



Kirkland & Ellis LLP Library

5 0628 01267166 2

TCP/IP Illustrated, Volume 1

The Protocols

W. Richard Stevens



ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and we were aware of a trademark claim, the designations have been printed in initial capital letters or in all capitals.

The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

The publisher offers discounts on this book when ordered in quantity for special sales. For more information, please contact:

Pearson Education Corporate Sales Division
201 W. 103rd Street
Indianapolis, IN 46290
(800) 428-5331
corpsales@pearsoned.com

Visit AW on the Web: www.awl.com/cseng/

Library of Congress Cataloging-in-Publication Data
Stevens, W. Richard

TCP/IP Illustrated: the protocols/W. Richard Stevens.
p. cm.—(Addison-Wesley professional computing series)
Includes bibliographical references and index.

ISBN 0-201-63346-9 (v.1)

I. TCP/IP (Computer network protocol) I. Title. II. Series.

TK5105.55S74 1994

004.6'2—dc20

Copyright © 1994 by Addison Wesley

UNIX is a technology trademark of X/Open Company, Ltd.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or other-wise, without the prior consent of the publisher. Printed in the United States of America. Published simultaneously in Canada.

Text printed on recycled and acid-free paper.

ISBN 0201633469

26 2728293031 HT 07 06 05

26th Printing February 2005

Preface

Introduction

This book describes the TCP/IP protocol suite, but from a different perspective than other texts on TCP/IP. Instead of just describing the protocols and what they do, we'll use a popular diagnostic tool to watch the protocols in action. Seeing how the protocols operate in varying circumstances provides a greater understanding of how they work and why certain design decisions were made. It also provides a look into the implementation of the protocols, without having to wade through thousands of lines of source code.

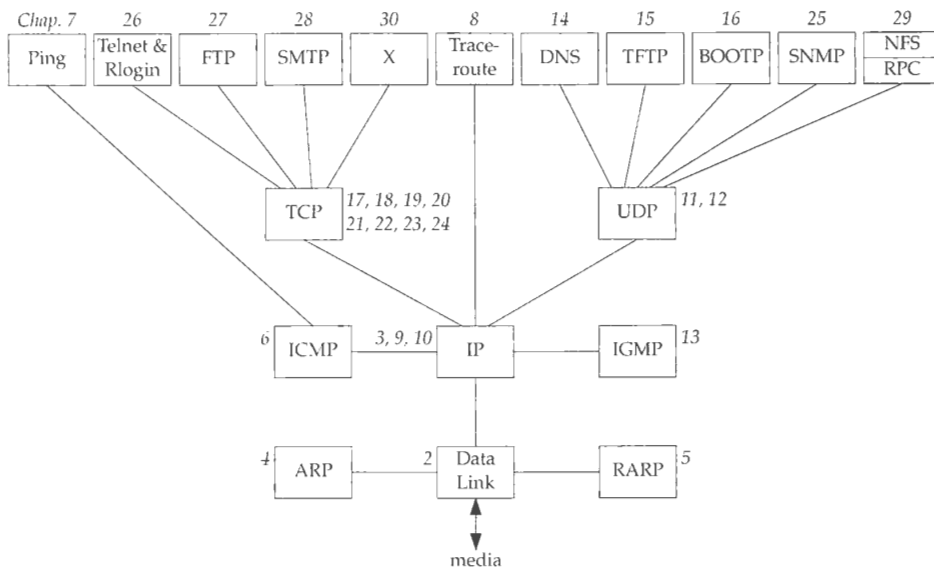
When networking protocols were being developed in the 1960s through the 1980s, expensive, dedicated hardware was required to see the packets going "across the wire." Extreme familiarity with the protocols was also required to comprehend the packets displayed by the hardware. Functionality of the hardware analyzers was limited to that built in by the hardware designers.

Today this has changed dramatically with the ability of the ubiquitous workstation to monitor a local area network [Mogul 1990]. Just attach a workstation to your network, run some publicly available software (described in Appendix A), and watch what goes by on the wire. While many people consider this a tool to be used for *diagnosing* network problems, it is also a powerful tool for *understanding* how the network protocols operate, which is the goal of this book.

This book is intended for anyone wishing to understand how the TCP/IP protocols operate: programmers writing network applications, system administrators responsible for maintaining computer systems and networks utilizing TCP/IP, and users who deal with TCP/IP applications on a daily basis.

