Third Edition

Includes IPng and IP over ATM

INTERNETWORKING WITH TCP/IP

VOLUME I

PRINCIPLES, PROTOCOLS, AND ARCHITECTURE

DOUGLAS E. COMER

Third Edition

INTERNETWORKING WITH TCP/IP

VOLUME I
PRINCIPLES, PROTOCOLS, AND ARCHITECTURE

DOUGLAS E. COMER

Over 200,000 Copies Sold

"*THE* classic text for an introduction to TCP/IP."
—Jon Postel, RFC editor and former Deputy Internet Architect

"Although others have tried, there is no better written or organized explanation of the core of TCP/IP."
—Joel Snyder, *Network Computing*

"As an introduction to the TCP/IP protocol suite and its underpinnings, this is an excellent book. It is also a good reference book to keep around for anyone who is working with TCP/IP."
—George V. Neville-Neil, *USENIX ;login:*

The all-time best-selling TCP/IP book, *Internetworking with TCP/IP*, is still THE reference for anyone who wants to learn about or work with the TCP/IP protocol suite. Volume I of the series by Douglas Comer provides the most up-to-date conceptual introduction to TCP/IP protocols and the latest developments in Internet technology.

Renowned for its clarity and accessibility, this superb text covers wide area (WAN) Internet backbones as well as local area network (LAN) technologies like Ethernet and FDDI. The text explains address binding (ARP), IP connectionless datagram delivery, error detection, multicasting, and routing.

THIS NEW EDITION OF VOLUME I:

- Discusses how to use TCP/IP over an ATM network.
- Covers the latest IPng (next generation) developments and information.
- Describes CIDR (Classless Inter-Domain Routing) and supernetting.
- Discusses security in TCP/IP environments and firewall design.
- Categorizes hundreds of new RFCs and the protocols they describe.

In addition, Volume I:

- Compares the ISO 7-layer reference model to the TCP/IP 5-layer reference model.
- Explains TCP: reliability, acknowledgments, flow control, and sliding windows.
- Details adaptive retransmission, including slow-start and silly window avoidance.
- Describes the socket interface that applications use to access TCP/IP protocols.
- Presents routing architectures for large and small internets.
- Discusses bridges and routers.
- Examines application services:
  - Domain Name System (DNS)
  - Electronic mail (SMTP, MIME)
  - File transfer and access (FTP, TFTP, NFS)
  - Remote login (TELENET, rlogin)
  - Network management (SNMP, MIB, ANS.1)

PRENTICE HALL
Upper Saddle River, NJ 07458

ISBN 0-13-216987-8
90000
9 780132 169875

Internetworking With TCP/IP

Internetworking With TCP/IP
Vol I:
Principles, Protocols, and Architecture
Third Edition

DOUGLAS E. COMER

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

PRENTICE HALL
Upper Saddle River, New Jersey 07458

Library of Congress Cataloging-in-Publication Data

Comer, Douglas
Internetworking with TCP/IP / Douglas E. Comer. -- 3rd ed.
p. cm.
Includes bibliographical references and index.
Contents: v. 1. Principles, protocols, and architecture
ISBN 0-13-216987-8 (v. 1)
1. TCP/IP (Computer network protocol) 2. Client/server computing.
3. Internetworking (Telecommunication) I. Title.
TK5105.585.C66 1995
005.2--dc20 95-1830
CIP

Acquisitions editor: ALAN APT
Production editor: IRWIN ZUCKER
Cover designer: WENDY ALLING JUDY
Buyer: LORI BULWIN
Editorial assistant: SHIRLEY MCGUIRE

Printed in the United States of America

10 9 8 7 6 5 4

ISBN 0-13-216987-8

Prentice-Hall International (UK) Limited, London
Prentice-Hall of Australia Pty. Limited, Sydney
Prentice-Hall Canada Inc., Toronto
Prentice-Hall Hispanoamericana, S.A., Mexico
Prentice-Hall of India Private Limited, New Delhi
Prentice-Hall of Japan, Inc., Tokyo
Simon & Schuster Asia Pte. Ltd., Singapore
Editora Prentice-Hall do Brasil, Ltda., Rio de Janeiro

To Chris

Additional Enthusiastic Comments About
Internetworking With TCP/IP Volume 1

"Unquestionably THE reference for TCP/IP; both informative and easy to read, this book is liked by both novice and experienced."

– Raj Yavatkar
University of Kentucky
US Editor, Computer Communications

"The third edition maintains Comer's Internetworking with TCP/IP as the acknowledged leader in TCP/IP books by adding up-to-the-minute material on ATM, CIDR, firewalls, DHCP and the next version of IP, IPng."

– Ralph Droms
Bucknell University
IFTF Working Group Chair

"Doug Comer remains the first and best voice of Internet technology. Despite the legion of 'Internet carpetbaggers' (the current crop of 'authors' who can barely spell F-T-P) which contributes noise – but no knowledge – on the Internet and its infrastructure, Dr. Comer shines through as the premiere source for lucid explanations and accurate information. He sets a standard for which many strive, but precious few attain."

– Marshall Rose
Dover Beach Consulting
IETF Area Director

"Comer's Volume 1 drastically changed the course of networking history."

– Dan Lynch
Interop Company
IAB Member

"When you need to teach the details of TCP/IP, you need the latest information. Once again, Comer separates the chaff from the wheat with his latest edition of the TCP/IP book that a generation of networkers grew up with."

– Shawn Ostermann
Ohio University

Contents

Foreword

Professor Douglas Comer's book has become *the* classic text for an introduction to TCP/IP. Writing an introduction to TCP/IP for the uninitiated is a very difficult task. While combining the explanation of the general principles of computer communication with the specific examples from the TCP/IP protocol suite, Doug Comer has provided a very readable book.

While this book is specifically about the TCP/IP protocol suite, it is a good book for learning about computer communications protocols in general. The principles of architecture, layering, multiplexing, encapsulation, addressing and address mapping, routing, and naming are quite similar in any protocol suite, though, of course, different in detail.

Computer communication protocols do not do anything themselves. Like operating systems, they are in the service of application processes. Processes are the active elements that request communication and are the ultimate senders and receivers of the data transmitted. The various layers of protocols are like the various layers in a computer operating system, especially the file system. Understanding protocol architecture is like understanding operating system architecture. In this book Doug Comer has taken the ''bottom up'' approach – starting with the physical networks and moving up in levels of abstraction to the applications.

Since application processes are the active elements using the communication supported by the protocols, TCP/IP is an ''interprocess communication'' (IPC) mechanism. While there are several experiments in progress with operating system style message passing and procedure call types of IPC based on IP, the focus in this book is on more traditional applications that use the UDP datagram or TCP logical connection forms of IPC. Typically in operating systems there is a set of functions provided by the operating system to the application processes. This system call interface usually includes calls for opening, reading, writing, and closing files, among other things. In many systems there are similar system calls for IPC functions including network communication. As an example of such an interface Doug Comer presents an overview of the socket interface.

One of the key ideas inherent in TCP/IP and in the title of this book is ''internetworking.'' The power of a communication system is directly related to the number of entities in that system. The telephone network is very useful because (nearly) all the telephones are connected to one network (as it appears to the users). Computer communication systems and networks are currently separated and fragmented. As more users and enterprises adopt TCP/IP as their network communication technology and are joining the Internet this is becoming less of a problem, but there is still a long way to go. The goal of interconnection and internetworking, to have a single powerful computer communication network, is fundamental to the design of TCP/IP.

Essential to internetworking is addressing, and a universal protocol – the Internet Protocol. Of course, the individual networks have their own protocols which are used to carry the IP datagrams, and there must be a mapping between the individual network address and the IP address. Over the lifetime of TCP/IP, the nature of these individual networks have changed from the early days of the ARPANET to the recently developed ATM networks. A new chapter in this edition discusses IP over ATM networks. This book now includes recent developments in Dynamic Host Configuration (DHCP) that will ease the administration of networks and the installation of new computers.

To have an internetwork, the individual networks must be connected. The connecting devices are called routers. Further, these routers must have some procedures for forwarding data from one network to the next. The data is in the form of IP datagrams and the destination is specified by an IP address, but the router must make a routing decision based on the IP address and what it knows about the connectivity of the networks making up the Internet. The procedures for distributing the current connectivity information to the routers are called routing algorithms, and these are currently the subject of much study and development. In particular, the recent development of the Classless InterDomain Routing (CIDR) technique to reduce the amount of routing information exchanged is important.

Like all communication systems, the TCP/IP protocol suite is an unfinished system. It is evolving to meet changing requirements and new opportunities. Thus, this book is, in a sense, a snapshot of TCP/IP. And, as Doug Comer points out, there are many loose ends. With the recent rapid growth of the Internet there is concern about it outgrowing the capabilities of the TCP/IP protocols, particularly the address space. In response the research and engineering community has developed a ''next generation'' version of the Internet Protocol called IPng. Many of the enterprises now joining the Internet have concerns about security. A new chapter in this edition discusses the security and firewalls.

Most chapters end with a few pointers to material ''for further study.'' Many of these refer to memos of the RFC series of notes. This series of notes is the result of a policy of making the working ideas and the protocol specifications developed by the TCP/IP research and development community widely available. This availability of the basic and detailed information about these protocols, and the availability of the early implementations of them, has had much to do with their current widespread use. This commitment to public documentation at this level of detail is unusual for a research effort, and has had significant benefits for the development of computer communication.

This book brings together information about the various parts of the TCP/IP architecture and protocols and makes it accessible. Its publication is a very significant milestone in the evolution of computer communications.

Jon Postel,
Associate Director for Networking
Information Sciences Institute
University of Southern California

January 1995

Preface

The world has changed dramatically since the second edition of this book was published. It hardly seems possible only four years have elapsed. When I began the second edition in the summer of 1990, the Internet had grown to nearly 300,000 host computers, up from 5,000 hosts when the book was first written. At the time, we marveled at how large an obscure research project had become. Cynics predicted that continued growth would lead to a complete collapse by 1993. Instead of collapsing, the Internet has continued its explosive expansion; the ''large'' Internet of 1990 is only 7% of the current Internet.

