of the protocols that belong to it: the Transmission Control Protocol (TCP) and the Internet Protocol (IP). TCP/IP is the traditional name for this protocol suite and it is the name used in this book. The TCP/IP protocol suite is also called the Internet Protocol Suite (IPS). Both names are acceptable.
This book is a practical, step-by-step guide to configuring and managing TCP/IP networking software on Unix computer systems. TCP/IP is the leading communications software for local area networks and enterprise intranets, and it is the foundation of the worldwide Internet. TCP/IP is the most important networking software available to a Unix network administrator.

The first part of this book discusses the basics of TCP/IP and how it moves data across a network. The second part explains how to configure and run TCP/IP on a Unix system. Let’s start with a little history.

TCP/IP and the Internet

In 1969 the Advanced Research Projects Agency (ARPA) funded a research and development project to create an experimental packet-switching network. This network, called the ARPAnet, was built to study techniques for providing robust, reliable, vendor-independent data communications. Many techniques of modern data communications were developed in the ARPAnet.

The experimental network was so successful that many of the organizations attached to it began to use it for daily data communications. In 1975 the ARPAnet was converted from an experimental network to an operational network, and the responsibility for administering the network was given to the Defense Communications Agency (DCA).* However, development of the ARPAnet did not stop just because it was being used as an operational network; the basic TCP/IP protocols were developed after the network was operational.

The TCP/IP protocols were adopted as Military Standards (MIL STD) in 1983, and all hosts connected to the network were required to convert to the new protocols. To ease this conversion, DARPA† funded Bolt, Beranek, and Newman (BBN) to implement TCP/IP in Berkeley (BSD) Unix. Thus began the marriage of Unix and TCP/IP.

About the time that TCP/IP was adopted as a standard, the term Internet came into common usage. In 1983 the old ARPAnet was divided into MILNET, the unclassified part of the Defense Data Network (DDN), and a new, smaller ARPAnet. “Internet” was used to refer to the entire network: MILNET plus ARPAnet.

In 1985 the National Science Foundation (NSF) created NSFNet and connected it to the then-existing Internet. The original NSFNet linked together the five NSF supercomputer centers. It was smaller than the ARPAnet and no faster: 56Kbps. Still, the

* DCA has since changed its name to Defense Information Systems Agency (DISA).
† During the 1980s, ARPA, which is part of the U.S. Department of Defense, became Defense Advanced Research Projects Agency (DARPA). Whether it is known as ARPA or DARPA, the agency and its mission of funding advanced research have remained the same.
creation of the NSFNet was a significant event in the history of the Internet because NSF brought with it a new vision of the use of the Internet. NSF wanted to extend the network to every scientist and engineer in the United States. To accomplish this, in 1987 NSF created a new, faster backbone and a three-tiered network topology that included the backbone, regional networks, and local networks. In 1990 the ARPAnet formally passed out of existence, and in 1995 the NSFNet ceased its role as a primary Internet backbone network.

Today the Internet is larger than ever and encompasses hundreds of thousands of networks worldwide. It is no longer dependent on a core (or backbone) network or on governmental support. Today’s Internet is built by commercial providers. National network providers, called tier-one providers, and regional network providers create the infrastructure. Internet Service Providers (ISPs) provide local access and user services. This network of networks is linked together in the United States at several major interconnection points called Network Access Points (NAPs).

The Internet has grown far beyond its original scope. The original networks and agencies that built the Internet no longer play an essential role for the current network. The Internet has evolved from a simple backbone network, through a three-tiered hierarchical structure, to a huge network of interconnected, distributed network hubs. It has grown exponentially since 1983—doubling in size every year. Through all of this incredible change one thing has remained constant: the Internet is built on the TCP/IP protocol suite.

A sign of the network’s success is the confusion that surrounds the term internet. Originally it was used only as the name of the network built upon IP. Now internet is a generic term used to refer to an entire class of networks. An internet (lowercase “i”) is any collection of separate physical networks, interconnected by a common protocol, to form a single logical network. The Internet (uppercase “I”) is the worldwide collection of interconnected networks, which grew out of the original ARPAnet, that uses IP to link the various physical networks into a single logical network. In this book, both “internet” and “Internet” refer to networks that are interconnected by TCP/IP.

