To Chris

About The Author

Dr. Douglas Comer is an internationally recognized expert on TCP/IP protocols and the Internet. One of the researchers who contributed to the Internet as it was being formed in the late 1970s and 1980s, he was a member of the Internet Architecture Board, the group responsible for guiding the Internet's development. He was also chairman of the CSNET technical committee and a member of the CSNET executive committee.

Comer consults for companies on the design and implementation of networks, and gives professional seminars on TCP/IP and internetworking to both technical and nontechnical audiences around the world. His operating system, Xinu, and implementation of TCP/IP protocols are documented in his books, and used in commercial products.

Comer is a professor of computer science at Purdue University, where he teaches courses and does research on computer networking, internetworking, and operating systems. In addition to writing a series of best-selling technical books, he serves as the North American editor of the journal *Software — Practice and Experience.* Comer is a Fellow of the ACM.

Additional information can be found at:

www.cs.purdue.edu/people/comer

Contents

Foreword

This is the fourth edition of a landmark book, the book that signaled the coming of age of the Internet. Development of the protocols for the Internet started around 1974, and they had been in limited but real use starting in the early 80's, but as of 1987, there was still no good introduction to how they worked or how to code them. The standards documents for TCP, IP and the other protocols existed, of course, but the true truth — the collection of knowledge and wisdom necessary to implement a protocol stack and actually expect it to work — that was a mystery, known only to a small band of the initiated. That was not a good thing, and the initiated knew it. But it takes a lot of effort to pull all the right stuff together and write it down. We waited, knowing that a good book explaining TCP/IP would be an important step towards the broad acceptance of our protocols.

And Doug wrote the book.

We told jokes, waiting for the book. We looked to see how many books there were in mature fields, and speculated that the number of books was a metric of success. I actually went and looked to see how many books there were on "how to build a compiler" (a post-mature field by now, perhaps — time to count the books again). The compiler community was well off, and even "how to build a database" was available. But nothing on "how to build a TCP/IP." And then we got our book.

Of course, knowing that back then this was a landmark book is not enough to make you buy it. Collectors might want to find the first edition, but that gives the true truth as of 12 years ago, a long time in Internet years. And that is why this is the fourth edition. A lot has changed over that time. We have learned a lot more, the field has grown up, whole new protocols have emerged, and Doug has rewritten the book three times. That is a measure both of how much and how fast the field changes, and how much work must go into keeping this book current. It has all the new stuff, and our best current knowledge about all the old stuff.

Other things have changed in 12 years. Not only has the Internet grown up, but some of our heroes have grown old, and some have died. The foreword to the first edition was written by Jon Postel, one of the true Internet pioneers, who died in the fall of 1998. Below, we have reprinted the foreword he wrote for the first edition. Much is the same, but much has changed. This is still a very readable book both for details on TCP/IP and for an introduction to communications protocols in general. But in 1987, Jon wrote "Computer communication systems and networks are currently separated and

fragmented. The goal of interconnection and internetworking, to have a single powerful computer communication network, is fundamental to the design of TCP/IP.'' Only 12 years ago networks were fragmented; today the Internet unites the world. And TCP/IP is still the glue, at the core of the Internet, that makes all this work. And this is still the book to read to learn about it.

David Clark
Massachusetts Institute of Technology

December, 1999

Foreword To The First Edition By The Late Jon Postel

In this book Professor Douglas Comer has provided a long sought overview and introduction to TCP/IP. There have been many requests for "the" article, report, or book to read to get started on understanding the TCP/IP protocols. At last, this book satisfies those requests. 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 (See Chapters 3, 10, 17, and 18)†. Computer communication protocols do not do anything themselves. Like operating systems, they are in the service of applications 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 (See Chapters 11, 12, 17, 18, and 19).

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 of the

†Editor's note: chapter numbers have changed since the first edition.

telephones are in (as it appears to the users) one network. Computer communication systems and networks are currently separated and fragmented. 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 (See Chapters 4, 5, and 6), and a universal protocol — the Internet Protocol (See Chapters 7, 8, and 9).

To have an internetwork the individual networks must be connected. The connecting devices are called gateways. Further, these gateways 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 gateway 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 gateways are called routing algorithms, and these are currently the subject of much study and development (See Chapters 13, 14, 15, and 16).

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 circa 1987. And, as Doug Comer points out, there are many loose ends (See Chapter 20).

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 (See Appendix 3).

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,
Internet Protocol Designer and
Deputy Internet Architect

December, 1987

Preface

The explosive growth of the Internet continues. When the third edition of this book was written five years ago, the Internet connected 4.8 million computers, up from 5,000 when the first edition was published. The Internet now reaches over 56 million computers, meaning that the 1995 Internet was only about 8% of its current size. During the early 1990s, those of us who were involved with the Internet marveled at how large an obscure research project had become. Now, it pervades almost every aspect of society.

TCP/IP has accommodated change well. The basic technology has survived nearly two decades 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 in the original design. 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.

This edition contains updated information throughout the text as well as new material that describes technical advances and changes. For example, because classless addressing has become widely deployed, the description of IP forwarding examines techniques for classless lookup. In addition, the chapters on IP describe the Differentiated Services (DiffServe) scheme for classes of service as well as path MTU discovery and anonymous networks. The chapter on TCP describes Random Early Drop (RED). The chapter on exterior routing has been updated to use BGP as the primary example. The descriptions of protocols such as RIP, IGMP, SNMP, and IPv6 have been revised to incorporate new versions and recent changes. Finally, the chapter on security discusses IPsec.

Four new chapters contain detailed information about significant developments. Chapter 19 describes mobile IP — a technology that allows a computer to move from one network to another without changing its IP address. Chapter 20 considers two technologies used to interconnect private intranets and the global Internet: Virtual Private Network (VPN) and Network Address Translation (NAT). Each solves a slightly different problem; both are widely deployed. Chapter 28 covers the HTML and HTTP protocols that form the basis for the most significant Internet application: the world wide web. Chapter 29 focuses on an exciting new area: sending real-time data such as

voice and video over an IP network. The chapter examines the RTP protocol that allows a receiver to coordinate and play such data as well as the RSVP and COPS protocols that can be used to provide resource reservation, and describes the H.323 suite of protocols used for IP telephony.

The fourth edition retains the same general contents and overall organization as the third 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 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 im-

pressive (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 14-20 and 32 describe the architecture of an internet when viewed globally. They explore routing architecture and the protocols routers use to exchange routing information. Finally, Chapters 21-31 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 clearly, and discusses motivations and consequences.

I thank all the people who have contributed to versions of this book. Michael Evangelista provided extensive assistance with this edition, including classifying RFCs. Jeff Case provided the SNMPv3 example. John Lin and Dennis Totin commented on some of the new chapters. Jin Zhang, Kechiun He, and Sara Steinbrueck proofread parts of the text. Special thanks go to my wife and partner, Chris, whose careful editing made many improvements throughout.

Douglas E. Comer

January, 2000

What Others Have Said About The Fourth Edition Of Internetworking With TCP/IP

"This is the book I go to for clear explanantions of the basic principles and latest developments in TCP/IP technologies. It's a 'must have' reference for networking professionals."

Dr. Ralph Droms
Professor at Bucknell University

"When the Nobel committee turns its attention to the Internet, Doug gets the prize for literature. This is an updated classic that is the best way to master Internet technology."

Dr. Paul V. Mockapetris
Inventor of the Domain Name System

"The best-written TCP/IP book I have ever read. Dr. Comer explains complex ideas clearly, with excellent diagrams and explanations."

Dr. John Lin,
Bell Laboratories

"Comer continues to prove himself the Baedeker of the Internet Protocols with this fine 4th edition."

Dr. Vinton Cerf
Senior Vice president, MCI WorldCom

"There are many TCP/IP books on the shelves today, but Doug Comer's 'Internetworking with TCP/IP' is the one that comes *off* the shelf for accessible and authoritative answers to questions about Internet technology."

Dr. Lyman Chapin,
Chief Scientist, BBN Technologies

Other Books In the Internetworking Series from Douglas Comer and Prentice Hall

Internetworking With TCP/IP Volume II: Design, Implementation, and Internals (with David Stevens), 3rd edition: 1999, ISBN 0-13-973843-6

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 algorithm and how routing updates interact with datagram forwarding.

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

BSD Socket Version, 2nd edition: 1996, ISBN 0-13-260969-X
AT&T TLI Version: 1994, ISBN 0-13-474230-3
Windows Sockets Version: 1997, ISBN 0-13-848714-6

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. Three versions of Volume III are available for the socket API (Unix), the TLI API (AT&T System V), and the Windows Sockets API (Microsoft).