TCP/IP and the Internet have accommodated change well. The basic technology has survived over a decade of exponential growth and the associated increases in traffic. The protocols have worked over new high-speed network technologies, and the design has handled applications that could not be imagined a decade ago. Of course, the entire protocol suite has not remained static. New protocols have been deployed, and new techniques have been developed to adapt existing protocols to new network technologies. Changes are documented in RFCs, which have increased by over 50 percent.

This edition contains updated information throughout the text (including use of the commercially popular term *IP router* in place of the traditional scientific term *IP gateway*) as well as new material that describes technical advances and changes. The chapter on subnet addressing now describes supernetting as well as subnetting, and shows how the two techniques are motivated by the same goal. The chapter on bootstrapping explains a significant advance that will eliminate the need for manual configuration of host computers and allow a computer to obtain an IP address automatically: the Dynamic Host Configuration Protocol (DHCP). The chapter on TCP includes a description of Silly Window Syndrome and an explanation of the heuristics TCP uses to prevent the problem. The chapter on electronic mail includes a description of the Multipurpose Internet Mail Extensions (MIME), which permit non-ASCII data to be sent in a standard e-mail message.

Three new chapters contain detailed information about significant developments. Chapter 18 explains how TCP/IP is being used over ATM networks. The chapter discusses the organization of ATM hardware, the purpose of adaptation layer protocols, IP encapsulation, address binding, routing, and virtual circuit management. The chapter illustrates how a connectionless protocol like IP can use the connection-oriented interface that ATM provides. Chapter 28 covers a topic that is crucial to many organizations as they contemplate connecting to the global Internet – security. The chapter describes the internet firewall concept, and shows how a firewall architecture can be used to protect networks and computers inside an organization from unwanted access. The chapter also discusses the principles underlying a two-level firewall design, and considers outside access from a secure computer. Finally, a new chapter is devoted to what may be the most significant change in TCP/IP since its inception: the imminent adoption of a next generation Internet Protocol (IPng). Chapter 29 describes the protocol that the IETF has developed to serve as IPng. Although it has not been thoroughly tested or approved as a permanent standard, the new design appears to be the consensus choice. The chapter presents the proposed design and address assignment scheme.

The third edition retains the same general contents and overall organization as the second edition. The entire text focuses on the concept of internetworking in general and the TCP/IP internet technology in particular. Internetworking is a powerful abstraction that allows us to deal with the complexity of multiple underlying communication technologies. It hides the details of network hardware and provides a high level communication environment. The text reviews both the architecture of network interconnections and the principles underlying protocols that make such interconnected networks function as a single, unified communication system. It also shows how an internet communication system can be used for distributed computation.

After reading this book, you will understand how it is possible to interconnect multiple physical networks into a coordinated system, how internet protocols operate in that environment, and how application programs use the resulting system. As a specific example, you will learn the details of the global TCP/IP Internet, including the architecture of its router system and the application protocols it supports. In addition, you will understand some of the limitations of the internet approach.

Designed as both a college text and as a professional reference, the book is written at an advanced undergraduate or graduate level. For professionals, the book provides a comprehensive introduction to the TCP/IP technology and the architecture of the Internet. Although it is not intended to replace protocol standards, the book is an excellent starting point for learning about internetworking because it provides a uniform overview that emphasizes principles. Moreover, it gives the reader perspective that can be extremely difficult to obtain from individual protocol documents.

When used in the classroom, the text provides more than sufficient material for a single semester network course at either the undergraduate or graduate level. Such a course can be extended to a two-semester sequence if accompanied by programming projects and readings from the literature. For undergraduate courses, many of the details are unnecessary. Students should be expected to grasp the basic concepts described in the text, and they should be able to describe or use them. At the graduate level, students should be expected to use the material here as a basis for further exploration. They should understand the details well enough to answer exercises or solve problems that require them to explore extensions and subtleties. Many of the exercises suggest such subtleties; solving them often requires students to read protocol standards and apply creative energy to comprehend consequences.

At all levels, hands-on experience sharpens the concepts and helps students gain intuition. Thus, I encourage instructors to invent projects that force students to use Internet services and protocols. The semester project in my graduate Internetworking course at Purdue requires students to build an IP router. We supply hardware and the source code for an operating system, including device drivers for network interfaces; students build a working router that interconnects three networks with different MTUs. The course is extremely rigorous, students work in teams, and the results have been impressive (many industries recruit graduates from the course). Although such experimentation is safest when the instructional laboratory network is isolated from production computing facilities, we have found that students exhibit the most enthusiasm, and benefit the most, when they have access to a functional TCP/IP internet.

The book is organized into four main parts. Chapters 1 and 2 form an introduction that provides an overview and discusses existing network technologies. In particular, Chapter 2 reviews physical network hardware. The intention is to provide basic intuition about what is possible, not to spend inordinate time on hardware details. Chapters 3-13 describe the TCP/IP Internet from the viewpoint of a single host, showing the protocols a host contains and how they operate. They cover the basics of Internet addressing and routing as well as the notion of protocol layering. Chapters 14-18 and 28 describe the architecture of an internet when viewed globally. They explore routing architecture and the protocols routers use to exchange routing information. Finally, Chapters 19-27 discuss application level services available in the Internet. They present the client-server model of interaction, and give several examples of client and server software.

The chapters have been organized bottom up. They begin with an overview of hardware and continue to build new functionality on top of it. This view will appeal to anyone who has developed Internet software because it follows the same pattern one uses in implementation. The concept of layering does not appear until Chapter 11. The discussion of layering emphasizes the distinction between conceptual layers of functionality and the reality of layered protocol software in which multiple objects appear at each layer.

A modest background is required to understand the material. The reader is expected to have a basic understanding of computer systems, and to be familiar with data structures like stacks, queues, and trees. Readers need basic intuition about the organization of computer software into an operating system that supports concurrent programming and application programs that users invoke to perform computation. Readers do not need sophisticated mathematics, nor do they need to know information theory or theorems from data communications; the book describes the physical network as a black box around which an internetwork can be built. It states design principles in English and discusses motivations and consequences.

I thank all the people who have contributed to versions of this book. John Lin provided extensive assistance with this edition, including classifying RFCs. Ralph Droms reviewed the chapter on bootstrapping, and Sandeep Kumar, Steve Lodin, and Christoph Schuba, from the COAST security project at Purdue, commented on the security chapter. Special thanks go to my wife, Chris, whose careful editing made many improvements in wording.

FOR FURTHER STUDY

The protocols described in this chapter are all specified in Internet RFCs. Postel [RFC 821] describes the Simple Mail Transfer Protocol and gives many examples. The exact format of mail messages is given by Crocker [RFC 822]. Borenstein and Freed [RFC 1521] specifies the standard for MIME, including the syntax of header declarations, the interpretation of content types, and the *base64* encoding mentioned in this chapter. Moore [RFC 1522] defines MIME header extensions for non-ASCII text, and Postel [RFC 1590] describes the procedure for registration of new content and encoding types. Partridge [RFC 974] discusses the relationship between mail routing and the domain name system. Horton [RFC 976] proposes a standard for the UNIX UUCP mail system.

EXERCISES

25.1 Some mail systems force the user to specify a sequence of machines through which the message should travel to reach its destination. The mail protocol in each machine merely passes the message on to the next machine. List three disadvantages of such a scheme.

25.2 Find out if your computing system allows you to invoke SMTP directly.

25.3 Build an SMTP client and use it to deliver a mail message.

25.4 See if you can send mail through a mail gateway and back to yourself.

25.5 Make a list of mail address forms that your site handles and write a set of rules for parsing them.

25.6 Find out how the UNIX *sendmail* program can be used to implement a mail gateway.

25.7 Find out how often your local mail system attempts delivery and how long it will continue before giving up.

25.8 Many mail systems allow users to direct incoming mail to a program instead of storing it in a mailbox. Build a program that accepts your incoming mail, places your mail in a file, and then sends a reply to tell the sender you are on vacation.

25.9 Read the SMTP standard carefully. Then use TELNET to connect to the SMTP port on a remote machine and ask the remote SMTP server to expand a mail alias.

25.10 A user receives mail in which the *To* field specifies the string *important-people*. The mail was sent from a computer on which the alias *important-people* includes no valid mailbox identifiers. Read the SMTP specification carefully to see how such a situation is possible.

25.11 Read the MIME standard carefully. What servers can be specified in a MIME external reference?

26

Applications: Internet Management (SNMP, SNMPv2)

26.1 Introduction

In addition to protocols that provide network level services and application programs that use those services, an internet needs software that allows managers to debug problems, control routing, and find computers that violate protocol standards. We refer to such activities as *internet management*. This chapter considers the ideas behind TCP/IP internet management software, and describes an internet management protocol.

26.2 The Level Of Management Protocols

Originally, many wide area networks included management protocols as part of their link level protocols. If a packet switch began misbehaving, the network manager could instruct a neighboring packet switch to send it a special *control packet*. Control packets caused the receiver to suspend normal operation and respond to commands from the manager. The manager could interrogate the packet switch to identify problems, examine or change routes, test one of the communication interfaces, or reboot the switch. Once managers repaired the problem, they could instruct the switch to resume normal operations. Because management tools were part of the lowest level protocol, managers were often able to control switches even if higher level protocols failed.