Because TCP/IP is required for Internet connection, the growth of the Internet spurred interest in TCP/IP. As more organizations became familiar with TCP/IP, they saw that its power can be applied in other network applications as well. The Internet protocols are often used for local area networking even when the local network is not connected to the Internet. TCP/IP is also widely used to build enterprise networks. TCP/IP-based enterprise networks that use Internet techniques and web tools to disseminate internal corporate information are called intranets. TCP/IP is the foundation of all of these varied networks.
TCP/IP Features

The popularity of the TCP/IP protocols did not grow rapidly just because the protocols were there, or because connecting to the Internet mandated their use. They met an important need (worldwide data communication) at the right time, and they had several important features that allowed them to meet this need. These features are:

- Open protocol standards, freely available and developed independently from any specific computer hardware or operating system. Because it is so widely supported, TCP/IP is ideal for uniting different hardware and software components, even if you don’t communicate over the Internet.
- Independence from specific physical network hardware. This allows TCP/IP to integrate many different kinds of networks. TCP/IP can be run over an Ethernet, a DSL connection, a dial-up line, an optical network, and virtually any other kind of physical transmission medium.
- A common addressing scheme that allows any TCP/IP device to uniquely address any other device in the entire network, even if the network is as large as the worldwide Internet.
- Standardized high-level protocols for consistent, widely available user services.

Protocol Standards

Protocols are formal rules of behavior. In international relations, protocols minimize the problems caused by cultural differences when various nations work together. By agreeing to a common set of rules that are widely known and independent of any nation’s customs, diplomatic protocols minimize misunderstandings; everyone knows how to act and how to interpret the actions of others. Similarly, when computers communicate, it is necessary to define a set of rules to govern their communications.

In data communications, these sets of rules are also called protocols. In homogeneous networks, a single computer vendor specifies a set of communications rules designed to use the strengths of the vendor’s operating system and hardware architecture. But homogeneous networks are like the culture of a single country—only the natives are truly at home in it. TCP/IP creates a heterogeneous network with open protocols that are independent of operating system and architectural differences. TCP/IP protocols are available to everyone and are developed and changed by consensus, not by the fiat of one manufacturer. Everyone is free to develop products to meet these open protocol specifications.

The open nature of TCP/IP protocols requires an open standards development process and publicly available standards documents. Internet standards are developed by the Internet Engineering Task Force (IETF) in open, public meetings. The protocols
developed in this process are published as Requests for Comments (RFCs). As the title “Request for Comments” implies, the style and content of these documents are much less rigid than in most standards documents. RFCs contain a wide range of interesting and useful information, and are not limited to the formal specification of data communications protocols. There are three basic types of RFCs: standards (STD), best current practices (BCP), and informational (FYI).

RFCs that define official protocol standards are STDs and are given an STD number in addition to an RFC number. Creating an official Internet standard is a rigorous process. Standards track RFCs pass through three maturity levels before becoming standards:

**Proposed Standard**
This is a protocol specification that is important enough and has received enough Internet community support to be considered for a standard. The specification is stable and well understood, but it is not yet a standard and may be withdrawn from consideration to be a standard.

**Draft Standard**
This is a protocol specification for which at least two independent, interoperable implementations exist. A draft standard is a final specification undergoing widespread testing. It will change only if the testing forces a change.

**Internet Standard**
A specification is declared a standard only after extensive testing and only if the protocol defined in the specification is considered to be of significant benefit to the Internet community.

There are two categories of standards. A Technical Specification (TS) defines a protocol. An Applicability Statement (AS) defines when the protocol is to be used. There are three requirement levels that define the applicability of a standard:

**Required**
This standard protocol is a required part of every TCP/IP implementation. It must be included for the TCP/IP stack to be compliant.

**Recommended**
This standard protocol should be included in every TCP/IP implementation, although it is not required for minimal compliance.

**Elective**
This standard is optional. It is up to the software vendor to implement it or not.

Two other requirements levels (limited use and not recommended) apply to RFCs that are not part of the standards track. A “limited use” protocol is used only in special

* Interested in finding out how Internet standards are created? Read RFC 2026, The Internet Standards Process.
circumstances, such as during an experiment. A protocol is “not recommended” when it has limited functionality or is outdated. There are three types of non-standards track RFCs:

**Experimental**

An experimental RFC is limited to use in research and development.

**Historic**

A historic RFC is outdated and no longer recommended for use.

**Informational**

An informational RFC provides information of general interest to the Internet community; it does not define an Internet standard protocol.