Organization of the Book

The following figure shows the various protocols and applications that are covered. The italic number by each box indicates the chapter in which that protocol or application is described.



(Numerous fine points are missing from this figure that will be discussed in the appropriate chapter. For example, both the DNS and RPC use TCP, which we don't show.)

We take a bottom-up approach to the TCP/IP protocol suite. After providing a basic introduction to TCP/IP in Chapter 1, we will start at the link layer in Chapter 2 and work our way up the protocol stack. This provides the required background for later chapters for readers who aren't familiar with TCP/IP or networking in general.

This book also uses a functional approach instead of following a strict bottom-to-top order. For example, Chapter 3 describes the IP layer and the IP header. But there are numerous fields in the IP header that are best described in the context of an application that uses or is affected by a particular field. Fragmentation, for example, is best understood in terms of UDP (Chapter 11), the protocol often affected by it. The time-to-live field is fully described when we look at the Traceroute program in Chapter 8, because this field is the basis for the operation of the program. Similarly, many features of ICMP are described in the later chapters, in terms of how a particular ICMP message is used by a protocol or an application.

We also don't want to save all the good stuff until the end, so we describe TCP/IP applications as soon as we have the foundation to understand them. Ping and Traceroute are described after IP and ICMP have been discussed. The applications built on UDP (multicasting, the DNS, TFTP, and BOOTP) are described after UDP has been

examined. The TCP applications, however, along with network management, must be saved until the end, after we've thoroughly described TCP. This text focuses on how these applications use the TCP/IP protocols. We do not provide all the details on running these applications.

Readers

This book is self-contained and assumes no specific knowledge of networking or TCP/IP. Numerous references are provided for readers interested in additional details on specific topics.

This book can be used in many ways. It can be used as a self-study reference and covered from start to finish by someone interested in all the details on the TCP/IP protocol suite. Readers with some TCP/IP background might want to skip ahead and start with Chapter 7, and then focus on the specific chapters in which they're interested. Exercises are provided at the end of the chapters, and most solutions are in Appendix D. This is to maximize the usefulness of the text as a self-study reference.

When used as part of a one- or two-semester course in computer networking, the focus should be on IP (Chapters 3 and 9), UDP (Chapter 11), and TCP (Chapters 17–24), along with some of the application chapters.

Many forward and backward references are provided throughout the text, along with a thorough index, to allow individual chapters to be studied by themselves. A list of all the acronyms used throughout the text, along with the compound term for the acronym, appears on the inside back covers.

If you have access to a network you are encouraged to obtain the software used in this book (Appendix F) and experiment on your own. Hands-on experimentation with the protocols will provide the greatest knowledge (and make it more fun).

Systems Used for Testing

Every example in the book was run on an actual network and the resulting output saved in a file for inclusion in the text. Figure 1.11 (p. 18) shows a diagram of the different hosts, routers, and networks that are used. (This figure is also duplicated on the inside front cover for easy reference while reading the book.) This collection of networks is simple enough that the topology doesn't confuse the examples, and with four systems acting as routers, we can see the error messages generated by routers.

Most of the systems have a name that indicates the type of software being used: `bsdi`, `svr4`, `sun`, `solaris`, `aix`, `slip`, and so on. In this way we can identify the type of software that we're dealing with by looking at the system name in the printed output.

A wide range of different operating systems and TCP/IP implementations are used:

- BSD/386 Version 1.0 from Berkeley Software Design, Inc., on the hosts named `bsdi` and `slip`. This system is derived from the BSD Networking Software, Release 2.0. (We show the lineage of the various BSD releases in Figure 1.10 on p. 17.)

- Unix System V/386 Release 4.0 Version 2.0 from U.H. Corporation, on the host named `svr4`. This is vanilla SVR4 and contains the standard implementation of TCP/IP from Lachman Associates used with most versions of SVR4.
- SunOS 4.1.3 from Sun Microsystems, on the host named `sun`. The SunOS 4.1.x systems are probably the most widely used TCP/IP implementations. The TCP/IP code is derived from 4.2BSD and 4.3BSD.
- Solaris 2.2 from Sun Microsystems, on the host named `solaris`. The Solaris 2.x systems have a different implementation of TCP/IP from the earlier SunOS 4.1.x systems, and from SVR4. (This operating system is really SunOS 5.2, but is commonly called Solaris 2.2.)
- AIX 3.2.2 from IBM on the host named `aix`. The TCP/IP implementation is based on the 4.3BSD Reno release.
- 4.4BSD from the Computer Systems Research Group at the University of California at Berkeley, on the host `vangogh.cs.berkeley.edu`. This system has the latest release of TCP/IP from Berkeley. (This system isn't shown in the figure on the inside front cover, but is reachable across the Internet.)