Unlike a homogeneous wide area network, a TCP/IP internet does not have a single link level protocol. Instead, the internet consists of multiple physical networks interconnected by IP routers. As a result, internet management differs from network management. First, a single manager can control heterogeneous routers†. Second, the controlled entities may not share a common link level protocol. Third, the set of machines a manager controls may lie at arbitrary points in an internet. In particular, a manager may need to control one or more machines that do not attach to the same physical network as the manager's computer. Thus, it may not be possible for a manager to communicate with machines being controlled unless the management software uses protocols that provide end-to-end connectivity across an internet. As a consequence, the internet management protocol used with TCP/IP operates above the transport level:

> *In a TCP/IP internet, IP routers form the active switches that managers need to examine and control. Because routers attach to heterogeneous networks, protocols for internet management operate at the application level and communicate using TCP/IP transport-level protocols.*

Designing internet management software to operate at the application level has several advantages. Because the protocols can be designed without regard to the underlying network hardware, one set of protocols can be used for all networks. Because the protocols can be designed without regard to the hardware on the managed machine, the same protocols can be used for all managed devices. From a manager's point of view, having a single set of management protocols means uniformity – all routers respond to exactly the same set of commands. Furthermore, because the management software uses IP for communication, a manager can control the routers across an entire TCP/IP internet without having direct attachment to every physical network or router.

Of course, building management software at the application level also has disadvantages. Unless the operating system, IP software, and transport protocol software work correctly, the manager may not be able to contact the router. For example, if the router's routing table becomes damaged, it may be impossible to correct the table or reboot the machine from a remote site. If the operating system on a router crashes, it will be impossible to reach the application program that implements the internet management protocols even if the router can still field hardware interrupts and route packets.

26.3 Architectural Model

Despite the potential disadvantages, having TCP/IP management software operate at the application level has worked well in practice. The most significant advantage of placing network management protocols at a high level becomes apparent when one considers a large internet, where a manager's computer does not need to attach directly to all physical networks that contain managed entities. Figure 26.1 shows an example of the architecture.

†Although managers can control both routers and hosts, we will focus on control of routers because they present the most complexity.

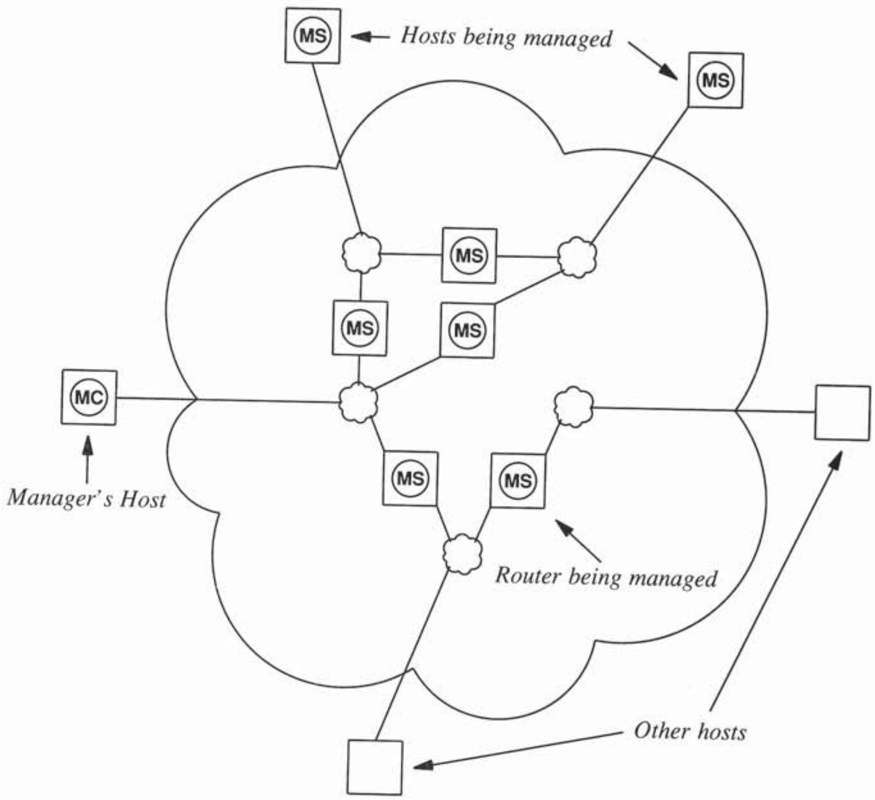

Figure 26.1 Example of network management. A manager invokes management client (MC) software that contacts management server (MS) software on routers throughout the internet.

As the figure shows, each participating host or router runs a server program. Technically, the server is called a *management agent*. A manager invokes client software on the local host computer and specifies an agent with which it communicates. After the client contacts the agent, it sends queries to obtain information or it sends commands to change conditions in the router. Of course, not all routers in a large internet fall under a single manager. Most managers only control a few routers at their local sites.

Internet management software uses an authentication mechanism to ensure only authorized managers can access or control a particular router. Some management protocols support multiple levels of authorization, allowing a manager specific privileges on each router. For example, a specific router could be configured to allow several managers to obtain information while only allowing a select subset of them to change information or control the router.

26.4 Protocol Architecture

TCP/IP network† management protocols divide the management problem into two parts and specify separate standards for each part. The first part concerns communication of information. A protocol specifies how client software running on a manager's host communicates with an agent. The protocol defines the format and meaning of messages clients and servers exchange as well as the form of names and addresses. The second part concerns the data being managed. A protocol specifies which data items a router must keep as well as the name of each data item and the syntax used to express the name.

26.4.1 A Standard Network Management Protocol

The current standard TCP/IP network management protocol is the *Simple Network Management Protocol* (*SNMP*). A second version has been approved, but is not widely in use at the time this is being written. Known as *SNMPv2*, the new version adds new capabilities, including stronger security.

26.4.2 A Standard For Managed Information

A router being managed must keep control and status information that the manager can access. For example, a router keeps statistics on the status of its network interfaces, incoming and outgoing traffic, dropped datagrams, and error messages generated. Although it allows a manager to access these statistics, SNMP does not specify exactly which data can be accessed. Instead, a separate standard specifies the details. Known as a *Management Information Base* (*MIB*), the standard specifies the data items a host or router must keep and the operations allowed on each. For example, the MIB specifies that IP software must keep a count of all octets that arrive over each network interface, and it specifies that network management software can only read those values.

The MIB for TCP/IP divides management information into eight categories as Figure 26.2 shows. The choice of categories is important because identifiers used to specify items include a code for the category.

†Technically, there is a distinction between internet management protocols and network management protocols. Historically, however, TCP/IP internet management protocols are known as *network management* protocols; we will follow the accepted terminology.

MIB category	Includes Information About
system	The host or router operating system
interfaces	Individual network interfaces
addr. trans.	Address translation (e.g., ARP mappings)
ip	Internet Protocol software
icmp	Internet Control Message Protocol software
tcp	Transmission Control Protocol software
udp	User Datagram Protocol software
egp	Exterior Gateway Protocol software

Figure 26.2 Categories of information in the MIB. The category is encoded in the identifier used to specify an object.

Keeping the MIB definition independent of the network management protocol has advantages for both vendors and users. A vendor can include SNMP agent software in a product such as a router, with the guarantee that the software will continue to adhere to the standard after new MIB items are defined. A customer can use the same network management client software to manage multiple routers that have different versions of a MIB. Of course, a router that does not have new MIB items cannot provide the information in those items. However, because all routers use the same language for communication, they can all parse a query and either provide the requested information or send an error message explaining that they do not have the requested item.

26.5 Examples of MIB Variables

In addition to the standard TCP/IP MIB, which is known as *MIB-II*, many RFCs document MIB variables for specific devices. Examining a few of the data items the standard MIB includes will help clarify the contents. Figure 26.3 lists example MIB variables along with their categories.

MIB Variable	Category	Meaning
sysUpTime	system	Time since last reboot
ifNumber	interfaces	Number of network interfaces
ifMtu	interfaces	MTU for a particular interface
ipDefaultTTL	ip	Value IP uses in time-to-live field
ipInReceives	ip	Number of datagrams received
ipForwDatagrams	ip	Number of datagrams forwarded
ipOutNoRoutes	ip	Number of routing failures
ipReasmOKs	ip	Number of datagrams reassembled
ipFragOKs	ip	Number of datagrams fragmented
ipRoutingTable	ip	IP Routing table
icmpInEchos	icmp	Number of ICMP Echo Requests received
tcpRtoMin	tcp	Minimum retransmission time TCP allows
tcpMaxConn	tcp	Maximum TCP connections allowed
tcpInSegs	tcp	Number of segments TCP has received
udpInDatagrams	udp	Number of UDP datagrams received
egpInMsgs	egp	Number of EGP messages received

Figure 26.3 Examples of MIB variables along with their categories.

Values for most of the items listed in Figure 26.3 can be stored in a single integer. However, the MIB also defines more complex structures. For example, the MIB variable *ipRoutingTable* refers to a router's routing table. Additional MIB variables define the contents of a routing table entry, and allow the network management protocols to reference the data for individual entries. Of course, MIB variables present only a logical definition of each data item – the internal data structures a router uses may differ from the MIB definition. When a query arrives, software in the agent on the router is responsible for mapping between the MIB variable and the data structure the router uses to store the information.

26.6 The Structure Of Management Information

In addition to the MIB standard, which specifies network management variables and their meanings, a separate standard specifies a set of rules used to define and identify MIB variables. The rules are known as the *Structure of Management Information* (*SMI*) specification. To keep network management protocols simple, the SMI places restrictions on the types of variables allowed in the MIB, specifies the rules for naming those variables, and creates rules for defining variable types. For example, the SMI standard includes definitions of terms like *IpAddress* (defining it to be a 4-octet string) and *Counter* (defining it to be an integer in the range of 0 to $2^{32}-1$), and specifies that they are the terms used to define MIB variables. More important, the rules in the SMI describe how the MIB refers to tables of values (e.g., the IP routing table).

26.7 Formal Definitions Using ASN.1

The SMI standard specifies that all MIB variables must be defined and referenced using ISO's *Abstract Syntax Notation 1* (*ASN.1*†). ASN.1 is a formal language that has two main features: a notation used in documents that humans read, and a compact encoded representation of the same information used in communication protocols. In both cases, the precise, formal notation removes any possible ambiguities from both the representation and meaning. For example, instead of saying that a variable contains an integer value, a protocol designer who uses ASN.1 must state the exact form and range of numeric values. Such precision is especially important when implementations include heterogeneous computers that do not all use the same representations for data items.