A subset of the informational RFCs is called the FYI (For Your Information) notes. An FYI document is given an FYI number in addition to an RFC number. FYI documents provide introductory and background material about the Internet and TCP/IP networks. FYI documents are not mentioned in RFC 2026 and are not included in the Internet standards process. But there are several interesting FYI documents available.*

Another group of RFCs that go beyond documenting protocols are the Best Current Practices (BCP) RFCs. BCPs formally document techniques and procedures. Some of these document the way that the IETF conducts itself; RFC 2026 is an example of this type of BCP. Others provide guidelines for the operation of a network or service; RFC 1918, *Address Allocation for Private Internets*, is an example of this type of BCP. BCPs that provide operational guidelines are often of great interest to network administrators.

There are now more than 3,000 RFCs. As a network system administrator, you will no doubt read several. It is as important to know which ones to read as it is to understand them when you do read them. Use the RFC categories and the requirements levels to help you determine which RFCs are applicable to your situation. (A good starting point is to focus on those RFCs that also have an STD number.) To understand what you read, you need to understand the language of data communications. RFCs contain protocol implementation specifications defined in terminology that is unique to data communications.

**A Data Communications Model**

To discuss computer networking, it is necessary to use terms that have special meaning. Even other computer professionals may not be familiar with all the terms in the networking alphabet soup. As is always the case, English and computer-speak are

* To find out more about FYI documents, read RFC 1150, *FYI on FYI: An Introduction to the FYI Notes*. 

6 | Chapter 1: Overview of TCP/IP
not equivalent (or even necessarily compatible) languages. Although descriptions and examples should make the meaning of the networking jargon more apparent, sometimes terms are ambiguous. A common frame of reference is necessary for understanding data communications terminology.

An architectural model developed by the International Standards Organization (ISO) is frequently used to describe the structure and function of data communications protocols. This architectural model, which is called the Open Systems Interconnect (OSI) Reference Model, provides a common reference for discussing communications. The terms defined by this model are well understood and widely used in the data communications community—so widely used, in fact, that it is difficult to discuss data communications without using OSI’s terminology.

The OSI Reference Model contains seven layers that define the functions of data communications protocols. Each layer of the OSI model represents a function performed when data is transferred between cooperating applications across an intervening network. Figure 1-1 identifies each layer by name and provides a short functional description for it. Looking at this figure, the protocols are like a pile of building blocks stacked one upon another. Because of this appearance, the structure is often called a stack or protocol stack.

![Figure 1-1. The OSI Reference Model](image-url)
A layer does not define a single protocol—it defines a data communications function that may be performed by any number of protocols. Therefore, each layer may contain multiple protocols, each providing a service suitable to the function of that layer. For example, a file transfer protocol and an electronic mail protocol both provide user services, and both are part of the Application Layer.

Every protocol communicates with its peers. A peer is an implementation of the same protocol in the equivalent layer on a remote system; i.e., the local file transfer protocol is the peer of a remote file transfer protocol. Peer-level communications must be standardized for successful communications to take place. In the abstract, each protocol is concerned only with communicating to its peers; it does not care about the layers above or below it.

However, there must also be agreement on how to pass data between the layers on a single computer, because every layer is involved in sending data from a local application to an equivalent remote application. The upper layers rely on the lower layers to transfer the data over the underlying network. Data is passed down the stack from one layer to the next until it is transmitted over the network by the Physical Layer protocols. At the remote end, the data is passed up the stack to the receiving application. The individual layers do not need to know how the layers above and below them function; they need to know only how to pass data to them. Isolating network communications functions in different layers minimizes the impact of technological change on the entire protocol suite. New applications can be added without changing the physical network, and new network hardware can be installed without rewriting the application software.

Although the OSI model is useful, the TCP/IP protocols don’t match its structure exactly. Therefore, in our discussions of TCP/IP, we use the layers of the OSI model in the following way:

**Application Layer**

The Application Layer is the level of the protocol hierarchy where user-accessed network processes reside. In this text, a TCP/IP application is any network process that occurs above the Transport Layer. This includes all of the processes that users directly interact with as well as other processes at this level that users are not necessarily aware of.

**Presentation Layer**

For cooperating applications to exchange data, they must agree about how data is represented. In OSI, the Presentation Layer provides standard data presentation routines. This function is frequently handled within the applications in TCP/IP, though TCP/IP protocols such as XDR and MIME also perform this function.