Although these are all Unix systems, TCP/IP is operating system independent, and is available on almost every popular non-Unix system. Most of this text also applies to these non-Unix implementations, although some programs (such as Traceroute) may not be provided on all systems.

Typographical Conventions

When we display interactive input and output we'll show our typed input in a **bold font**, and the computer output like *this*. *Comments are added in italics.*

```
bsdi % telnet svr4 discard      connect to the discard server
Trying 140.252.13.34...        this line and next output by Telnet client
Connected to svr4.
```

Also, we always include the name of the system as part of the shell prompt (`bsdi` in this example) to show on which host the command was run.

Throughout the text we'll use indented, parenthetical notes such as *this* to describe historical points or implementation details.

We sometimes refer to the complete description of a command in the Unix manual as in `ifconfig(8)`. This notation, the name of the command followed by a number in parentheses, is the normal way of referring to Unix commands. The number in parentheses is the section number in the Unix manual of the "manual page" for the command, where additional information can be located. Unfortunately not all Unix systems organize their manuals the same, with regard to the section numbers used for various groupings of commands. We'll use the BSD-style section numbers (which is the same for BSD-derived systems such as SunOS 4.1.3), but your manuals may be organized differently.

Introduction

1.1 Introduction

The TCP/IP protocol suite allows computers of all sizes, from many different computer vendors, running totally different operating systems, to communicate with each other. It is quite amazing because its use has far exceeded its original estimates. What started in the late 1960s as a government-financed research project into packet switching networks has, in the 1990s, turned into the most widely used form of networking between computers. It is truly an *open system* in that the definition of the protocol suite and many of its implementations are publicly available at little or no charge. It forms the basis for what is called the *worldwide Internet*, or the *Internet*, a wide area network (WAN) of more than one million computers that literally spans the globe.

This chapter provides an overview of the TCP/IP protocol suite, to establish an adequate background for the remaining chapters. For a historical perspective on the early development of TCP/IP see [Lynch 1993].

1.2 Layering

Networking *protocols* are normally developed in *layers*, with each layer responsible for a different facet of the communications. A *protocol suite*, such as TCP/IP, is the combination of different protocols at various layers. TCP/IP is normally considered to be a 4-layer system, as shown in Figure 1.1.

Application	Telnet, FTP, e-mail, etc.
Transport	TCP, UDP
Network	IP, ICMP, IGMP
Link	device driver and interface card

Figure 1.1 The four layers of the TCP/IP protocol suite.

Each layer has a different responsibility.

1. The *link* layer, sometimes called the *data-link* layer or *network interface* layer, normally includes the device driver in the operating system and the corresponding network interface card in the computer. Together they handle all the hardware details of physically interfacing with the cable (or whatever type of media is being used).
2. The *network* layer (sometimes called the *internet* layer) handles the movement of packets around the network. Routing of packets, for example, takes place here. IP (Internet Protocol), ICMP (Internet Control Message Protocol), and IGMP (Internet Group Management Protocol) provide the network layer in the TCP/IP protocol suite.
3. The *transport* layer provides a flow of data between two hosts, for the application layer above. In the TCP/IP protocol suite there are two vastly different transport protocols: TCP (Transmission Control Protocol) and UDP (User Datagram Protocol).

TCP provides a reliable flow of data between two hosts. It is concerned with things such as dividing the data passed to it from the application into appropriately sized chunks for the network layer below, acknowledging received packets, setting timeouts to make certain the other end acknowledges packets that are sent, and so on. Because this reliable flow of data is provided by the transport layer, the application layer can ignore all these details.

UDP, on the other hand, provides a much simpler service to the application layer. It just sends packets of data called *datagrams* from one host to the other, but there is no guarantee that the datagrams reach the other end. Any desired reliability must be added by the application layer.

There is a use for each type of transport protocol, which we'll see when we look at the different applications that use TCP and UDP.