Besides keeping standards documents unambiguous, ASN.1 also helps simplify the implementation of network management protocols and guarantees interoperability. It defines precisely how to encode both names and data items in a message. Thus, once the documentation of a MIB has been expressed using ASN.1, the human readable form can be translated directly and mechanically into the encoded form used in messages. In summary:

> *The TCP/IP network management protocols use a formal notation called ASN.1 to define names and types for variables in the management information base. The precise notation makes the form and contents of variables unambiguous.*

26.8 Structure And Representation Of MIB Object Names

We said that ASN.1 specifies how to represent both data items and names. However, understanding the names used for MIB variables requires us to know about the underlying namespace. Names used for MIB variables are taken from the *object identifier* namespace administered by ISO and ITU. The key idea behind the object identifier namespace is that it provides a namespace in which all possible objects can be named. The namespace is not restricted to variables used in network management – it includes names for arbitrary objects (e.g., each international protocol standard document has a name).

The object identifier namespace is *absolute* (*global*), meaning that names are structured to make them globally unique. Like most namespaces that are large and absolute, the object identifier namespace is hierarchical. Authority for parts of the namespace is subdivided at each level, allowing individual groups to obtain authority to assign some of the names without consulting a central authority for each assignment‡.

The root of the object identifier hierarchy is unnamed, but has three direct descendants managed by: ISO, ITU, and jointly by ISO and ITU. The descendants are assigned both short text strings and integers to identify them (the text strings are used

†ASN.1 is usually pronounced by reading the dot: 'A-S-N dot 1'.

‡Readers should recall from the Domain Name System discussion in Chapter 22 how authority for a hierarchical namespace is subdivided.

when humans need to understand object names; computer software uses the integers to form compact, encoded representations of the names). ISO has allocated one subtree for use by other national or international standards organizations (including U.S. standards organizations), and the U.S. National Institute for Standards and Technology† has allocated a subtree for the U.S. Department of Defense. Finally, the IAB has petitioned the Department of Defense to allocate it a subtree in the namespace.

Figure 26.4 illustrates pertinent parts of the object identifier hierarchy and shows the position of the node used by TCP/IP network management protocols.

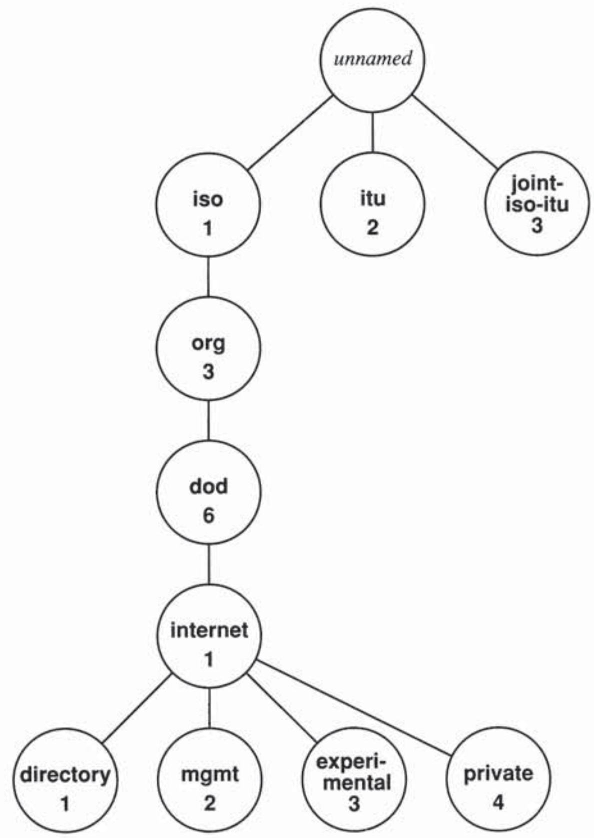

Figure 26.4 Part of the hierarchical object identifier namespace used to name MIB variables. An object's name consists of the numeric labels along a path from the root to the object.

†NIST was formerly the National Bureau of Standards.

The name of an object in the hierarchy is the sequence of numeric labels on the nodes along a path from the root to the object. The sequence is written with periods separating the individual components. For example, the name *1.3.6.1.1* denotes the node labeled *directory*. The MIB has been assigned a node under the *internet mgmt* subtree with label *mib* and numeric value *1*. Because all MIB variables fall under that node, they all have names beginning with the prefix *1.3.6.1.2.1*.

Earlier we said that the MIB groups all variables into eight categories. The exact meaning of the categories can now be explained: they are the eight subtrees of the *mib* node of the object identifier namespace. Figure 26.5 illustrates the idea by showing part of the naming subtree under the *mib* node.

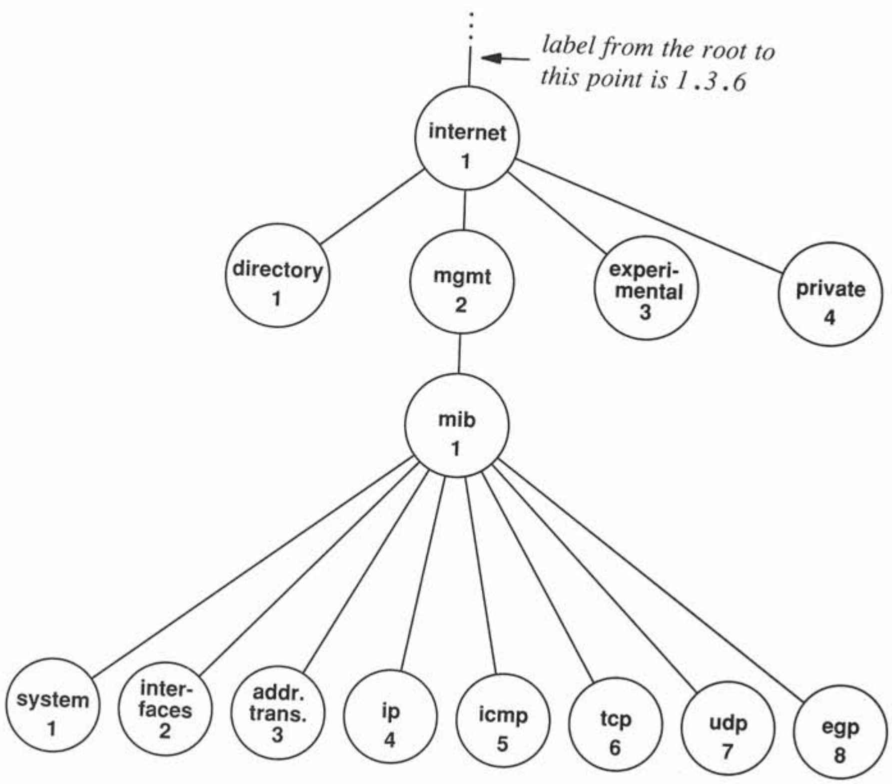

Figure 26.5 The object identifier namespace under the IAB *mib* node. Each subtree corresponds to one of the eight categories of MIB variables.

Two examples will make the naming syntax clear. Figure 26.5 shows that the category labeled *ip* has been assigned the numeric value *4*. Thus, the names of all MIB variables corresponding to IP have an identifier that begins with the prefix *1.3.6.1.2.1.4.* If one wanted to write out the textual labels instead of the numeric representation, the name would be:

iso.org.dod.internet.mgmt.mib.ip

A MIB variable named *ipInReceives* has been assigned numeric identifier *3* under the *ip* node in the namespace, so its name is:

iso.org.dod.internet.mgmt.mib.ip.ipInReceives

and the corresponding numeric representation is:

1.3.6.1.2.1.4.3

When network management protocols use names of MIB variables in messages, each name has a suffix appended. For simple variables, the suffix *0* refers to the instance of the variable with that name. So, when it appears in a message sent to a router, the numeric representation of *ipInReceives* is:

1.3.6.1.2.1.4.3.0

which refers to the instance of *ipInReceives* on that router. Note that there is no way to guess the numeric value or suffix assigned to a variable. One must consult the published standards to find which numeric values have been assigned to each object type. Thus, programs that provide mappings between the textual form and underlying numeric values do so entirely by consulting tables of equivalences – there is no closed-form computation that performs the transformation.

As a second, more complex example, consider the MIB variable *ipAddrTable*, which contains a list of the IP addresses for each network interface. The variable exists in the namespace as a subtree under *ip*, and has been assigned the numeric value *20*. Therefore, a reference to it has the prefix:

iso.org.dod.internet.mgmt.mib.ip.ipAddrTable

with a numeric equivalent:

1.3.6.1.2.1.4.20

In programming language terms, we think of the IP address table as a one-dimensional array, where each element of the array consists of a structure (record) that contains five items: an IP address, the integer index of an interface corresponding to the entry, an IP subnet mask, an IP broadcast address, and an integer that specifies the maximum datagram size that the router will reassemble. Of course, not all routers have such an array in memory. The router may keep this information in many variables or may need to follow pointers to find it. However, the MIB provides a name for the array as if it existed, and allows network management software on individual routers to map table references into appropriate internal variables.

Using ASN.1 style notation, we can define *ipAddrTable*:

```
ipAddrTable ::= SEQUENCE OF IpAddrEntry
```

where *SEQUENCE* and *OF* are keywords that define an ipAddrTable to be a one-dimensional array of *IpAddrEntry*s. Each entry in the array is defined to consist of five fields (the definition assumes that *IpAddress* has already been defined).

```
IpAddrEntry ::= SEQUENCE {
    ipAdEntAddr
            IpAddress,
    ipAdEntIfIndex
            INTEGER,
    ipAdEntNetMask
            IpAddress,
    ipAdEntBcastAddr
            IpAddress,
    ipAdEntReasmMaxSize
            INTEGER (0..65535)
}
```

Further definitions must be given to assign numeric values to *ipAddrEntry* and to each item in the *IpAddrEntry* sequence. For example, the definition:

```
ipAddrEntry { ipAddrTable 1 }
```

specifies that *ipAddrEntry* falls under *ipAddrTable* and has numeric value *1*. Similarly, the definition:

```
ipAdEntNetMask { ipAddrEntry 3 }
```

assigns *ipAdEntNetMask* numeric value *3* under *ipAddrEntry*.