**Session Layer**

As with the Presentation Layer, the Session Layer is not identifiable as a separate layer in the TCP/IP protocol hierarchy. The OSI Session Layer manages the
sessions (connections) between cooperating applications. In TCP/IP, this function largely occurs in the Transport Layer, and the term “session” is not used; instead, the terms “socket” and “port” are used to describe the path over which cooperating applications communicate.

**Transport Layer**

Much of our discussion of TCP/IP is directed to the protocols that occur in the Transport Layer. The Transport Layer in the OSI reference model guarantees that the receiver gets the data exactly as it was sent. In TCP/IP, this function is performed by the *Transmission Control Protocol* (TCP). However, TCP/IP offers a second Transport Layer service, *User Datagram Protocol* (UDP), that does not perform the end-to-end reliability checks.

**Network Layer**

The Network Layer manages connections across the network and isolates the upper layer protocols from the details of the underlying network. The *Internet Protocol* (IP), which isolates the upper layers from the underlying network and handles the addressing and delivery of data, is usually described as TCP/IP’s Network Layer.

**Data Link Layer**

The reliable delivery of data across the underlying physical network is handled by the Data Link Layer. TCP/IP rarely creates protocols in the Data Link Layer. Most RFCs that relate to the Data Link Layer discuss how IP can make use of existing data link protocols.

**Physical Layer**

The Physical Layer defines the characteristics of the hardware needed to carry the data transmission signal. Features such as voltage levels and the number and location of interface pins are defined in this layer. Examples of standards at the Physical Layer are interface connectors such as RS232C and V.35, and standards for local area network wiring such as IEEE 802.3. TCP/IP does not define physical standards—it makes use of existing standards.

The terminology of the OSI reference model helps us describe TCP/IP, but to fully understand it, we must use an architectural model that more closely matches the structure of TCP/IP. The next section introduces the protocol model we’ll use to describe TCP/IP.

**TCP/IP Protocol Architecture**

While there is no universal agreement about how to describe TCP/IP with a layered model, TCP/IP is generally viewed as being composed of fewer layers than the seven used in the OSI model. Most descriptions of TCP/IP define three to five functional levels in the protocol architecture. The four-layer model illustrated in Figure 1-2 is based on the three layers (Application, Host-to-Host, and Network Access) shown in
the DOD Protocol Model in the *DDN Protocol Handbook Volume 1*, with the addition of a separate Internet layer. This model provides a reasonable pictorial representation of the layers in the TCP/IP protocol hierarchy.

As in the OSI model, data is passed down the stack when it is being sent to the network, and up the stack when it is being received from the network. The four-layered structure of TCP/IP is seen in the way data is handled as it passes down the protocol stack from the Application Layer to the underlying physical network. Each layer in the stack adds control information to ensure proper delivery. This control information is called a **header** because it is placed in front of the data to be transmitted. Each layer treats all the information it receives from the layer above as data, and places its own header in front of that information. The addition of delivery information at every layer is called **encapsulation**. (See Figure 1-3 for an illustration of this.) When data is received, the opposite happens. Each layer strips off its header before passing the data on to the layer above. As information flows back up the stack, information received from a lower layer is interpreted as both a header and data.

Each layer has its own independent data structures. Conceptually, a layer is unaware of the data structures used by the layers above and below it. In reality, the data structures of a layer are designed to be compatible with the structures used by the surrounding layers for the sake of more efficient data transmission. Still, each layer has its own data structure and its own terminology to describe that structure.

Figure 1-4 shows the terms used by different layers of TCP/IP to refer to the data being transmitted. Applications using TCP refer to data as a **stream**, while applications using UDP refer to data as a **message**. TCP calls data a **segment**, and UDP calls its data a **packet**. The Internet layer views all data as blocks called **datagrams**. TCP/IP uses many different types of underlying networks, each of which may have a different terminology for the data it transmits. Most networks refer to transmitted data as **packets** or **frames**. Figure 1-4 shows that one network transmits pieces of data it calls **frames**.
Let’s look more closely at the function of each layer, working our way up from the Network Access Layer to the Application Layer.

**Network Access Layer**

The **Network Access Layer** is the lowest layer of the TCP/IP protocol hierarchy. The protocols in this layer provide the means for the system to deliver data to the other devices on a directly attached network. This layer defines how to use the network to transmit an IP datagram. Unlike higher-level protocols, Network Access Layer
protocols must know the details of the underlying network (its packet structure, addressing, etc.) to correctly format the data being transmitted to comply with the network constraints. The TCP/IP Network Access Layer can encompass the functions of all three lower layers of the OSI Reference Model (Network, Data Link, and Physical).