4. The *application* layer handles the details of the particular application. There are many common TCP/IP applications that almost every implementation provides:

- Telnet for remote login,
- FTP, the File Transfer Protocol,
- SMTP, the Simple Mail Transfer protocol, for electronic mail,
- SNMP, the Simple Network Management Protocol,

and many more, some of which we cover in later chapters.

If we have two hosts on a local area network (LAN) such as an Ethernet, both running FTP, Figure 1.2 shows the protocols involved.

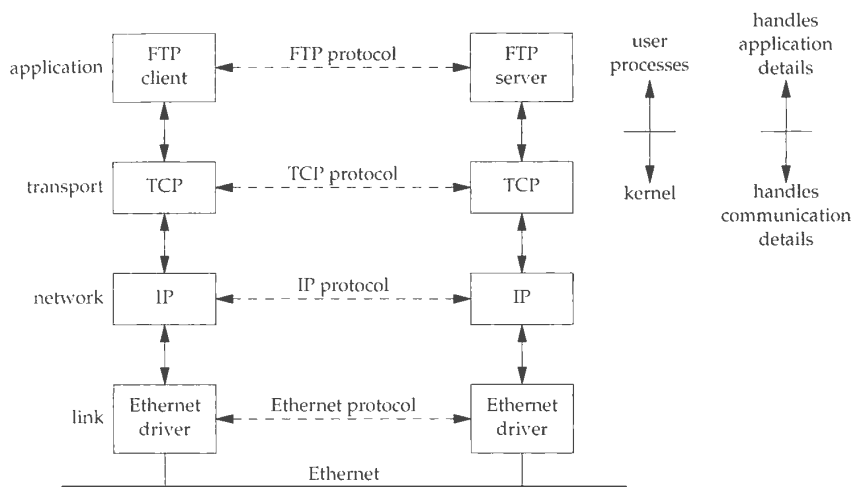


Figure 1.2 Two hosts on a LAN running FTP.

We have labeled one application box the *FTP client* and the other the *FTP server*. Most network applications are designed so that one end is the client and the other side the server. The server provides some type of service to clients, in this case access to files on the server host. In the remote login application, Telnet, the service provided to the client is the ability to login to the server's host.

Each layer has one or more protocols for communicating with its *peer* at the same layer. One protocol, for example, allows the two TCP layers to communicate, and another protocol lets the two IP layers communicate.

On the right side of Figure 1.2 we have noted that normally the application layer is a user process while the lower three layers are usually implemented in the kernel (the operating system). Although this isn't a requirement, it's typical and this is the way it's done under Unix.

1.8 Client–Server Model

Most networking applications are written assuming one side is the client and the other the server. The purpose of the application is for the server to provide some defined service for clients.

We can categorize servers into two classes: iterative or concurrent. An *iterative server* iterates through the following steps.

- I1. Wait for a client request to arrive.
- I2. Process the client request.
- I3. Send the response back to the client that sent the request.
- I4. Go back to step I1.

The problem with an iterative server is when step I2 takes a while. During this time no other clients are serviced.

A *concurrent server*, on the other hand, performs the following steps.

- C1. Wait for a client request to arrive.
- C2. Start a new server to handle this client's request. This may involve creating a new process, task, or thread, depending on what the underlying operating system supports. How this step is performed depends on the operating system.

This new server handles this client's entire request. When complete, this new server terminates.
- C3. Go back to step C1.

The advantage of a concurrent server is that the server just spawns other servers to handle the client requests. Each client has, in essence, its own server. Assuming the operating system allows multiprogramming, multiple clients are serviced concurrently.

The reason we categorize servers, and not clients, is because a client normally can't tell whether it's talking to an iterative server or a concurrent server.

As a general rule, TCP servers are concurrent, and UDP servers are iterative, but there are a few exceptions. We'll look in detail at the impact of UDP on its servers in Section 11.12, and the impact of TCP on its servers in Section 18.11.

1.9 Port Numbers

We said that TCP and UDP identify applications using 16-bit port numbers. How are these port numbers chosen?