We said that *ipAddrTable* was like a one-dimensional array. However, there is a significant difference in the way programmers use arrays and the way network management software uses tables in the MIB. Programmers think of an array as a set of elements that have an index used to select a specific element. For example, the programmer might write *xyz[3]* to select the third element from array *xyz*. ASN.1 syntax does not use integer indices. Instead, MIB tables append a suffix onto the name to select a specific element in the table. For our example of an IP address table, the standard specifies that the suffix used to select an item consists of an IP address. Syntactically, the IP address (in dotted decimal notation) is concatenated onto the end of the object name to form the reference. Thus, to specify the network mask field in the IP address table entry corresponding to address `128.10.2.3`, one uses the name:

iso.org.dod.internet.mgmt.mib.ip.ipAddrTable.ipAddrEntry.ipAdEntNetMask.128.10.2.3

which, in numeric form, becomes:

1.3.6.1.2.1.4.20.1.3.128.10.2.3

Although concatenating an index to the end of a name may seem awkward, it provides a powerful tool that allows clients to search tables without knowing the number of items or the type of data used as an index. The next section shows how network management protocols use this feature to step through a table one element at a time.

26.9 Simple Network Management Protocol

Network management protocols specify communication between the network management client program a manager invokes and a network management server program executing on a host or router. In addition to defining the form and meaning of messages exchanged and the representation of names and values in those messages, network management protocols also define administrative relationships among routers being managed. That is, they provide for authentication of managers.

One might expect network management protocols to contain a large number of commands. Some early protocols, for example, supported commands that allowed the manager to: *reboot* the system, *add* or *delete* routes, *disable* or *enable* a particular network interface, or *remove cached address bindings*. The main disadvantage of building management protocols around commands arises from the resulting complexity. The protocol requires a separate command for each operation on a data item. For example, the command to delete a routing table entry differs from the command to disable an interface. As a result, the protocol must change to accommodate new data items.

SNMP takes an interesting alternative approach to network management. Instead of defining a large set of commands, SNMP casts all operations in a *fetch-store paradigm*†. Conceptually, SNMP contains only two commands that allow a manager to fetch a value from a data item or store a value into a data item. All other operations are defined as side-effects of these two operations. For example, although SNMP does not have an explicit *reboot* operation, an equivalent operation can be defined by declaring a data item that gives the time until the next reboot and allowing the manager to assign the item a value (including zero).

The chief advantages of using a fetch-store paradigm are stability, simplicity, and flexibility. SNMP is especially stable because its definition remains fixed, even though new data items are added to the MIB and new operations are defined as side-effects of storing into those items. SNMP is simple to implement, understand, and debug because

†The fetch-store paradigm comes from a management protocol system known as HEMS. See Partridge and Trewitt [RFCs 1021, 1022, 1023, and 1024] for details.

it avoids the complexity of having special cases for each command. Finally, SNMP is especially flexible because it can accommodate arbitrary commands in an elegant framework.

From a manager's point of view, of course, SNMP remains hidden. The user interface to network management software can phrase operations as imperative commands (e.g., *reboot*). Thus, there is little visible difference between the way a manager uses SNMP and other network management protocols. In fact, vendors have begun to sell network management software that offers a graphical user interface. Such software displays diagrams of network connectivity, and uses a point-and-click style of interaction.

As Figure 26.6 shows, SNMP offers more than the two operations we have described.

Command	Meaning
get-request	Fetch a value from a specific variable
get-next-request	Fetch a value without knowing its exact name
get-response	Reply to a fetch operation
set-request	Store a value in a specific variable
trap	Reply triggered by an event

Figure 26.6 The set of possible SNMP operations†. *Get-next-request* allows the manager to iterate through a table of items.

Operations *get-request*, *get-response*, and *set-request* provide the basic fetch and store operations (as well as replies to those operations). SNMP specifies that operations must be *atomic*, meaning that if a single SNMP message specifies operations on multiple variables, the server either performs all operations or none of them. In particular, no assignments will be made if any of them are in error. The *trap* operation allows managers to program servers to send information when an event occurs. For example, an SNMP server can be programmed to send a manager a *trap* message whenever one of the attached networks becomes unusable (i.e., an interface goes down).

26.9.1 Searching Tables Using Names

We said that ASN.1 does not provide mechanisms for declaring arrays or indexing them in the usual sense. However, it is possible to denote individual elements of a table by appending a suffix to the object identifier for the table. Unfortunately, a client program may wish to examine entries in a table for which it does not know all valid suffixes. The *get-next-request* operation allows a client to iterate through a table without knowing how many items the table contains. The rules are quite simple. When sending a *get-next-request*, the client supplies a prefix of a valid object identifier, *P*. The server examines the set of object identifiers for all variables it controls, and responds by sending a *get-response* command for the one that has an object identifier lexicographically

†SNMPv2 adds a *get-bulk* operation that permits a manager to fetch multiple values with a single request.

greater than *P*. Because the MIB uses suffixes to index tables, a client can send the prefix of an object identifier corresponding to a table and receive the first element in the table. The client can send the name of the first element in a table and receive the second, and so on.

Consider an example search. Recall that the *ipAddrTable* uses IP addresses to identify entries in the table. A client that does not know which IP addresses are in the table on a given router cannot form a complete object identifier. However, the client can still use the *get-next-request* operation to search the table by sending the prefix:

iso . org . dod . internet . mgmt . mib . ip . ipAddrTable . ipAddrEntry . ipAdEntNetMask

which, in numeric form, is:

1 . 3 . 6 . 1 . 2 . 1 . 4 . 20 . 1 . 3

The server returns the network mask field of the first entry in *ipAddrTable*. The client uses the full object identifier returned by the server to request the next item in the table.

26.10 SNMP Message Format

Unlike most TCP/IP protocols, SNMP messages do not have fixed fields. Instead, they use the standard ASN.1 encoding. Thus, they can be difficult for humans to decode and understand. After examining the SNMP message definition in ASN.1 notation, we will review the ASN.1 encoding scheme briefly, and see an example of an encoded SNMP message.

An SNMP message consists of three main parts: a protocol *version*, an SNMP *community* identifier (used to group together the routers managed by a given manager), and a *data* area. The data area is divided into *protocol data units* (*PDUs*). Each PDU consists of a request (sent by client) or a response (sent by server). Figure 26.7 shows how the message can be described in ASN.1 notation.

```
SNMP-Message ::=
    SEQUENCE {
        version INTEGER {
            version-1 (0)
        },
        community
            OCTET STRING,
        data
            ANY
    }
```

Figure 26.7 The SNMP message format in ASN.1 notation. The *data* area contains one or more protocol data units.

The five types of protocol data units are further described in ASN.1 notation in Figure 26.8.

```
SNMP-PDUs ::=
    CHOICE {
        get-request
            GetRequest-PDU,
        get-next-request-PDU
            GetNextRequest-PDU,
        get-response
            GetResponse-PDU,
        set-request
            SetRequest-PDU,
        trap
            Trap-PDU,
    }
```

Figure 26.8 The ASN.1 definitions of an SNMP PDU. The syntax for each request type must be specified further.

The definition specifies that each protocol data unit consists of one of the five request or response types. To complete the definition of an SNMP message, we must further specify the syntax of the five individual types. For example, Figure 26.9 shows the definition of a *get-request*.

```
GetRequest-PDU ::= [0]
    IMPLICIT SEQUENCE {
        request-id
            RequestID,
        error-status
            ErrorStatus,
        error-index
            ErrorIndex,
        variable-bindings
            VarBindList
    }
```

Figure 26.9 The ASN.1 definition of a *get-request* message. Formally, the message is defined to be a *GetRequest-PDU*.

Further definitions in the standard specify the remaining undefined terms. *RequestID* is defined to be a 4-octet integer (used to match responses to queries). Both *ErrorStatus* and *ErrorIndex* are single octet integers which contain the value zero in a request. Finally, *VarBindList* contains a list of object identifiers for which the client seeks values. In ASN.1 terms, the definitions specify that *VarBindList* is a sequence of pairs of object name and value. ASN.1 represents the pairs as a sequence of two items. Thus, in the simplest possible request, *VarBindList* is a sequence of two items: a name and a *null*.

26.11 Example Encoded SNMP Message

The encoded form of ASN.1 uses variable-length fields to represent items. In general, each field begins with a header that specifies the type of object and its length in bytes. For example, Figure 26.10 shows the string of encoded octets in a *get-request* message for data item *sysDescr* (numeric object identifier *1.3.6.1.2.1.1.1*).

```
   30       29       02       01       00
SEQUENCE len=41  INTEGER  len=1    vers=0

   04       06       70       75       62       6C       69       63
string   len=6    p        u        b        l        i        c

   A0       1C       02       04       05       AE       56       02
getreq.  len=28  INTEGER  len=4    ------- request ID -------

   02       01       00       02       01       00
INTEGER  len=1   status  INTEGER  len=1 error index

   30       0E       30       0C       06       08
SEQUENCE len=14 SEQUENCE len=12 objectid len=8

   2B       06       01       02       01       01       01       00
  1.3   .   6    .   1    .   2    .   1    .   1    .   1    .   0

   05       00
  null   len=0
```

Figure 26.10 The encoded form of a *get-request* for data item *sysDescr* with octets shown in hexadecimal and their meanings below. Related octets have been grouped onto lines; they are contiguous in the message.