The Network Access Layer is often ignored by users. The design of TCP/IP hides the function of the lower layers, and the better-known protocols (IP, TCP, UDP, etc.) are all higher-level protocols. As new hardware technologies appear, new Network Access protocols must be developed so that TCP/IP networks can use the new hardware. Consequently, there are many access protocols—one for each physical network standard.

Functions performed at this level include encapsulation of IP datagrams into the frames transmitted by the network, and mapping of IP addresses to the physical addresses used by the network. One of TCP/IP’s strengths is its universal addressing scheme. The IP address must be converted into an address that is appropriate for the physical network over which the datagram is transmitted.

Two RFCs that define Network Access Layer protocols are:

- RFC 826, Address Resolution Protocol (ARP), which maps IP addresses to Ethernet addresses
- RFC 894, A Standard for the Transmission of IP Datagrams over Ethernet Networks, which specifies how IP datagrams are encapsulated for transmission over Ethernet networks

As implemented in Unix, protocols in this layer often appear as a combination of device drivers and related programs. The modules that are identified with network device names usually encapsulate and deliver the data to the network, while separate programs perform related functions such as address mapping.

**Internet Layer**

The layer above the Network Access Layer in the protocol hierarchy is the Internet Layer. The Internet Protocol (IP) is the most important protocol in this layer. The release of IP used in the current Internet is IP version 4 (IPv4), which is defined in RFC 791. There are more recent versions of IP. IP version 5 is an experimental Stream Transport (ST) protocol used for real-time data delivery. IPv5 never came into operational use. IPv6 is an IP standard that provides greatly expanded addressing capacity. Because IPv6 uses a completely different address structure, it is not interoperable with IPv4. While IPv6 is a standard version of IP, it is not yet widely used in operational, commercial networks. Since our focus is on practical, operational networks, we do not cover IPv6 in detail. In this chapter and throughout the main body of the text, “IP” refers to IPv4. IPv4 is the protocol you will configure on your system when you want to exchange data with other systems, and it is the focus of this text.
The Internet Protocol is the heart of TCP/IP. IP provides the basic packet delivery service on which TCP/IP networks are built. All protocols, in the layers above and below IP, use the Internet Protocol to deliver data. All incoming and outgoing TCP/IP data flows through IP, regardless of its final destination.

Internet Protocol

The Internet Protocol is the building block of the Internet. Its functions include:

- Defining the datagram, which is the basic unit of transmission in the Internet
- Defining the Internet addressing scheme
- Moving data between the Network Access Layer and the Transport Layer
- Routing datagrams to remote hosts
- Performing fragmentation and re-assembly of datagrams

Before describing these functions in more detail, let’s look at some of IP’s characteristics. First, IP is a connectionless protocol. This means that it does not exchange control information (called a “handshake”) to establish an end-to-end connection before transmitting data. In contrast, a connection-oriented protocol exchanges control information with the remote system to verify that it is ready to receive data before any data is sent. When the handshaking is successful, the systems are said to have established a connection. The Internet Protocol relies on protocols in other layers to establish the connection if they require connection-oriented service.

IP also relies on protocols in the other layers to provide error detection and error recovery. The Internet Protocol is sometimes called an unreliable protocol because it contains no error detection and recovery code. This is not to say that the protocol cannot be relied on—quite the contrary. IP can be relied upon to accurately deliver your data to the connected network, but it doesn’t check whether that data was correctly received. Protocols in other layers of the TCP/IP architecture provide this checking when it is required.

The datagram

The TCP/IP protocols were built to transmit data over the ARPAnet, which was a packet-switching network. A packet is a block of data that carries with it the information necessary to deliver it, similar to a postal letter, which has an address written on its envelope. A packet-switching network uses the addressing information in the packets to switch packets from one physical network to another, moving them toward their final destination. Each packet travels the network independently of any other packet.

The datagram is the packet format defined by the Internet Protocol. Figure 1-5 is a pictorial representation of an IP datagram. The first five or six 32-bit words of the datagram are control information called the header. By default, the header is five words long; the sixth word is optional. Because the header’s length is variable, it
includes a field called Internet Header Length (IHL) that indicates the header’s length in words. The header contains all the information necessary to deliver the packet.