Servers are normally known by their *well-known* port number. For example, every TCP/IP implementation that provides an FTP server provides that service on TCP port

Name	TCP port	UDP port	RFC	Description
echo	7	7	862	Server returns whatever the client sends.
discard	9	9	863	Server discards whatever the client sends.
daytime	13	13	867	Server returns the time and date in a human-readable format.
chargen	19	19	864	TCP server sends a continual stream of characters, until the connection is terminated by the client. UDP server sends a datagram containing a random number of characters each time the client sends a datagram.
time	37	37	868	Server returns the time as a 32-bit binary number. This number represents the number of seconds since midnight January 1, 1900, UTC.

Figure 1.9 Standard, simple services provided by most implementations.

1.13 The Internet

In Figure 1.3 we showed an *internet* composed of two networks—an Ethernet and a token ring. In Sections 1.4 and 1.9 we talked about the worldwide *Internet* and the need to allocate IP addresses centrally (the InterNIC) and the well-known port numbers (the IANA). The word *internet* means different things depending on whether it’s capitalized or not.

The lowercase *internet* means multiple networks connected together, using a common protocol suite. The uppercase *Internet* refers to the collection of hosts (over one million) around the world that can communicate with each other using TCP/IP. While the Internet is an internet, the reverse is not true.

1.14 Implementations

The de facto standard for TCP/IP implementations is the one from the Computer Systems Research Group at the University of California at Berkeley. Historically this has been distributed with the 4.x BSD system (Berkeley Software Distribution), and with the “BSD Networking Releases.” This source code has been the starting point for many other implementations.

Figure 1.10 shows a chronology of the various BSD releases, indicating the important TCP/IP features. The BSD Networking Releases shown on the left side are publicly available source code releases containing all of the networking code: both the protocols themselves and many of the applications and utilities (such as Telnet and FTP).

Throughout the text we’ll use the term *Berkeley-derived implementation* to refer to vendor implementations such as SunOS 4.x, SVR4, and AIX 3.2 that were originally developed from the Berkeley sources. These implementations have much in common, often including the same bugs!

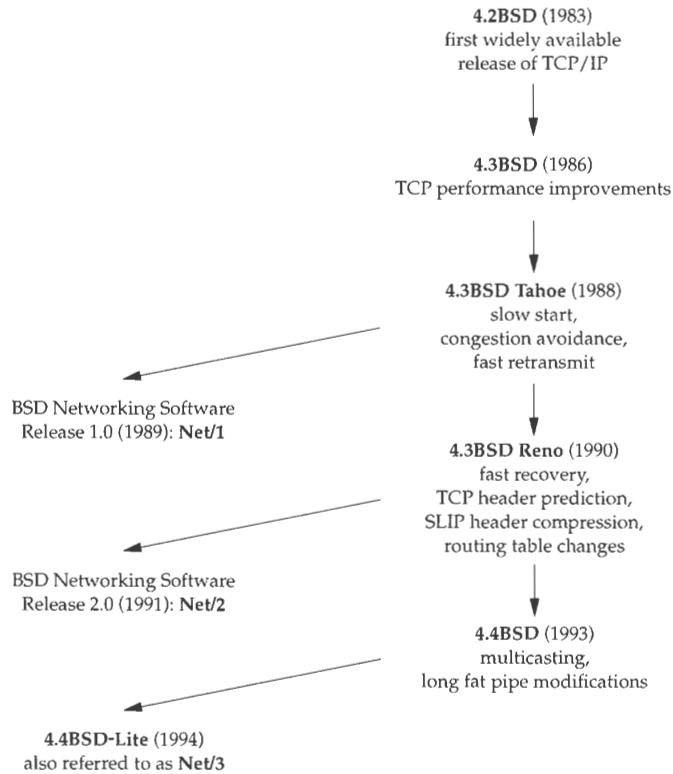


Figure 1.10 Various BSD releases with important TCP/IP features.

Much of the original research in the Internet is still being applied to the Berkeley system—new congestion control algorithms (Section 21.7), multicasting (Section 12.4), “long fat pipe” modifications (Section 24.3), and the like.

1.15 Application Programming Interfaces

Two popular *application programming interfaces* (APIs) for applications using the TCP/IP protocols are called *sockets* and *TLI* (Transport Layer Interface). The former is sometimes called “Berkeley sockets,” indicating where it was originally developed. The latter, originally developed by AT&T, is sometimes called *XTI* (X/Open Transport Interface), recognizing the work done by X/Open, an international group of computer vendors that produce their own set of standards. XTI is effectively a superset of TLI.