As Figure 26.10 shows, the message starts with a code for *SEQUENCE* which has a length of 41 octets. The first item in the sequence is a 1-octet integer that specifies the protocol *version*. The *community* field is stored in a character string, which in the example, is a 6-octet string that contains the word *public*.

The *GetRequest-PDU* occupies the remainder of the message. The initial code specifies a *get-Request* operation. Because the high-order bit is turned on, the interpretation is *context specific*. That is, the hexadecimal value *A0* only specifies a *GetRequest-PDU* when used in an SNMP message; it is not a universally reserved value. Following the request octet, the length octet specifies the request is *28* octets long. The request ID is *4* octets, but each of the error status and error index are one octet. Finally, the sequence of pairs contains one binding, a single object identifier bound to a *null* value. The identifier is encoded as expected except that the first two numeric labels are combined into a single octet.

26.12 Summary

Network management protocols allow a manager to monitor and control routers and hosts. A network management client program executing on the manager's workstation contacts one or more servers, called agents, running on the computers to be controlled. Because an internet consists of heterogeneous machines and networks, TCP/IP management software executes as application programs and uses internet transport protocols (e.g., UDP) for communication between clients and servers.

The standard TCP/IP network management protocol is SNMP, the Simple Network Management Protocol. SNMP defines a low-level management protocol that provides two basic operations: fetch a value from a variable or store a value into a variable. In SNMP, all operations occur as side-effects of storing values into variables. SNMP defines the format of messages that travel between a manager's computer and a managed entity.

A companion standard to SNMP defines the set of variables that a managed entity maintains. The standard is known as a Management Information Base, or MIB. MIB variables are described using ASN.1, a formal language that provides a concise encoded form as well as a precise human-readable notation for names and objects. ASN.1 uses a hierarchical namespace to guarantee that all MIB names are globally unique while still allowing subgroups to assign parts of the namespace.

FOR FURTHER STUDY

Schoffstall, Fedor, Davin, and Case [RFC 1157] contains the standard for SNMP. ISO [May 87a] and [May 87b] contain the standard for ASN.1 and specify the encoding. McCloghrie and Rose [RFC 1213] defines the variables that comprise MIB-II, while McCloghrie and Rose [RFC 1211] contains the SMI rules for naming MIB variables.

A series of RFCs defines SNMPv2, which is a proposed standard at the time of this writing. Case, McCloghrie, Rose, Waldbusser [RFC 1441] contains an introduction to SNMPv2. Case, McCloghrie, Rose, and Waldbusser [RFC 1450] defines the

SNMPv2 MIB. Galvin and McCloghrie [RFC 1446] discusses SNMPv2 security protocols. Case, McCloghrie, Rose, Waldbusser [RFC 1448] specifies protocol operations.

An older proposal for a network management protocol called HEMS can be found in Trewitt and Partridge [RFCs 1021, 1022, 1023, and 1024]. Davin, Case, Fedor, and Schoffstall [RFC 1028] specifies a predecessor to SNMP known as the Simple Gateway Monitoring Protocol (SGMP).

EXERCISES

26.1 Capture an SNMP packet with a network analyzer and decode the fields.

26.2 Read the standard to find out how ASN.1 encodes the first two numeric values from an object identifier in a single octet. Why does it do so?

26.3 Read the specification for CMIP. How many commands does it support?

26.4 Suppose the MIB designers needed to define a variable that corresponded to a two-dimensional array. How can ASN.1 notation accommodate references to such a variable?

26.5 What are the advantages and disadvantages of defining globally unique ASN.1 names for MIB variables?

26.6 If you have SNMP client code available, try using it to read MIB variables in a local router. What is the advantage of allowing arbitrary managers to read variables in all routers?

26.7 Read the MIB specification to find the definition of variable *ipRoutingTable* that corresponds to an IP routing table. Design a program that will use SNMP to contact multiple routers, and see if any entries in their routing tables cause a routing loop. Exactly what ASN.1 names should such a program generate?

Bibliography

ABRAMSON, N. [1970], The ALOHA System – Another Alternative for Computer Communications, *Proceedings of the Fall Joint Computer Conference.*

ABRAMSON, N. and F. KUO (EDS.) [1973], *Computer Communication Networks,* Prentice Hall, Englewood Cliffs, New Jersey.

ANDREWS, D. W., and G. D. SHULTZ [1982], A Token-Ring Architecture for Local Area Networks: An Update, *Proceedings of Fall 82 COMPCON,* IEEE.

BALL, J. E., E. J. BURKE, I. GERTNER, K. A. LANTZ, and R. F. RASHID [1979], Perspectives on Message-Based Distributed Computing, *IEEE Computing Networking Symposium,* 46-51.

BBN [1981], A History of the ARPANET: The First Decade, *Technical Report* Bolt, Beranek, and Newman, Inc.

BBN [December 1981], Specification for the Interconnection of a Host and an IMP (revised), *Technical Report 1822*, Bolt, Beranek, and Newman, Inc.

BIAGIONI E., E. COOPER, and R. SANSOM [March 1993], Designing a Practical ATM LAN, *IEEE Network*, 32-39.

BERTSEKAS D. and R. GALLAGER [1987], *Data Networks*, Prentice-Hall, Englewood Cliffs, New Jersey.

BIRRELL, A., and B. NELSON [February 1984], Implementing Remote Procedure Calls, *ACM Transactions on Computer Systems,* 2(1), 39-59.

BOGGS, D., J. SHOCH, E. TAFT, and R. METCALFE [April 1980], Pup: An Internetwork Architecture, *IEEE Transactions on Communications.*

BORMAN, D., [April 1989], Implementing TCP/IP on a Cray Computer, *Computer Communication Review,* 19(2), 11-15.

BROWN, M., K. KOLLING, and E. TAFT [November 1985], The Alpine File System, *ACM Transactions on Computer Systems,* 3(4), 261-293.

BROWNBRIDGE, D., L. MARSHALL, and B. RANDELL [December 1982], The Newcastle Connections or UNIXes of the World Unite!, *Software – Practice and Experience*, 12(12), 1147-1162.

CASNER, S., and S. DEERING [July 1992], First IETF Internet Audiocast, *Computer Communications Review*, 22(3), 92-97.

CERF, V., and E. CAIN [October 1983], The DOD Internet Architecture Model, *Computer Networks.*

CERF, V., and R. KAHN [May 1974], A Protocol for Packet Network Interconnection, *IEEE Transactions of Communications,* Com-22(5).

CERF, V. [October 1989], A History of the ARPANET, *ConneXions, The Interoperability Report,* 480 San Antonio Rd, Suite 100, Mountain View, California.

CHERITON, D. R. [1983], Local Networking and Internetworking in the V-System, *Proceedings of the Eighth Data Communications Symposium.*

CHERITON, D. R. [April 1984], The V Kernel: A Software Base for Distributed Systems, *IEEE Software,* 1(2), 19-42.

CHERITON, D. [August 1986], VMTP: A Transport Protocol for the Next Generation of Communication Systems, *Proceedings of ACM SIGCOMM '86*, 406-415.

CHERITON, D., and T. MANN [May 1984], Uniform Access to Distributed Name Interpretation in the V-System, *Proceedings IEEE Fourth International Conference on Distributed Computing Systems,* 290-297.

CHESSON, G. [June 1987], Protocol Engine Design, *Proceedings of the 1987 Summer USENIX Conference,* Phoenix, AZ.

CHESWICK, W., and S. BELLOVIN [1994], *Firewalls And Internet Security: Repelling the Wiley Hacker,* Addison-Wesley, Reading, Massachusetts.

CLARK, D. [December 1985], The structure of Systems Using Upcalls, *Proceedings of the Tenth ACM Symposium on Operating Systems Principles,* 171-180.

CLARK, D., M. LAMBERT, and L. ZHANG [August 1987], NETBLT: A High Throughput Transport Protocol, *Proceedings of ACM SIGCOMM '87.*

CLARK, D., V. JACOBSON, J. ROMKEY, and H. SALWEN [June 1989], An Analysis of TCP Processing Overhead, *IEEE Communications*, 23-29.

COHEN, D., [1981], On Holy Wars and a Plea for Peace, *IEEE Computer,* 48-54.

COMER, D. E. and J. T. KORB [1983], CSNET Protocol Software: The IP-to-X25 Interface, *Computer Communications Review,* 13(2).

COMER, D. E. [1984], *Operating System Design – The XINU Approach,* Prentice-Hall, Englewood Cliffs, New Jersey.

COMER, D. E. [1987], *Operating System Design Vol II. – Internetworking With XINU,* Prentice-Hall, Englewood Cliffs, New Jersey.

COMER, D. E. and D. L. STEVENS [1994] *Internetworking With TCP/IP Volume II – Design, Implementation, and Internals,* 2nd edition, Prentice-Hall, Englewood Cliffs, New Jersey.

COMER, D. E. and D. L. STEVENS [1993] *Internetworking With TCP/IP Volume III – Client-Server Programming And Applications, BSD socket version,* Prentice-Hall, Englewood Cliffs, New Jersey.

COMER, D. E. and D. L. STEVENS [1994] *Internetworking With TCP/IP Volume III – Client-Server Programming And Applications, AT&T TLI version,* Prentice-Hall, Englewood Cliffs, New Jersey.

COMER, D. E., T. NARTEN, and R. YAVATKAR [April 1987], The Cypress Network: A Low-Cost Internet Connection Technology, *Technical Report TR-653,* Purdue University, West Lafayette, IN.

COMER, D. E., T. NARTEN, and R. YAVATKAR [1987], The Cypress Coaxial Packet Switch, *Computer Networks and ISDN Systems*, vol. 14:2-5, 383-388.

COTTON, I. [1979], Technologies for Local Area Computer Networks, *Proceedings of the Local Area Communications Network Symposium.*

CROWLEY, T., H, FORSDICK, M. LANDAU, and V. TRAVERS [June 1987], The Diamond Multimedia Editor, *Proceedings of the 1987 Summer USENIX Conference,* Phoenix, AZ.