Computer Networks And Internets (with a CD-ROM by Ralph Droms), 2nd edition: 1999, ISBN 0-13-083617-6

A broad introduction to data communication, networking, internetworking, and client-server applications, *Computer Networks And Internets* examines the hardware and software components that make up computer networks, from the lowest levels through applications. The text covers transmission and modems, LANs and LAN extensions, access technologies, WANs, protocols (including TCP/IP), and network applications. The CD-ROM features animations and data sets.

The Internet Book: Everything you need to know about computer networking and how the Internet works, 2nd edition: 1997, ISBN 0-13-890161-9, paperback

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.

To order, visit the Prentice Hall Web page at www.prenhall.com/ or contact your local bookstore or Prentice Hall representative. In North America, call 1-515-284-6751, or send a FAX to 1-515-284-6719.

1

Introduction And Overview

1.1 The Motivation For Internetworking

Internet communication has become a fundamental part of life. The World Wide Web contains information about such diverse subjects as atmospheric conditions, crop production, stock prices, and airline traffic. Groups establish electronic mailing lists so they can share information of common interest. Professional colleagues exchange business correspondence electronically, and relatives exchange personal greetings.

Unfortunately, most network technologies are designed for a specific purpose. Each enterprise chooses hardware technology appropriate for specific communication needs and budget. More important, it is impossible to engineer a universal network from a single network technology because no single network suffices for all uses. Some groups need high-speed networks to connect computers in a single building. Low-cost technologies that fill the need cannot span large geographic distances. Other groups settle for a slower speed network that connects machines thousands of miles apart.

For over two decades, a new technology has evolved that makes it possible to interconnect many disparate physical networks and make them function as a coordinated unit. The technology, called *internetworking*, accommodates multiple, diverse underlying hardware technologies by providing a way to interconnect heterogeneous networks and a set of communication conventions that makes them interoperate. The internet technology hides the details of network hardware, and permits computers to communicate independent of their physical network connections.

The internet technology described in this book is an example of *open system interconnection*. It is called *open* because, unlike proprietary communication systems available from one specific vendor, the specifications are publicly available. Thus, anyone can build the software needed to communicate across an internet. More important, the entire technology has been designed to foster communication among machines with

diverse hardware architectures, to use almost any packet switched network hardware, to accommodate a wide variety of applications, and to accommodate multiple computer operating systems.

To appreciate internet technology, think of how it has changed business. In addition to high-speed communication among employees in the office environment, networking technologies provide instant feedback among the production side of the business, sales and marketing, and customers. As a result, the speed with which business can plan, implement, assess, and retool has increased; the change is dramatic.

1.2 The TCP/IP Internet

U.S. government agencies realized the importance and potential of internet technology many years ago, and have funded research that has made possible a global Internet. This book discusses principles and ideas underlying the internet technology that has resulted from research funded by the *Advanced Research Projects Agency* (*ARPA*)†. The ARPA technology includes a set of network standards that specify the details of how computers communicate, as well as a set of conventions for interconnecting networks and routing traffic. Officially named the TCP/IP Internet Protocol Suite and commonly referred to as *TCP/IP* (after the names of its two main standards), it can be used to communicate across any set of interconnected networks. For example, some corporations use TCP/IP to interconnect all networks within their corporation, even though the corporation has no connection to outside networks. Other groups use TCP/IP for communication among geographically distant sites.

Although the TCP/IP technology is noteworthy by itself, it is especially interesting because its viability has been demonstrated on a large scale. It forms the base technology for the global Internet that connects over 170 million individuals in homes, schools, corporations, and government labs in virtually all populated countries. In the U.S., The *National Science Foundation* (*NSF*), the *Department of Energy* (*DOE*), the *Department of Defense* (*DOD*), the *Health and Human Services Agency* (*HHS*), and the *National Aeronautics and Space Administration* (*NASA*) have all participated in funding the Internet, and use TCP/IP to connect many of their research sites. Known as the *ARPA/NSF Internet*, the *TCP/IP Internet*, the *global Internet*, or just the *Internet*‡, the resulting communication system allows subscribers to share information with anyone around the world as easily as they share it with someone in the next room. An outstanding success, the Internet demonstrates the viability of the TCP/IP technology and shows how it can accommodate a wide variety of underlying network technologies.

Most of the material in this book applies to any internet that uses TCP/IP, but some chapters refer specifically to the global Internet. Readers interested only in the technology should be careful to watch for the distinction between the Internet architecture as it exists and general TCP/IP internets as they might exist. It would be a mistake, however, to ignore all sections of the text that describe the global Internet — many corporate networks are already more complex than the global Internet of a dozen

†At various times, *ARPA* was called the *Defense Advanced Research Projects Agency* (*DARPA*).

‡We will follow the usual convention of capitalizing *Internet* when referring specifically to the global Internet, and use lower case to refer to private internets that use TCP/IP technology.

years ago, and many of the problems they face have already been solved in the global Internet.

1.3 Internet Services

One cannot appreciate the technical details underlying TCP/IP without understanding the services it provides. This section reviews internet services briefly, highlighting the services most users access, and leaves to later chapters the discussion of how computers connect to a TCP/IP internet and how the functionality is implemented.

Much of our discussion of services will focus on standards called *protocols*. Protocols like TCP and IP provide the syntactic and semantic rules for communication. They contain the details of message formats, describe how a computer responds when a message arrives, and specify how a computer handles errors or other abnormal conditions. Most important, they allow us to discuss computer communication independent of any particular vendor's network hardware. In a sense, protocols are to communication what algorithms are to computation. An algorithm allows one to specify or understand a computation without knowing the details of a particular CPU instruction set. Similarly, a communication protocol allows one to specify or understand data communication without depending on detailed knowledge of a particular vendor's network hardware.

Hiding the low-level details of communication helps improve productivity in several ways. First, because programmers deal with higher-level protocol abstractions, they do not need to learn or remember as many details about a given hardware configuration. Thus, they can create new programs quickly. Second, because programs built using higher-level abstractions are not restricted to a particular computer architecture or a particular network hardware, they do not need to be changed when computers or networks are replaced or reconfigured. Third, because application programs built using higher-level protocols are independent of the underlying hardware, they can provide direct communication between an arbitrary pair of computers. Programmers do not need to build a special version of application software for each type of computer or each type of network. Instead, software built to use protocols is general-purpose; the same code can be compiled and run on an arbitrary computer.

We will see that the details of each service available on the Internet are given by a separate protocol. The next sections refer to protocols that specify some of the application-level services as well as those used to define network-level services. Later chapters explain each of these protocols in detail.

1.3.1 Application Level Internet Services

From the user's point of view, the Internet appears to consist of a set of application programs that use the underlying network to carry out useful communication tasks. We use the term *interoperability* to refer to the ability of diverse computing systems to cooperate in solving computational problems. Internet application programs exhibit a high degree of interoperability. Most users that access the Internet do so merely by run-

ning application programs without understanding the types of computers being accessed, the TCP/IP technology, the structure of the underlying internet, or even the path the data travels to its destination; they rely on the application programs and the underlying network software to handle such details. Only programmers who write network application programs need to view a TCP/IP internet as a network and need to understand some of the technology.

The most popular and widespread Internet application services include:

- *World Wide Web.* The Web allows users to view documents that contain text and graphics, and to follow hypermedia links from one document to another. The Web grew to become the largest source of traffic on the global Internet between 1994 and 1995, and continues to dominate. Some service providers estimate that the Web now accounts for 80% of their Internet traffic.
- *Electronic mail (e-mail).* Electronic mail allows a user to compose a memo and send a copy to individuals or groups. Another part of the mail application allows users to read memos that they have received. A recent innovation allows users to include "attachments" with a mail message that consist of arbitrary files. Electronic mail has been so successful that many Internet users depend on it for most correspondence. One reason for the popularity of Internet e-mail arises from a careful design: the protocol makes delivery reliable. Not only does the mail system on the sender's computer contact the mail system on the receiver's computer directly, but the protocol specifies that a message cannot be deleted by the sender until the receiver has successfully placed a copy on permanent storage.
- *File transfer.* The file transfer application allows users to send or receive a copy of a data file. File transfer is one of the oldest, and still among the most heavily used application services in the Internet. Although small files can now be attached to an e-mail message, the file transfer service is still needed to handle arbitrarily large files. The system provides a way to check for authorized users, or even to prevent all access. Like mail, file transfer across a TCP/IP internet is reliable because the two machines involved communicate directly, without relying on intermediate machines to make copies of the file along the way.
- *Remote login.* Remote login allows a user sitting at one computer to connect to a remote machine and establish an interactive login session. The remote login makes it appear that a window on the user's screen connects directly to the remote machine by sending each keystroke from the user's keyboard to the remote machine and displaying each character the remote computer prints in the user's window. When the remote login session terminates, the application returns the user to the local system.

We will return to these and other applications in later chapters to examine them in more detail. We will see exactly how they use the underlying TCP/IP protocols, and why having standards for application protocols has helped ensure that they are widespread.