The Internet Protocol delivers the datagram by checking the Destination Address in word 5 of the header. The Destination Address is a standard 32-bit IP address that identifies the destination network and the specific host on that network. (The format of IP addresses is explained in Chapter 2.) If the Destination Address is the address of a host on the local network, the packet is delivered directly to the destination. If the Destination Address is not on the local network, the packet is passed to a gateway for delivery. Gateways are devices that switch packets between the different physical networks. Deciding which gateway to use is called routing. IP makes the routing decision for each individual packet.

Routing datagrams

Internet gateways are commonly (and perhaps more accurately) referred to as IP routers because they use Internet Protocol to route packets between networks. In traditional TCP/IP jargon, there are only two types of network devices—gateways and hosts. Gateways forward packets between networks, and hosts don’t. However, if a host is connected to more than one network (called a multi-homed host), it can forward packets between the networks. When a multi-homed host forwards packets, it acts just like any other gateway and is in fact considered to be a gateway. Current data communications terminology makes a distinction between gateways and routers,* but we’ll use the terms gateway and IP router interchangeably.

* In current terminology, a gateway moves data between different protocols, and a router moves data between different networks. So a system that routes X.400 packets over TCP/IP and X.400 is a gateway, but a traditional IP gateway is a router.
Figure 1-6 shows the use of gateways to forward packets. The hosts (or end systems) process packets through all four protocol layers, while the gateways (or intermediate systems) process the packets only up to the Internet Layer where the routing decisions are made.

Systems can deliver packets only to other devices attached to the same physical network. Packets from A1 destined for host C1 are forwarded through gateways G1 and G2. Host A1 first delivers the packet to gateway G1, with which it shares network A. Gateway G1 delivers the packet to G2 over network B. Gateway G2 then delivers the packet directly to host C1 because they are both attached to network C. Host A1 has no knowledge of any gateways beyond gateway G1. It sends packets destined for both networks C and B to that local gateway and then relies on that gateway to properly forward the packets along the path to their destinations. Likewise, host C1 sends its packets to G2 to reach a host on network A, as well as any host on network B.

Figure 1-7 shows another view of routing. This figure emphasizes that the underlying physical networks a datagram travels through may be different and even incompatible. Host A1 on the token ring network routes the datagram through gateway G1 to reach host C1 on the Ethernet. Gateway G1 forwards the data through the X.25 network to gateway G2 for delivery to C1. The datagram traverses three physically different networks, but eventually arrives intact at C1.

**Fragmenting datagrams**

As a datagram is routed through different networks, it may be necessary for the IP module in a gateway to divide the datagram into smaller pieces. A datagram received from one network may be too large to be transmitted in a single packet on a different network. This condition occurs only when a gateway interconnects dissimilar physical networks.
Each type of network has a *maximum transmission unit* (MTU), which is the largest packet that it can transfer. If the datagram received from one network is longer than the other network’s MTU, the datagram must be divided into smaller *fragments* for transmission. This process is called *fragmentation*. Think of a train delivering a load of steel. Each railway car can carry more steel than the trucks that will take it along the highway, so each railway car’s load is unloaded onto many different trucks. In the same way that a railroad is physically different from a highway, an Ethernet is physically different from an X.25 network; IP must break an Ethernet’s relatively large packets into smaller packets before it can transmit them over an X.25 network.

The format of each fragment is the same as the format of any normal datagram. Header word 2 contains information that identifies each datagram fragment and provides information about how to re-assemble the fragments back into the original datagram. The Identification field identifies what datagram the fragment belongs to, and the Fragmentation Offset field tells what piece of the datagram this fragment is. The Flags field has a “More Fragments” bit that tells IP if it has assembled all of the datagram fragments.

**Passing datagrams to the transport layer**

When IP receives a datagram that is addressed to the local host, it must pass the data portion of the datagram to the correct Transport Layer protocol. This is done by...
using the *protocol number* from word 3 of the datagram header. Each Transport Layer protocol has a unique protocol number that identifies it to IP. Protocol numbers are discussed in Chapter 2.

You can see from this short overview that IP performs many important functions. Don’t expect to fully understand datagrams, gateways, routing, IP addresses, and all the other things that IP does from this short description; each chapter will add more details about these topics. So let’s continue on with the other protocol in the TCP/IP Internet Layer.