DALAL Y. K., and R. S. PRINTIS [1981], 48-Bit Absolute Internet and Ethernet Host Numbers, *Proceedings of the Seventh Data Communications Symposium.*

DEERING S. E., and D. R. CHERITON [May 1990], Multicast Routing in Datagram Internetworks and Extended LANs, *ACM Transactions on Computer Systems,* 8(2), 85-110.

DEERING, S., D. ESTRIN, D. FARINACCI, V. JACOBSON, C-G LIU, and L. WEI [August 1994], An Architecture for Wide-Area Multicasting Routing, *Proceedings of ACM SIGCOMM '94*, 126-135.

DENNING P. J., [September-October 1989], *The Science of Computing: Worldnet*, in American Scientist, 432-434.

DENNING P. J., [November-December 1989], *The Science of Computing: The ARPANET After Twenty Years*, in American Scientist, 530-534.

DE PRYCKER, M. [1993] *Asynchronous Transfer Mode Solution for Broadband ISDN,* 2nd edition, Ellis Horwood, UK.

DIGITAL EQUIPMENT CORPORATION., INTEL CORPORATION, and XEROX CORPORATION [September 1980], *The Ethernet: A Local Area Network Data Link Layer and Physical Layer Specification.*

DION, J. [Oct. 1980], The Cambridge File Server, *Operating Systems Review,* 14(4), 26-35.

DRIVER, H., H. HOPEWELL, and J. IAQUINTO [September 1979], How the Gateway Regulates Information Control, *Data Communications.*

EDGE, S. W. [1979], Comparison of the Hop-by-Hop and Endpoint Approaches to Network Interconnection, in *Flow Control in Computer Networks,* J-L. GRANGE and M. GIEN (EDS.), North-Holland, Amsterdam, 359-373.

EDGE, S. [1983], An Adaptive Timeout Algorithm for Retransmission Across a Packet Switching Network, *Proceedings of ACM SIGCOMM '83.*

ENSLOW, P. [January 1978], What is a 'Distributed' Data Processing System? *Computer,* 13-21.

ERIKSSON, H. [August 1994] MBONE: The Multicast Backbone, *Communications of the ACM,* 37(8), 54-60.

FALK, G. [1983], The Structure and Function of Network Protocols, in *Computer Communications, Volume I: Principles,* CHOU, W. (ED.), Prentice-Hall, Englewood Cliffs, New Jersey.

FARMER, W. D., and E. E. NEWHALL [1969], An Experimental Distributed Switching System to Handle Bursty Computer Traffic, *Proceedings of the ACM Symposium on Probabilistic Optimization of Data Communication Systems,* 1-33.

FCCSET [November 1987], A Research and Development Strategy for High Performance Computing, *Report from the Executive Office of the President and Office of Science and Technology Policy.*

FEDOR, M. [June 1988], GATED: A Multi-Routing Protocol Daemon for UNIX, *Proceedings of the 1988 Summer USENIX conference*, San Francisco, California.

FEINLER, J., O. J. JACOBSEN, and M. STAHL [December 1985], *DDN Protocol Handbook Volume Two, DARPA Internet Protocols,* DDN Network Information Center, SRI International, 333 Ravenswood Avenue, Room EJ291, Menlo Park, California.

FLOYD, S. and V. JACOBSON [August 1993], Random Early Detection Gateways for Congestion Avoidance, *IEEE/ACM Transactions on Networking*, 1(4).

FRANK, H., and W. CHOU [1971], Routing in Computer Networks, *Networks,* 1(1), 99-112.

FRANK, H., and J. FRISCH [1971], *Communication, Transmission, and Transportation Networks,* Addison-Wesley, Reading, Massachusetts.

FRANTA, W. R., and I. CHLAMTAC [1981], *Local Networks,* Lexington Books, Lexington, Massachusetts.

FRICC [May 1989], *Program Plan for the National Research and Education Network*, Federal Research Internet Coordinating Committee, US Department of Energy, Office of Scientific Computing report ER-7.

FRIDRICH, M., and W. OLDER [December 1981], The Felix File Server, *Proceedings of the Eighth Symposium on Operating Systems Principles,* 37-46.

FULTZ, G. L., and L. KLEINROCK, [June 14-16, 1971], Adaptive Routing Techniques for Store-and-Forward Computer Communication Networks, presented at *IEEE International Conference on Communications,* Montreal, Canada.

GERLA, M., and L. KLEINROCK [April 1980], Flow Control: A Comparative Survey, *IEEE Transactions on Communications.*

GOSIP [April 1989], U.S. Government Open Systems Interconnection Profile (GOSIP) version 2.0, GOSIP Advanced Requirements Group, National Institute of Standards and Technology (NIST).

GRANGE, J-L., and M. GIEN (EDS.) [1979], *Flow Control in Computer Networks,* North-Holland, Amsterdam.

GREEN, P. E. (ED.) [1982], *Computer Network Architectures and Protocols,* Plenum Press, New York.

HINDEN, R., J. HAVERTY, and A. SHELTZER [September 1983], The DARPA Internet: Interconnecting Heterogeneous Computer Networks with Gateways, *Computer.*

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION [June 1986a], Information processing systems — Open Systems Interconnection — *Transport Service Definition,* International Standard number 8072, ISO, Switzerland.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION [July 1986b], Information processing systems — Open Systems Interconnection — *Connection Oriented Transport Protocol Specification,* International Standard number 8073, ISO, Switzerland.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION [May 1987a], Information processing systems — Open Systems Interconnection — *Specification of Basic Specification of Abstract Syntax Notation One (ASN.1),* International Standard number 8824, ISO, Switzerland.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION [May 1987b], Information processing systems — Open Systems Interconnection — *Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1),* International Standard number 8825, ISO, Switzerland.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION [May 1988a], Information processing systems — Open Systems Interconnection — *Management Information Service Definition, Part 2: Common Management Information Service,* Draft International Standard number 9595-2, ISO, Switzerland.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION [May 1988a], Information processing systems — Open Systems Interconnection — *Management Information Protocol Definition, Part 2: Common Management Information Protocol,* Draft International Standard number 9596-2.

JACOBSEN, O. J. (PUBLISHER) [1987-], ConneXions, The Interoperability Report, *Interop Company*, a division of Softbank Exposition and Conference Company, Foster City, California.

JACOBSON, V. [August 1988], Congestion Avoidance and Control, *Proceedings ACM SIGCOMM '88.*

JAIN, R. [January 1985], On Caching Out-of-Order Packets in Window Flow Controlled Networks, *Technical Report*, DEC-TR-342, Digital Equipment Corporation.

JAIN, R. [March 1986], Divergence of Timeout Algorithms for Packet Retransmissions, *Proceedings Fifth Annual International Phoenix Conference on Computers and Communications*, Scottsdale, AZ.

JAIN, R. [October 1986], A Timeout-Based Congestion Control Scheme for Window Flow-Controlled Networks, *IEEE Journal on Selected Areas in Communications*, Vol. SAC-4, no. 7.

JAIN, R. [May 1992], Myths About Congestion Management in High-speed Networks, *Internetworking: Research and Experience*, 3(3), 101-113.

JENNINGS, D. M., L. H. LANDWEBER, and I. H. FUCHS [February 28, 1986], Computer Networking for Scientists and Engineers, *Science* vol 231, 941-950.

JUBIN, J. and J. TORNOW [January 1987], The DARPA Packet Radio Network Protocols, *IEEE Proceedings.*

KAHN, R. [November 1972], Resource-Sharing Computer Communications Networks, *Proceedings of the IEEE,* 60(11), 1397-1407.

KARN, P., H. PRICE, and R. DIERSING [May 1985], Packet Radio in the Amateur Service, *IEEE Journal on Selected Areas in Communications,*

KARN, P., and C. PARTRIDGE [August 1987], Improving Round-Trip Time Estimates in Reliable Transport Protocols, *Proceedings of ACM SIGCOMM '87.*

KENT, C., and J. MOGUL [August 1987], Fragmentation Considered Harmful, *Proceedings of ACM SIGCOMM '87.*

KLINE, C. [August 1987], Supercomputers on the Internet: A Case Study, *Proceedings of ACM SIGCOMM '87*.

KOCHAN, S. G., and P. H. WOODS [1989], *UNIX Networking,* Hayden Books, Indianapolis, IN.

LABARRE, L. (ED.) [December 1989], OSI Internet Management: Management Information Base, *Internet Draft <IETF.DRAFTS>DRAFT-IETF-SNMP-MIB2-01.TXT,* DDN Network Information Center, SRI International, Ravenswood, CA.

LAMPSON, B. W., M. PAUL, and H. J. SIEGERT (EDS.) [1981], *Distributed Systems - Architecture and Implementation (An Advanced Course),* Springer-Verlag, Berlin.

LANZILLO, A. L., and C. PARTRIDGE [January 1989], Implementation of Dial-up IP for UNIX Systems, *Proceedings 1989 Winter USENIX Technical Conference,* San Diego, CA.

LAQUEY, T. L., [July 1989], *User's Directory of Computer Networks*, Digital Press, Bedford, MA.

LAZAR, A. [November 1983], Optimal Flow Control of a Class of Queuing Networks in Equilibrium. *IEEE Transactions on Automatic Control*, Vol. AC-28:11.

LEFFLER, S., M. McKUSICK, M. KARELS, and J. QUARTERMAN [1989], *The Design and Implementation of the 4.3BSD UNIX Operating System,* Addison-Wesley, Reading, Massachusetts.

LYNCH, D. C., (FOUNDER) [1987-], The NETWORLD+INTEROP Conference, *Interop Company,* a division of Softbank Exposition and Conference Company, Foster City, California.

MCNAMARA, J. [1982], *Technical Aspects of Data Communications,* Digital Press, Digital Equipment Corporation, Bedford, Massachusetts.