1.3.2 Network-Level Internet Services

A programmer who creates application programs that use TCP/IP protocols has an entirely different view of an internet than a user who merely executes applications like electronic mail. At the network level, an internet provides two broad types of service that all application programs use. While it is unimportant at this time to understand the details of these services, they cannot be omitted from any overview of TCP/IP:

- *Connectionless Packet Delivery Service.* This service, explained in detail throughout the text, forms the basis for all other internet services. Connectionless delivery is an abstraction of the service that most packet-switching networks offer. It means simply that a TCP/IP internet routes small messages from one computer to another based on address information carried in the message. Because the connectionless service routes each packet separately, it does not guarantee reliable, in-order delivery. Because it usually maps directly onto the underlying hardware, the connectionless service is extremely efficient. More important, having connectionless packet delivery as the basis for all internet services makes the TCP/IP protocols adaptable to a wide range of network hardware.

- *Reliable Stream Transport Service.* Most applications need much more than packet delivery because they require the communication software to recover automatically from transmission errors, lost packets, or failures of intermediate switches along the path between sender and receiver. The reliable transport service handles such problems. It allows an application on one computer to establish a "connection" with an application on another computer, and then to send a large volume of data across the connection as if it were a permanent, direct hardware connection. Underneath, of course, the communication protocols divide the stream of data into small messages and send them, one at a time, waiting for the receiver to acknowledge reception.

Many networks provide basic services similar to those outlined above, so one might wonder what distinguishes TCP/IP services from others. The primary distinguishing features are:

- *Network Technology Independence.* Although TCP/IP is based on conventional packet switching technology, it is independent of any particular vendor's hardware. The global Internet includes a variety of network technologies ranging from networks designed to operate within a single building to those designed to span large distances. TCP/IP protocols define the unit of data transmission, called a *datagram*, and specify how to transmit datagrams on a particular network.

- *Universal Interconnection.* A TCP/IP internet allows any pair of computers to which it attaches to communicate. Each computer is assigned an *address* that is universally recognized throughout the internet. Every datagram carries the addresses of its source and destination. Intermediate switching computers use the destination address to make routing decisions.

- *End-to-End Acknowledgements.* The TCP/IP internet protocols provide acknowledgements between the original source and ultimate destination instead of between successive machines along the path, even if the source and destination do not connect to a common physical network.
- *Application Protocol Standards.* In addition to the basic transport-level services (like reliable stream connections), the TCP/IP protocols include standards for many common applications including electronic mail, file transfer, and remote login. Thus, when designing application programs that use TCP/IP, programmers often find that existing software provides the communication services they need.

Later chapters will discuss the details of the services provided to the programmer as well as many of the application protocol standards.

1.4 History And Scope Of The Internet

Part of what makes the TCP/IP technology so exciting is its universal adoption as well as the size and growth rate of the global Internet. ARPA began working toward an internet technology in the mid 1970s, with the architecture and protocols taking their current form around 1977-79. At that time, ARPA was known as the primary funding agency for packet-switched network research and had pioneered many ideas in packet-switching with its well-known *ARPANET*. The ARPANET used conventional point-to-point leased line interconnection, but ARPA had also funded exploration of packet-switching over radio networks and satellite communication channels. Indeed, the growing diversity of network hardware technologies helped force ARPA to study network interconnection, and pushed internetworking forward.

The availability of research funding from ARPA caught the attention and imagination of several research groups, especially those researchers who had previous experience using packet switching on the ARPANET. ARPA scheduled informal meetings of researchers to share ideas and discuss results of experiments. Informally, the group was known as the *Internet Research Group*. By 1979, so many researchers were involved in the TCP/IP effort that ARPA created an informal committee to coordinate and guide the design of the protocols and architecture of the emerging Internet. Called the Internet Control and Configuration Board (*ICCB*), the group met regularly until 1983, when it was reorganized.

The global Internet began around 1980 when ARPA started converting machines attached to its research networks to the new TCP/IP protocols. The ARPANET, already in place, quickly became the backbone of the new Internet and was used for many of the early experiments with TCP/IP. The transition to Internet technology became complete in January 1983 when the Office of the Secretary of Defense mandated that all computers connected to long-haul networks use TCP/IP. At the same time, the *Defense Communication Agency* (*DCA*) split the ARPANET into two separate networks, one for further research and one for military communication. The research part retained the name ARPANET; the military part, which was somewhat larger, became known as the *military network*, *MILNET*.

To encourage university researchers to adopt and use the new protocols, ARPA made an implementation available at low cost. At that time, most university computer science departments were running a version of the UNIX operating system available in the University of California's *Berkeley Software Distribution*, commonly called *Berkeley UNIX* or *BSD UNIX*. By funding Bolt Beranek and Newman, Incorporated (*BBN*) to implement its TCP/IP protocols for use with UNIX and funding Berkeley to integrate the protocols with its software distribution, ARPA was able to reach over 90% of university computer science departments. The new protocol software came at a particularly significant time because many departments were just acquiring second or third computers and connecting them together with local area networks. The departments needed communication protocols.

The Berkeley software distribution became popular because it offered more than basic TCP/IP protocols. In addition to standard TCP/IP application programs, Berkeley offered a set of utilities for network services that resembled the UNIX services used on a single machine. The chief advantage of the Berkeley utilities lies in their similarity to standard UNIX. For example, an experienced UNIX user can quickly learn how to use Berkeley's remote file copy utility (*rcp*) because it behaves exactly like the UNIX file copy utility except that it allows users to copy files to or from remote machines.

Besides a set of utility programs, Berkeley UNIX provided a new operating system abstraction known as a *socket* that allowed application programs to access communication protocols. A generalization of the UNIX mechanism for I/O, the socket has options for several types of network protocols in addition to TCP/IP. Its design has been debated since its introduction, and many operating systems researchers have proposed alternatives. Independent of its overall merits, however, the introduction of the socket abstraction was important because it allowed programmers to use TCP/IP protocols with little effort. Thus, it encouraged researchers to experiment with TCP/IP.

The success of the TCP/IP technology and the Internet among computer science researchers led other groups to adopt it. Realizing that network communication would soon be a crucial part of scientific research, the National Science Foundation (*NSF*) took an active role in expanding the TCP/IP Internet to reach as many scientists as possible. In the late 1970s, NSF funded a project known as the *Computer Science NETwork* (*CSNET*), which had as its goal connecting all computer scientists. Starting in 1985, NSF began a program to establish access networks centered around its six supercomputer centers. In 1986 it expanded networking efforts by funding a new wide area backbone network, called the *NSFNET*†, that eventually reached all its supercomputer centers and tied them to the ARPANET. Finally, in 1986 NSF provided seed money for many regional networks, each of which now connects major scientific research institutions in a given area. All the NSF-funded networks use TCP/IP protocols, and all are part of the global Internet.

Within seven years of its inception, the Internet had grown to span hundreds of individual networks located throughout the United States and Europe. It connected nearly 20,000 computers at universities, government, and corporate research laboratories. Both the size and the use of the Internet continued to grow much faster than anticipated. By

†The term *NSFNET* is sometimes used loosely to mean all NSF-funded networking activities, but we will use it to refer to the backbone. The next chapter gives more details about the technology.

late 1987, it was estimated that the growth had reached 15% per month. By 2000, the global Internet reached over 50 million computers in 209 countries.

Early adoption of TCP/IP protocols and growth of the Internet has not been limited to government-funded projects. Major computer corporations connected to the Internet as did many other large corporations including: oil companies, the auto industry, electronics firms, pharmaceutical companies, and telecommunications carriers. Medium and small companies began connecting in the 1990s. In addition, many companies have used the TCP/IP protocols on their internal corporate internets even though they choose not to be part of the global Internet.

Rapid expansion introduced problems of scale unanticipated in the original design and motivated researchers to find techniques for managing large, distributed resources. In the original design, for example, the names and addresses of all computers attached to the Internet were kept in a single file that was edited by hand and then distributed to every site on the Internet. By the mid 1980s, it became apparent that a central database would not suffice. First, because computers were being added to the Internet at an increasing rate, requests to update the file would soon exceed the personnel available to process them. Second, even if a correct central file existed, network capacity was insufficient to allow either frequent distribution to every site or on-line access by each site.

New protocols were developed and a naming system was put in place across the global Internet that allows any user to resolve the name of a remote machine automatically. Known as the *Domain Name System* (*DNS*), the mechanism relies on machines called *name servers* to answer queries about names. No single machine contains the entire domain name database. Instead, data is distributed among a set of machines that use TCP/IP protocols to communicate among themselves when answering a query.