**Internet Control Message Protocol**

An integral part of IP is the *Internet Control Message Protocol* (ICMP) defined in RFC 792. This protocol is part of the Internet Layer and uses the IP datagram delivery facility to send its messages. ICMP sends messages that perform the following control, error reporting, and informational functions for TCP/IP:

**Flow control**

When datagrams arrive too fast for processing, the destination host or an intermediate gateway sends an ICMP Source Quench Message back to the sender. This tells the source to stop sending datagrams temporarily.

**Detecting unreachable destinations**

When a destination is unreachable, the system detecting the problem sends a Destination Unreachable Message to the datagram’s source. If the unreachable destination is a network or host, the message is sent by an intermediate gateway. But if the destination is an unreachable port, the destination host sends the message. (We discuss ports in Chapter 2.)

**Redirecting routes**

A gateway sends the ICMP Redirect Message to tell a host to use another gateway, presumably because the other gateway is a better choice. This message can be used only when the source host is on the same network as both gateways. To better understand this, refer to Figure 1-7. If a host on the X.25 network sent a datagram to G1, it would be possible for G1 to redirect that host to G2 because the host, G1, and G2 are all attached to the same network. On the other hand, if a host on the token ring network sent a datagram to G1, the host could not be redirected to use G2. This is because G2 is not attached to the token ring.

**Checking remote hosts**

A host can send the ICMP Echo Message to see if a remote system’s Internet Protocol is up and operational. When a system receives an echo message, it replies and sends the data from the packet back to the source host. The *ping* command uses this message.
About the Author

Craig Hunt has worked with computer systems for the last 25 years. His first computer job was as a programmer and systems programmer for the federal government. He left the government to work for Honeywell on the WWMCCS network in the days before TCP/IP, back when the global network used NCP. After Honeywell, Craig went to work for the National Institute of Standards and Technology (NIST) where he built their first enterprise TCP/IP network, administered the central servers on that network, and eventually moved into network research. Craig left NIST to work full time writing and teaching about Linux, Unix, and networking. In addition to TCP/IP Network Administration, Craig has written four other books, co-authored two, and edited five. He teaches Linux, Unix, and networking tutorials at major conferences such as USENIX and LinuxWorld. To find out more about what he is doing, visit his web site at http://www.wrotethebook.com.

Craig lives with his wife and youngest daughter in Gaithersburg, Maryland. He loves the outdoors, and has a newly discovered passion for exploring it on his mountain bike.

Colophon

Our look is the result of reader comments, our own experimentation, and feedback from distribution channels. Distinctive covers complement our distinctive approach to technical topics, breathing personality and life into potentially dry subjects.

The animal on the cover of TCP/IP Network Administration is a land crab. Land crabs are found in tropical America, West Africa, and the Indo-Pacific region where they can be found living in burrows in fields, swamps, and mangrove thickets. They occasionally are found as far as five miles inland, returning to the sea to spawn. Land crabs are a subgroup of over 4,500 species of crabs. Classified with shrimp, lobster, and crayfish, crabs differ from these in their tail structure. Unlike the rest of their order, crabs’ tails are curled under their thorax. In addition, their carapaces tend to be unusually broad. Though land crabs in the United States commonly grow to weigh no more than 18 ounces and measure 4 or 5 inches across, crabs in general range in size from less than a centimeter across to the largest, the Japanese spider crab, whose claws can span 12 feet.

Emily Quill was the production editor and copyeditor for TCP/IP Network Administration, Third Edition. Jeffrey Holcomb and Jane Ellin provided quality control. Derek Di Matteo and Sue Willing provided production assistance. Tom Dinse wrote the index.

Edie Freedman designed the cover of this book, using a 19th-century engraving from the Dover Pictorial Archive. Emma Colby produced the cover layout with Quark-XPress 4.1 using Adobe’s ITC Garamond font.
Melanie Wang designed the interior layout, based on a series design by David Futato. Neil Walls converted the files from Microsoft Word to FrameMaker 5.5.6 using tools created by Mike Sierra. The text font is Linotype Birka; the heading font is Adobe Myriad Condensed; and the code font is LucasFont’s TheSans Mono Condensed. The illustrations that appear in the book were produced by Robert Romano and Jessamyn Read using Macromedia FreeHand 9 and Adobe Photoshop 6. The tip and warning icons were drawn by Christopher Bing.