MCQUILLAN, J. M., I. RICHER, and E. ROSEN [May 1980], The New Routing Algorithm for the ARPANET, *IEEE Transactions on Communications,* (COM-28), 711-719.

MERIT [November 1987], Management and Operation of the NSFNET Backbone Network: A Proposal Funded by the National Science Foundation and the State of Michigan, *MERIT Incorporated,* Ann Arbor, Michigan.

METCALFE, R. M., and D. R. BOGGS [July 1976], Ethernet: Distributed Packet Switching for Local Computer Networks, *Communications of the ACM*, 19(7), 395-404.

MILLER, C. K., and D. M. THOMPSON [March 1982], Making a Case for Token Passing in Local Networks, *Data Communications.*

MILLS, D., and H-W. BRAUN [August 1987], The NSFNET Backbone Network, *Proceedings of ACM SIGCOMM '87*.

MITCHELL, J., and J. DION [April 1982], A Comparison of Two Network-Based File Servers, *Communications of the ACM*, 25(4), 233-245.

MORRIS, R. [1979], Fixing Timeout Intervals for Lost Packet Detection in Computer Communication Networks, *Proceedings AFIPS National Computer Conference*, AFIPS Press, Montvale, New Jersey.

NAGLE, J. [April 1987], On Packet Switches With Infinite Storage, *IEEE Transactions on Communications*, Vol. COM-35:4.

NARTEN, T. [Sept. 1989], Internet Routing, *Proceedings ACM SIGCOMM '89.*

NEEDHAM, R. M. [1979], System Aspects of the Cambridge Ring, *Proceedings of the ACM Seventh Symposium on Operating System Principles,* 82-85.

NELSON, J. [September 1983], 802: A Progress Report, *Datamation.*

OPPEN, D., and Y. DALAL [October 1981], The Clearinghouse: A Decentralized Agent for Locating Named Objects, Office Products Division, XEROX Corporation.

PARTRIDGE, C. [June 1986], Mail Routing Using Domain Names: An Informal Tour, *Proceedings of the 1986 Summer USENIX Conference*, Atlanta, GA.

PARTRIDGE, C. [June 1987], Implementing the Reliable Data Protocol (RDP), *Proceedings of the 1987 Summer USENIX Conference*, Phoenix, Arizona.

PARTRIDGE, C. [1994], *Gigabit Networking*, Addison-Wesley, Reading, Massachusetts.

PETERSON, L. [1985], *Defining and Naming the Fundamental Objects in a Distributed Message System,* Ph.D. Dissertation, Purdue University, West Lafayette, Indiana.

PIERCE, J. R. [1972], Networks for Block Switching of Data, *Bell System Technical Journal,* 51.

POSTEL, J. B. [April 1980], Internetwork Protocol Approaches, *IEEE Transactions on Communications,* COM-28, 604-611.

POSTEL, J. B., C. A. SUNSHINE, and D. CHEN [1981], The ARPA Internet Protocol, *Computer Networks.*

QUARTERMAN, J. S. [1990], *The Matrix: Computer Networks and Conferencing Systems Worldwide*, Digital Press, Digital Equipment Corporation, Maynard, MA.

QUARTERMAN, J. S., and J. C. HOSKINS [October 1986], Notable Computer Networks, *Communications of the ACM*, 29(10).

RAMAKRISHNAN, K. and R. JAIN [May 1990], A Binary Feedback Scheme For Congestion Avoidance In Computer Networks, *ACM Transactions on Computer Systems*, 8(2), 158-181.

REYNOLDS, J., J. POSTEL, A. R. KATZ, G. G. FINN, and A. L. DESCHON [October 1985], The DARPA Experimental Multimedia Mail System, *IEEE Computer*.

RITCHIE, D. M., and K. THOMPSON [July 1974], The UNIX Time-Sharing System, *Communications of the ACM,* 17(7), 365-375; revised and reprinted in *Bell System Technical Journal,* 57(6), [July-August 1978], 1905-1929.

ROSE, M. (ED.) [October 1989], Management Information Base for Network Management of TCP/IP-based Internets, *Internet Draft <IETF.DRAFTS>DRAFT-IETF-OIM-MIB2-00.TXT,* DDN Network Information Center, SRI International, Ravenswood, CA.

ROSENTHAL, R. (ED.) [November 1982], *The Selection of Local Area Computer Networks,* National Bureau of Standards Special Publication 500-96.

SALTZER, J. [1978], Naming and Binding of Objects, *Operating Systems, An Advanced Course,* Springer-Verlag, 99-208.

SALTZER, J. [April 1982], Naming and Binding of Network Destinations, *International Symposium on Local Computer Networks,* IFIP/T.C.6, 311-317.

SALTZER, J., D. REED, and D. CLARK [November 1984], End-to-End Arguments in System Design, *ACM Transactions on Computer Systems,* 2(4), 277-288.

SCHWARTZ, M., and T. STERN [April 1980], *IEEE Transactions on Communications,* COM-28(4), 539-552.

SHOCH, J. F. [1978], Internetwork Naming, Addressing, and Routing, *Proceedings of COMPCON.*

SHOCH, J. F., Y. DALAL, and D. REDELL [August 1982], Evolution of the Ethernet Local Computer Network, *Computer.*

SNA [1975], *IBM System Network Architecture – General Information,* IBM System Development Division, Publications Center, Department E01, P.O. Box 12195, Research Triangle Park, North Carolina, 27709.

SOLOMON, M., L. LANDWEBER, and D. NEUHEGEN [1982], The CSNET Name Server, *Computer Networks* (6), 161-172.

STALLINGS, W. [1984], *Local Networks: An Introduction,* Macmillan Publishing Company, New York.

STALLINGS, W. [1985], *Data and Computer Communications,* Macmillan Publishing Company, New York.

SWINEHART, D., G. MCDANIEL, and D. R. BOGGS [December 1979], WFS: A Simple Shared File System for a Distributed Environment, *Proceedings of the Seventh Symposium on Operating System Principles,* 9-17.

TANENBAUM, A. [1981], *Computer Networks: Toward Distributed Processing Systems,* Prentice-Hall, Englewood Cliffs, New Jersey.

TICHY, W., and Z. RUAN [June 1984], Towards a Distributed File System, *Proceedings of Summer 84 USENIX Conference,* Salt Lake City, Utah, 87-97.

TOMLINSON. R. S. [1975], Selecting Sequence Numbers, *Proceedings ACM SIGOPS/SIGCOMM Interprocess Communication Workshop*, 11-23, 1975.

WARD, A. A. [1980], TRIX: A Network-Oriented Operating System, *Proceedings of COMPCON,* 344-349.

WATSON, R. [1981], Timer-Based Mechanisms in Reliable Transport Protocol Connection Management, *Computer Networks*, North-Holland Publishing Company.

WEINBERGER, P. J. [1985], The UNIX Eighth Edition Network File System, *Proceedings 1985 ACM Computer Science Conference,* 299-301.

WELCH, B., and J. OSTERHAUT [May 1986], Prefix Tables: A Simple Mechanism for Locating Files in a Distributed System, *Proceedings IEEE Sixth International Conference on Distributed Computing Systems,* 1845-189.

WILKES, M. V., and D. J. WHEELER [May 1979], The Cambridge Digital Communication Ring, *Proceedings Local Area Computer Network Symposium.*

XEROX [1981], Internet Transport Protocols, *Report XSIS 028112,* Xerox Corporation, Office Products Division, Network Systems Administration Office, 3333 Coyote Hill Road, Palo Alto, California.

ZHANG, L. [August 1986], Why TCP Timers Don't Work Well, *Proceedings of ACM SIGCOMM '86.*

Index

C

D

E

F

G

M

N

O

P

R

S

V

W

X

Z

The Series Of Internetworking Books from Douglas Comer and Prentice Hall

Internetworking With TCP/IP Volume I: Principles, Protocols, and Architecture 3rd ed., 1995, ISBN 0-13-216987-8

The classic reference in the field for anyone who wants to understand Internet technology, Volume I surveys the TCP/IP and describes each component. The highly accessible text presents the scientific principles used in the construction of TCP/IP, and shows how the components were designed to work together. It covers details of each protocol, including ARP, RARP, IP, TCP, UDP, RIP, OSPF and others.

Internetworking With TCP/IP Volume II: Design, Implementation, and Internals (with D. Stevens) 2nd ed: 1994, ISBN 0-13-125527-4

Ideal for implementors, Volume II continues the discussion of Volume I by using code from a running implementation of TCP/IP to illustrate all the details. The text shows, for example, how TCP's slow start algorithm interacts with the Partridge-Karn exponential retransmission backoff, and how routing updates interact with datagram forwarding.

Internetworking With TCP/IP Volume III: Client-Server Programming and Applications (with D. Stevens)

BSD Socket Version: 1992, ISBN 0-13-474222-2
AT&T TLI Version: 1993, ISBN 0-13-474230-3

Volume III describes the fundamental concept of client-server computing used to build all distributed computing systems. The text discusses various server designs as well as the tools and techniques used to build clients and servers, including Remote Procedure Call (RPC). It contains examples of running programs that illustrate each of the designs and tools. Two versions of volume III are available for the widely used BSD socket and AT&T TLI interfaces.

The Internet Book: Everything you need to know about computer networking and how the Internet works

Paperback: 1995, ISBN 0-13-151565-9
Study Guide: 1995, ISBN 0-13-188012-8
Book plus Study Guide: 1995, ISBN 0-13-400029-3

A gentle introduction to networking and the Internet, *The Internet Book* does not assume the reader has a technical background. It explains the Internet, how it works, and services available in general terms, without focusing on a particular computer or a particular brand of software. Ideal for someone who wants to become Internet and computer networking literate, *The Internet Book* explains the terminology as well as the concepts; an extensive glossary of terms and abbreviations is included. A separate *Study Guide* provides suggestions for readers, review questions, and exercises).