1.5 The Internet Architecture Board

Because the TCP/IP internet protocol suite did not arise from a specific vendor or from a recognized professional society, it is natural to ask, "who sets the technical direction and decides when protocols become standard?" The answer is a group known as the *Internet Architecture Board* (*IAB*†). The IAB provides the focus and coordination for much of the research and development underlying the TCP/IP protocols, and guides the evolution of the Internet. It decides which protocols are a required part of the TCP/IP suite and sets official policies.

Formed in 1983 when ARPA reorganized the Internet Control and Configuration Board, the IAB inherited much of its charter from the earlier group. Its initial goals were to encourage the exchange of ideas among the principals involved in research related to TCP/IP and the Internet, and to keep researchers focused on common objectives. Through the first six years, the IAB evolved from an ARPA-specific research group into an autonomous organization. During these years, each member of the IAB chaired an *Internet Task Force* charged with investigating a problem or set of issues deemed to be important. The IAB consisted of approximately ten task forces, with charters ranging from one that investigated how the traffic load from various applica-

†IAB originally stood for *Internet Activities Board*.

tions affects the Internet to one that handled short term Internet engineering problems. The IAB met several times each year to hear status reports from each task force, review and revise technical directions, discuss policies, and exchange information with representatives from agencies like ARPA and NSF, who funded Internet operations and research.

The chairman of the IAB had the title *Internet Architect* and was responsible for suggesting technical directions and coordinating the activities of the various task forces. The IAB chairman established new task forces on the advice of the IAB and also represented the IAB to others.

Newcomers to TCP/IP are sometimes surprised to learn that the IAB did not manage a large budget; although it set direction, it did not fund most of the research and engineering it envisioned. Instead, volunteers performed much of the work. Members of the IAB were each responsible for recruiting volunteers to serve on their task forces, for calling and running task force meetings, and for reporting progress to the IAB. Usually, volunteers came from the research community or from commercial organizations that produced or used TCP/IP. Active researchers participated in Internet task force activities for two reasons. On one hand, serving on a task force provided opportunities to learn about new research problems. On the other hand, because new ideas and problem solutions designed and tested by task forces often became part of the TCP/IP Internet technology, members realized that their work had a direct, positive influence on the field.

1.6 The IAB Reorganization

By the summer of 1989, both the TCP/IP technology and the Internet had grown beyond the initial research project into production facilities on which thousands of people depended for daily business. It was no longer possible to introduce new ideas by changing a few installations overnight. To a large extent, the literally hundreds of commercial companies that offer TCP/IP products determined whether products would interoperate by deciding when to incorporate changes in their software. Researchers who drafted specifications and tested new ideas in laboratories could no longer expect instant acceptance and use of the ideas. It was ironic that the researchers who designed and watched TCP/IP develop found themselves overcome by the commercial success of their brainchild. In short, TCP/IP became a successful, production technology and the market place began to dominate its evolution.

To reflect the political and commercial realities of both TCP/IP and the Internet, the IAB was reorganized in the summer of 1989. The chairmanship changed. Researchers were moved from the IAB itself to a subsidiary group and a new IAB board was constituted to include representatives from the wider community.

Figure 1.1 illustrates the IAB organization and the relationship of subgroups.

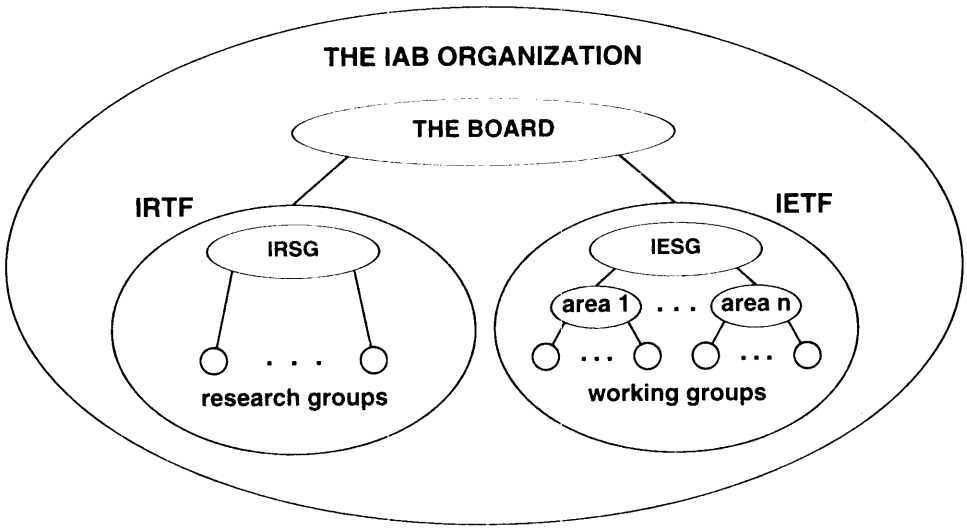

Figure 1.1 The structure of the IAB after the 1989 reorganization.

As Figure 1.1 shows, in addition to the board itself, the IAB organization contained two major groups: the *Internet Research Task Force* (*IRTF*) and the *Internet Engineering Task Force* (*IETF*).

As its name implies, the IETF concentrates on short-term or medium-term engineering problems. The IETF existed in the original IAB structure, and its success provided part of the motivation for reorganization. Unlike most IAB task forces, which were limited to a few individuals who focused on one specific issue, the IETF was large — before the reorganization, it had grown to include dozens of active members who worked on many problems concurrently. It was divided into over 20 *working groups*, each focusing on a specific problem. Working groups held individual meetings to formulate problem solutions. In addition, the entire IETF met regularly to hear reports from working groups and discuss proposed changes or additions to the TCP/IP technology. Usually held three times annually, full IETF meetings attracted hundreds of participants and spectators. The IETF had become too large for the chairman to manage.

Because the IETF was known throughout the Internet, and because its meetings were widely recognized and attended, the reorganized IAB structure retains the IETF, but splits it into approximately ten areas, each with its own manager. The IETF chairman and the area managers comprise the *Internet Engineering Steering Group* (*IESG*), the individuals responsible for coordinating the efforts of IETF working groups. The name "IETF" now refers to the entire body, including the chairman, area managers, and all members of working groups.

Created during the reorganization, the Internet Research Task Force is the research counterpart to the IETF. The IRTF coordinates research activities related to TCP/IP protocols or internet architecture in general. Like the IETF, the IRTF has a small group, called the *Internet Research Steering Group (IRSG)*, that sets priorities and coordinates research activities. Unlike the IETF, the IRTF is currently a much smaller and less active organization. In fact, most of the research is being done within the IETF.

1.7 The Internet Society

In 1992, as the Internet moved away from its U.S. government roots, a society was formed to encourage participation in the Internet. Called the *Internet Society (ISOC)*, the group is an international organization inspired by the National Geographic Society. The host for the IAB, the Internet Society continues to help people join and use the Internet around the world.

1.8 Internet Request For Comments

We have said that no vendor owns the TCP/IP technology nor does any professional society or standards body. Thus, the documentation of protocols, standards, and policies cannot be obtained from a vendor. Instead, the documentation is placed in on-line repositories and made available at no charge.

Documentation of work on the Internet, proposals for new or revised protocols, and TCP/IP protocol standards all appear in a series of technical reports called Internet *Requests For Comments*, or *RFC*s. RFCs can be short or long, can cover broad concepts or details, and can be standards or merely proposals for new protocols†. While RFCs are not refereed in the same way as academic research papers, they are edited. For many years, a single individual, Jon Postel‡, served as RFC editor. The task of editing RFCs now falls to area managers of the IETF; the IESG as a whole approves new RFCs.

Finally, a few reports pertinent to the Internet were published in an earlier, parallel series of reports called *Internet Engineering Notes*, or *IEN*s. Although the IEN series is no longer active, not all IENs appear in the RFC series. There are references to RFCs (and still a few to IENs) throughout the text.

The RFC series is numbered sequentially in the chronological order RFCs are written. Each new or revised RFC is assigned a new number, so readers must be careful to obtain the highest numbered version of a document; an RFC index is available to help identify the correct version.

To make document retrieval quicker, many sites around the world store copies of RFCs and make them available to the community. One can obtain RFCs by postal mail, by electronic mail, or directly across the Internet using a file transfer program. In addition, preliminary versions of RFC documents, which are known as *Internet drafts*,

---

†Appendix *1* contains an introduction to RFCs that examines the diversity of RFCs, including jokes that have appeared.

‡Jon passed away in the fall of 1998. He was one of the pioneers who made significant contributions to TCP/IP and the Internet. Those of us who knew him feel the loss deeply.

are also available. Ask a local network expert how to obtain RFCs or Internet drafts at your site, or refer to Appendix *1* for further instructions on how to retrieve them.

1.9 Internet Protocols And Standardization

Readers familiar with data communication networks realize that a myriad of communication protocol standards exist. Many of them precede the Internet, so the question arises, "Why did the Internet designers invent new protocols when so many international standards already existed?" The answer is complex, but follows a simple maxim:

> *Use existing protocol standards whenever such standards apply; invent new protocols only when existing standards are insufficient, and be prepared to use new standards when they become available and provide equivalent functionality.*

So, despite appearances to the contrary, the TCP/IP Internet Protocol Suite was not intended to ignore or avoid extant standards. It came about merely because none of the existing protocols satisfied the need for an interoperable internetworking communication system.

1.10 Future Growth And Technology

Both the TCP/IP technology and the Internet continue to evolve. New protocols are being proposed; old ones are being revised. NSF added considerable complexity to the system by introducing a backbone network, regional networks, and hundreds of campus networks. Other groups around the world continue to connect to the Internet as well. The most significant change comes not from added network connections, however, but from additional traffic. As new users connect to the Internet and new applications appear, traffic patterns change. When physicists, chemists, and biologists began to use the Internet, they exchanged files of data collected from experiments. Files of scientific data were large compared to electronic mail messages. As the Internet became popular and users began to browse information using services like the *World Wide Web*, traffic patterns increased again.

To accommodate growth in traffic, the capacity of the NSFNET backbone was increased three times. The final version, known as *ANSNET* after the company that supplied it, had a capacity approximately 840 times larger than the original. Since 1995, companies known as *Internet Service Providers* (*ISP*s) have each built their own backbone network, many of which have significantly more capacity than the last government-funded backbone. At the current time, it is difficult to foresee an end to the need for more capacity.

Growth in demands for networking is not unexpected. The computer industry has enjoyed a continual demand for increased processing power and larger data storage for many years. Users have only begun to understand how to use networks. In the future we can expect continual increases in the demand for communications. Soon, for example, TCP/IP technologies will be used for telephone and video services as well as data services. Thus, higher-capacity communication technologies will be needed to accommodate the growth.

Figure 1.2 summarizes expansion of the Internet and illustrates an important component of growth: much of the change in complexity has arisen because multiple groups now manage various parts of the whole. Because the technology was developed when a single person at ARPA had control of all aspects of the Internet, the designs of many subsystems depended on centralized management and control. As the Internet grew, responsibility and control were divided among multiple organizations. In particular, as the Internet became global, the operation and management needed to span multiple countries. Much of the effort since the early 1990s has been directed toward finding ways to extend the design to accommodate decentralized management.

	number of networks	number of computers	number of users	number of managers
1980	10	10^2	10^2	10^0
1990	10^3	10^5	10^6	10^1
2000	10^5	10^7	10^8	10^2

Figure 1.2 Growth of the connected Internet. In addition to traffic increases that result from increased size, the Internet faces complexity that results from decentralized management of both development and operations.

1.11 Organization Of The Text

The material on TCP/IP has been written in three volumes. This volume presents the TCP/IP technology, applications that use it, and the architecture of the global Internet in more detail. It discusses the fundamentals of protocols like TCP and IP, and shows how they fit together in an internet. In addition to giving details, the text highlights the general principles underlying network protocols, and explains why the TCP/IP protocols adapt easily to so many underlying physical network technologies. Volume II discusses in depth the internal details of the TCP/IP protocols and shows how they are implemented. It presents code from a working system to illustrate how the individual protocols work together, and contains details useful to people responsible

for building a corporate internet. Volume III shows how distributed applications use TCP/IP for communication. It focuses on the client-server paradigm, the basis for all distributed programming. It discusses the interface between programs and protocols†, and shows how client and server programs are organized. In addition, Volume III describes the remote procedure concept, middleware, and shows how programmers use tools to build client and server software.

So far, we have talked about the TCP/IP technology and the Internet in general terms, summarizing the services provided and the history of their development. The next chapter provides a brief summary of the type of network hardware used throughout the Internet. Its purpose is not to illuminate nuances of a particular vendor's hardware, but to focus on the features of each technology that are of primary importance to an internet architect. Later chapters delve into the protocols and the Internet, fulfilling three purposes: they explore general concepts and review the Internet architectural model, they examine the details of TCP/IP protocols, and they look at standards for high-level services like electronic mail and electronic file transfer. Chapters *3* through *14* review fundamental principles and describe the network protocol software found in any machine that uses TCP/IP. Later chapters describe services that span multiple machines, including the propagation of routing information, name resolution, and applications like electronic mail.

Two appendices follow the main text. The first appendix contains a guide to RFCs. It expands on the description of RFCs found in this chapter, and gives examples of information that can be found in RFCs. It describes in detail how to obtain RFCs by electronic mail, postal mail, and file transfer. Finally, because the standard RFC index comes in chronological order, the appendix presents a list of RFCs organized by topic to make it easier for beginners to find RFCs pertinent to a given subject.

The second appendix contains an alphabetical list of terms and abbreviations used throughout the literature and the text. Because beginners often find the new terminology overwhelming and difficult to remember, they are encouraged to use the alphabetical list instead of scanning back through the text.

1.12 Summary

An internet consists of a set of connected networks that act as a coordinated whole. The chief advantage of an internet is that it provides universal interconnection while allowing individual groups to use whatever network hardware is best suited to their needs. We will examine principles underlying internet communication in general and the details of one internet protocol suite in particular. We will also discuss how internet protocols are used in an internet. Our example technology, called TCP/IP after its two main protocols, was developed by the Advanced Research Projects Agency. It provides the basis for the global Internet, a large, operational internet that connects universities, corporations, and government departments in many countries around the world. The global Internet is expanding rapidly.

†Volume III is available in three versions: one that uses the Unix *socket interface* interface in examples, a second that uses the *Transport Layer Interface (TLI)*, and a third that uses the *Windows Sockets Interface* defined by Microsoft.

FOR FURTHER STUDY

Cerf's *A History Of The ARPANET* [1989] and *History of the Internet Activities Board* [RFC 1160] provide fascinating reading and point the reader to early research papers on TCP/IP and internetworking. Denning [Nov-Dec 1989] provides a different perspective on the history of the ARPANET. Jennings et. al. [1986] discusses the importance of computer networking for scientists. Denning [Sept-Oct 1989] also points out the importance of internetworking and gives one possible scenario for a world-wide Internet. The U.S. Federal Coordinating Committee for Science, Engineering and Technology [*FCCSET*] suggested networking should be a national priority.

The IETF (*ietf.org*) publishes minutes from its regular meetings. The Internet Society (*www.isoc.org*) produces newsletters that discuss the penetration of the Internet in countries around the world. The World Wide Web Consortium (*w3c.org*) produces protocols and standards for Web technologies. Finally, the reader is encouraged to remember that the TCP/IP protocol suite and the Internet continue to evolve; new information can be found in RFCs and at conferences such as the annual ACM SIGCOMM Symposium and NETWORLD+INTEROP events held around the world.

EXERCISES

1.1 Explore application programs at your site that use TCP/IP.

1.2 Plot the growth of TCP/IP technology and Internet access at your organization. How many computers, users, and networks were connected each year?

1.3 TCP/IP products account for several billion dollars per year in gross revenue. Read trade publications to find a list of vendors offering such products.

2

Review Of Underlying Network Technologies

2.1 Introduction

It is important to understand that the Internet is not a new kind of physical network. It is, instead, a method of interconnecting physical networks and a set of conventions for using networks that allow the computers they reach to interact. While network hardware plays only a minor role in the overall design, understanding the internet technology requires one to distinguish between the low-level mechanisms provided by the hardware itself and the higher-level facilities that the TCP/IP protocol software provides. It is also important to understand how the interfaces supplied by underlying packet-switched technology affect our choice of high-level abstractions.

This chapter introduces basic packet-switching concepts and terminology, and then reviews some of the underlying network hardware technologies that have been used in TCP/IP internets. Later chapters describe how these networks are interconnected and how the TCP/IP protocols accommodate vast differences in the hardware. While the list presented here is certainly not comprehensive, it clearly demonstrates the variety among physical networks over which TCP/IP operates. The reader can safely skip many of the technical details, but should try to grasp the idea of packet switching and try to imagine building a homogeneous communication system using such heterogeneous hardware. Most important, the reader should look closely at the details of the physical address schemes the various technologies use; later chapters will discuss in detail how high-level protocols use physical addresses.

2.2 Two Approaches To Network Communication

Whether they provide connections between one computer and another or between a terminal and a computer, communication networks can be divided into two basic types: *connection-oriented* (sometimes called *circuit-switched*) and *connectionless* (sometimes called *packet-switched*†). Connection-oriented networks operate by forming a dedicated *connection* or *circuit* between two points. The U.S. telephone system uses a connection-oriented technology — a telephone call establishes a connection from the originating phone through the local switching office, across trunk lines, to a remote switching office, and finally to the destination telephone. While a connection is in place, the phone equipment samples the microphone repeatedly, encodes the samples digitally, and transmits them across the connection to the receiver. The sender is guaranteed that the samples can be delivered and reproduced because the connection provides a guaranteed data path of 64 Kbps (thousand bits per second), the rate needed to send digitized voice. The advantage of connection-oriented networking lies in its guaranteed capacity: once a circuit is established, no other network activity will decrease the capacity of that circuit. One disadvantage of connection-oriented technology arises from cost: circuit costs are fixed, independent of use. For example, one pays a fixed rate for a phone call, even when the two parties do not talk.

Connectionless networks, the type often used to connect computers, take an entirely different approach. In a connectionless network, data to be transferred across a network is divided into small pieces called *packets* that are multiplexed onto high capacity intermachine connections. A packet, which usually contains only a few hundred bytes of data, carries identification that enables the network hardware to know how to send it to the specified destination. For example, a large file to be transmitted between two machines must be broken into many packets that are sent across the network one at a time. The network hardware delivers the packets to the specified destination, where software reassembles them into a single file again. The chief advantage of packet-switching is that multiple communications among computers can proceed concurrently, with intermachine connections shared by all pairs of computers that are communicating. The disadvantage, of course, is that as activity increases, a given pair of communicating computers receives less of the network capacity. That is, whenever a packet switched network becomes overloaded, computers using the network must wait before they can send additional packets.

Despite the potential drawback of not being able to guarantee network capacity, connectionless networks have become extremely popular. The motivations for adopting packet switching are cost and performance. Because multiple computers can share the network bandwidth, fewer connections are required and cost is kept low. Because engineers have been able to build high speed network hardware, capacity is not usually a problem. So many computer interconnections use connectionless networks that, throughout the remainder of this text, we will assume the term *network* refers to a connectionless network unless otherwise stated.

---

†In fact, it is possible to build hybrid hardware technologies; for our purposes, only the difference in functionality is important.

2.3 Wide Area And Local Area Networks

Data networks that span large geographical distances (e.g., the continental U.S.) are fundamentally different from those that span short distances (e.g., a single room). To help characterize the differences in capacity and intended use, packet switched technologies are often divided into two broad categories: *wide area networks* (*WANs*) and *Local Area Networks* (*LANs*). The two categories do not have formal definitions. Instead, vendors apply the terms loosely to help customers distinguish among technologies.

WAN technologies, sometimes called *long haul networks*, provide communication over long distances. Most WAN technologies do not limit the distance spanned; a WAN can allow the endpoints of a communication to be arbitrarily far apart. For example, a WAN can span a continent or can join computers across an ocean. Usually, WANs operate at slower speeds than LANs, and have much greater delay between connections. Typical speeds for a WAN range from 1.5 Mbps to 155 Mbps (million bits per second). Delays across a WAN can vary from a few milliseconds to several tenths of a second†.

LAN technologies provide the highest speed connections among computers, but sacrifice the ability to span long distances. For example, a typical LAN spans a small area like a single building or a small campus, and operates between 10 Mbps and 2 Gbps (billion bits per second). Because LAN technologies cover short distances, they offer lower delays than WANs. The delay across a LAN can be as short as a few tenths of a millisecond or as long as 10 milliseconds.

We have already stated the general tradeoff between speed and distance: technologies that provide higher speed communication operate over shorter distances. There are other differences among the technologies as well. In LAN technologies, each computer usually contains a device known as a *Network Interface Card* (*NIC*) that connects the machine directly to the network. The network itself need not contain much intelligence; it can depend on electronic interface devices in the attached computers to generate and receive the complex electrical signals. In WAN technologies, a network usually consists of a series of complex computers called *packet switches* interconnected by long-distance communication lines. The size of the network can be extended by adding a new switch and another communication line. Attaching a user's computer to a WAN means connecting it to one of the packet switches. Each switch along a path in the WAN introduces delay when it receives a packet and forwards it to the next switch. Thus, the larger the WAN becomes the longer it takes to route traffic across it.

This book discusses software that hides the technological differences among networks and makes interconnection independent of the underlying hardware. To appreciate design choices in the software, it is necessary to understand how it relates to network hardware. The next sections present examples of network technologies that have been used in the Internet, showing some of the differences among them. Later chapters show how the TCP/IP software isolates such differences and makes the communication system independent of the underlying hardware technology.

---

†Such long delays result from WANs that communicate by sending signals to a satellite orbiting the earth.

2.3.1 Network Hardware Addresses

Each network hardware technology defines an *addressing mechanism* that computers use to specify the destination for a packet. Every computer attached to a network is assigned a unique address, usually an integer. A packet sent across a network includes a *destination address field* that contains the address of the intended recipient. The destination address appears in the same location in all packets, making it possible for the network hardware to examine the destination address easily. A sender must know the address of the intended recipient, and must place the recipient's address in the destination address field of a packet before transmitting the packet.

Each hardware technology specifies how computers are assigned addresses. The hardware specifies, for example, the number of bits in the address as well as the location of the destination address field in a packet. Although some technologies use compatible addressing schemes, many do not. This chapter contains a few examples of hardware addressing schemes; later chapters explain how TCP/IP accommodates diverse hardware addressing schemes.

2.4 Ethernet Technology

Ethernet is the name given to a popular packet-switched LAN technology invented at Xerox PARC in the early 1970s. Xerox Corporation, Intel Corporation, and Digital Equipment Corporation standardized Ethernet in 1978; IEEE released a compatible version of the standard using the standard number *802.3*. Ethernet has become the most popular LAN technology; it now appears in virtually all corporate networks as well as many small installations. Because Ethernet is so popular, many variants exist. Although the original wiring scheme has been superceded, understanding the original design helps clarify the intent and some of the design decisions. Thus, we will discuss the original design first, and then cover variants.

Formally known as *10Base5*, the original Ethernet design uses a coaxial cable as Figure 2.1 illustrates.

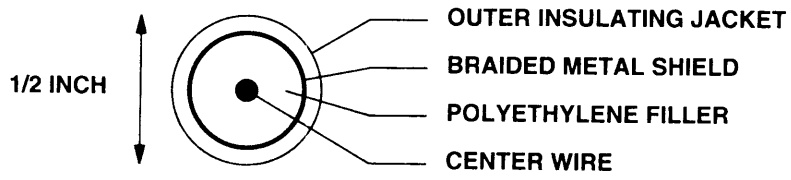

Figure 2.1 A cross-section of the coaxial cable used in the original Ethernet.

Called the *ether*, the cable itself is completely passive; all the active electronic components needed to make the network function are associated with the computers attached to the network. Each Ethernet cable is about 1/2 inch in diameter and up to 500

meters long. A resistor is added between the center wire and shield at each end to prevent reflection of electrical signals.

The connection between a computer and the original Ethernet coaxial cable requires a hardware device called a *transceiver*. Physically, the connection between a transceiver and the inner wire of an Ethernet cable enters through a small hole in the outer layers of the cable as Figure 2.2 illustrates. Technicians often use the term *tap* to describe such connections. Usually, small metal pins mounted in the transceiver go through the hole and provide electrical contacts to the center wire and the braided shield. Some manufacturers' connectors require that the cable be cut and a "T" inserted.

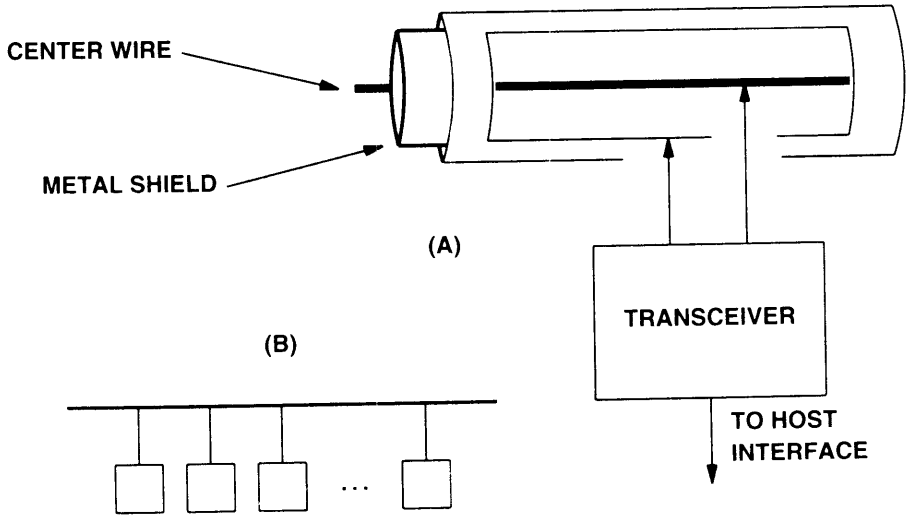

Figure 2.2 (a) A cutaway view of an Ethernet cable showing the details of electrical connections between a transceiver and the cable, and (b) the schematic diagram of an Ethernet with many computers connected.

Each connection to an original Ethernet uses two major electronic components. A *transceiver* connects to the center wire and braided shield on the cable, sensing and sending signals on the ether. A *host interface card* or *host adapter* plugs into the computer's bus (e.g., to a motherboard) and connects to the transceiver.

A transceiver is a small piece of hardware usually found physically adjacent to the ether. In addition to the analog hardware that senses and controls electrical signals on the ether, a transceiver contains digital circuitry that allows it to communicate with a digital computer. The transceiver senses when the ether is in use and translates analog electrical signals on the ether to (and from) digital form. A cable called the *Attachment Unit Interface* (*AUI*) cable connects the transceiver to an adapter board in a host com-

puter. Informally called a *transceiver cable*, the AUI cable contains many wires. The wires carry the electrical power needed to operate the transceiver, the signals that control the transceiver operation, and the contents of the packets being sent or received. Figure 2.3 illustrates how the components form a connection between a bus in a computer system and an Ethernet cable.

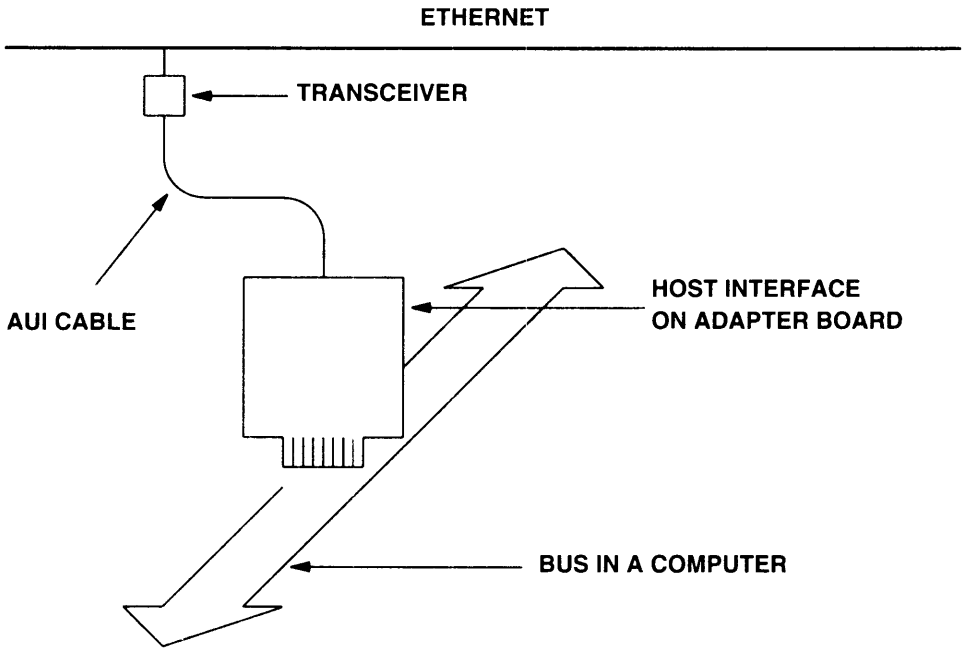

Figure 2.3 The two main electronic components that form a connection between a computer's bus and an Ethernet in the original scheme. The AUI cable that connects the host interface to the transceiver carry power and signals to control transceiver operation as well as packets being transmitted or received.

Each host interface controls the operation of one transceiver according to instructions it receives from the computer software. To the operating system software, the interface appears to be an input/output device that accepts basic data transfer instructions from the computer, controls the transceiver to carry them out, interrupts when the task has been completed, and reports status information. Although a transceiver is a simple hardware device, the host interface can be complex (e.g., some interfaces contain a microprocessor used to control transfers between the computer memory and the ether).

In practice, organizations that use the original Ethernet wiring in a conventional office environment run the Ethernet cable along the ceiling in each hall, and arrange for a connection from each office to attach to the cable. Figure 2.4 illustrates the resulting physical wiring scheme.

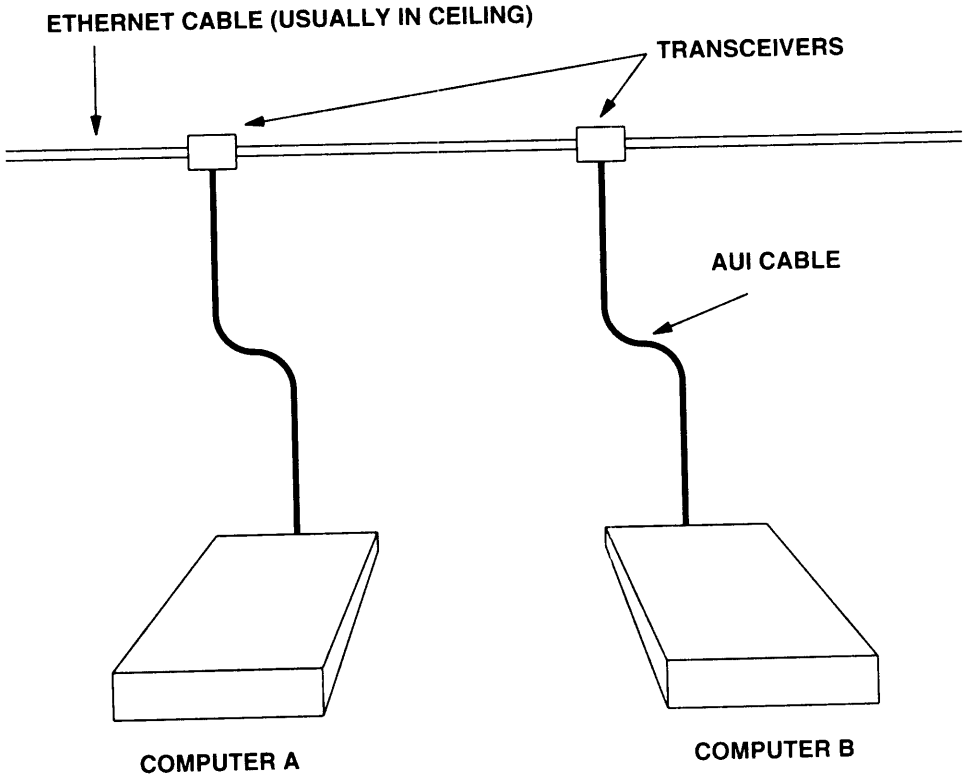

Figure 2.4 The physical connection of two computers to an Ethernet using the original wiring scheme. In an office environment, the Ethernet cable is usually placed in the hallway ceiling; each office has an AUI cable that connects a computer in the office to a transceiver attached to the Ethernet cable.

2.4.1 Thin-Wire Ethernet

Several components of the original Ethernet technology have undesirable properties. For example, because a transceiver contains electronic components, it has a nontrivial cost. Furthermore, because transceivers are located with the cable and not with computers, locating or replacing them is difficult. The coaxial cable that forms the ether is difficult to install. In particular, to provide maximum protection against electrical interference from devices like electric motors, the cable contains heavy shielding that makes it difficult to bend. Finally, the AUI cable is also thick and difficult to bend.

To reduce costs for environments like offices that do not contain much electrical interference, engineers developed an alternative Ethernet wiring scheme. Formally known as *10Base2* and usually called *thin-wire Ethernet* or *thinnet*†, the alternative coaxial cable is thinner, less expensive, and more flexible. However, thin-wire Ethernet

†To contrast it with thin-wire, the original Ethernet cable became known as *thick Ethernet*, or *thicknet*.

has some disadvantages. Because it does not provide as much protection from electrical interference, thin-wire Ethernet cannot be placed adjacent to powerful electrical equipment like that found in a factory. Furthermore, thin-wire Ethernet covers somewhat shorter distances and supports fewer computer connections per network than thick Ethernet.

When designing thin-wire Ethernet, engineers replaced costly transceiver hardware with special high-speed digital circuits, and provided a direct connection from a computer to the network. Thus, in a thin-wire scheme, a computer contains both the host interface and the circuitry that connects to the cable. Manufacturers of small computers and workstations find thin-wire Ethernet an especially attractive scheme because they can integrate Ethernet hardware into single board computers and mount connectors directly on the back of the computer.

Because a thin-wire Ethernet connects directly from one computer to another, the wiring scheme works well when many computers occupy a single room. The thin-wire cable runs directly from one computer to the next. To add a new computer, one only needs to link it into the chain. Figure 2.5 illustrates the connections used with thin-wire Ethernet.

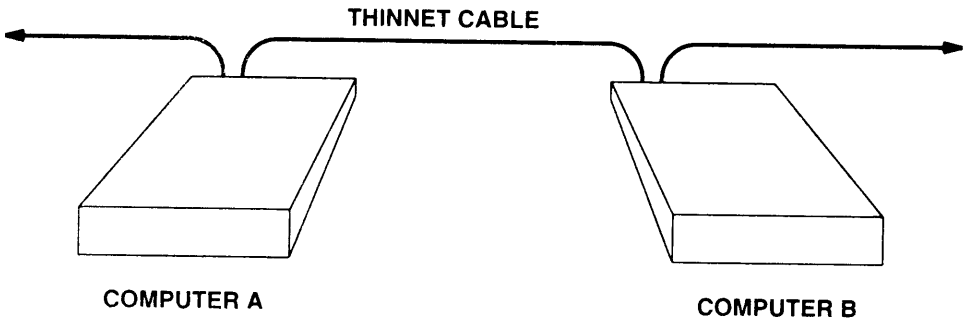

Figure 2.5 The physical connection of two computers using the thinnet wiring scheme. The ether passes directly from one computer to another; no external transceiver hardware is required.

Thin-wire Ethernets were designed to be easy to connect and disconnect. Thin-wire uses *BNC connectors*, which do not require tools to attach a computer to the cable. Thus, a user can connect a computer to a thin-wire Ethernet without the aid of a technician. Of course, allowing users to manipulate the ether has disadvantages: if a user disconnects the ether, it prevents all machines on the ether from communicating. In many situations, however, the advantages outweigh the disadvantages.

2.4.2 Twisted Pair Ethernet

Advances in technology have made it possible to build Ethernets that do not need the electrical shielding of a coaxial cable. Called *twisted pair Ethernet*, the technology allows a computer to access an Ethernet using conventional unshielded copper wires similar to the wires used to connect telephones†. The advantages of using twisted pair wiring are that it further reduces costs and protects other computers on the network from a user who disconnects a single computer. In some cases, a twisted pair technology can make it possible for an organization to use Ethernet over existing wiring; in others, the needed wiring (called *category 5 cable*) is cheaper and easier to install than coaxial cable.

Formally known as *10Base-T*, the first twisted pair Ethernet operated at 10 Mbps, exactly like thick or thin Ethernet. A set of eight wires (four pairs) is used to connect each computer to an Ethernet *hub* as Figure 2.6 shows.

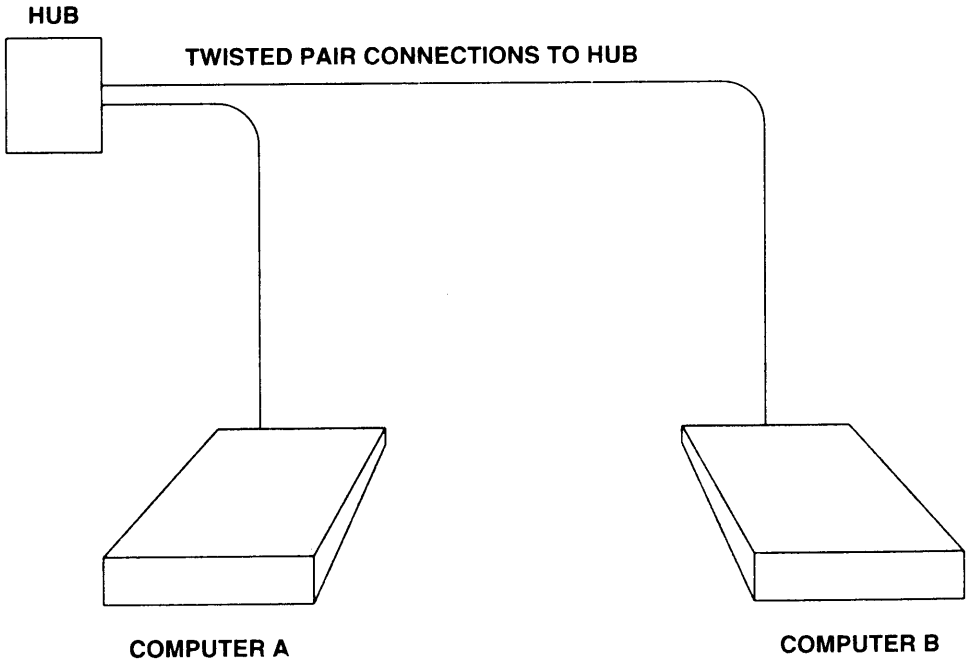

Figure 2.6 An illustration of Ethernet using twisted pair wiring. Each computer connects to a hub over four pairs of wire.

The hub is an electronic device that simulates the signals on an Ethernet cable. Physically, a hub consists of a small box that usually resides in a wiring closet; a connection between a hub and a computer must be less than 100 meters long. A hub requires power, and can allow authorized personnel to monitor and control its operation

†The term *twisted pair* arises because conventional telephone wiring uses the technique of twisting the wires to avoid interference.

over the network. To the host interface in a computer, a connection to a hub appears to operate the same way as a connection to a transceiver. That is, an Ethernet hub provides the same communication capability as a thick or thin Ethernet; hubs merely offer an alternative wiring scheme.

2.4.3 Ethernet Capacity

Although the wiring scheme evolved from the original thick cable to thin cable and finally to twisted pair, much of the original Ethernet design remained the same. In particular, the initial twisted pair Ethernet design operates at the same rate as the original thick Ethernet, which means that data can be transmitted at 10 million bits per second. Although a computer can generate data at Ethernet speed, raw network speed should not be thought of as the rate at which two computers can exchange data. Instead, network speed should be thought of as a measure of total traffic capacity. Think of a network as a highway connecting multiple cities, and think of packets as cars on the highway. High bandwidth makes it possible to carry heavy traffic loads, while low bandwidth means the highway cannot carry as much traffic. A 10 Mbps Ethernet, for example, can handle a few computers that generate heavy loads, or many computers that generate light loads.

In the late 1970s when Ethernet was standardized, a LAN operating at 10 Mbps had more than sufficient capacity for many computers because the available CPU speeds and network interface hardware prohibited a given computer from transmitting data rapidly. By the mid 1990s, however, CPU speeds had increased dramatically as had the use of networks. Consequently, an Ethernet operating at 10 Mbps did not have sufficient capacity to act as a central corporate backbone for even a moderate sized corporation — Ethernet had become a bottleneck.

2.4.4 Fast Ethernet

To overcome the throughput limitation of Ethernet, engineers designed a new version of Ethernet that operates an order of magnitude faster. Known formally as *100Base-T*, the technology is usually called *Fast Ethernet*. As the formal name implies, Fast Ethernet uses category 5 twisted pair wiring, the same wiring used for 10Base-T. However, through clever use of the wires, Fast Ethernet allows a station to transmit or receive data at 100 Mbps.

To understand the significance of the increase in capacity, it is important to understand two facts. First, although computers have become faster, few computer systems can transmit data at a sustained rate of 100 Mbps. Second, the 100Base-T standard did not change other parts of the Ethernet standard. In particular, the maximum packet size remains the same as for 10Base-T. These two facts imply that Fast Ethernet is not optimized to provide the highest possible throughput between a pair of computers. Instead, the design is optimized to allow more stations and more total traffic.

2.4.5 10/100 Ethernet

Soon after the invention of Fast Ethernet, manufacturers began to build devices that could accept either a 10 or 100 Mbps connection. The technology, which is known as *dual-speed Ethernet* or *10/100 Ethernet*, is available for computer interfaces as well as for hubs. In essence, all 100Base-T hardware interjects extra signals, making it possible for the hardware at one end of a cable to know which hardware type is connected to the other end. In fact, as long as all eight wires connect to the RJ-45 connector, the cabling and connectors used with 10Base-T are compatible with the cable and connectors used for 100Base-T.

Although 10/100 hardware is slightly more expensive than 10Base-T hardware, it has become extremely popular. Dual speed devices are especially helpful during a transition from 10 Mbps technology to 100 Mbps technology. For example, consider a computer that has a 10/100 interface card. If the computer is connected to a 10Base-T hub, the hardware in the card will automatically detect the speed and communicate at 10 Mbps. If the same computer is then unplugged from the 10Base-T hub and connected to a 100Base-T hub, the hardware will automatically detect the new speed and begin transmitting at 100 Mbps. The transition in speed is completely automatic: neither the software nor the hardware needs to be reconfigured.

2.4.6 Gigabit Ethernet

By the late 1990s, as the market share of 100Base-T Ethernet began to grow, it became obvious that there was a demand for even higher capacity Ethernet. Consequently, engineers extended the Ethernet technology to a bit rate of 1 Gbps (gigabits per second). Known as *1000Base-T*, the high throughput rate makes the technology extremely attractive for use in corporate backbone networks, where traffic from many computers passes through the network. The high data rate does have a slight disadvantage — it makes gigabit Ethernet more susceptible to electrical interference. Consequently, wiring that operates well with 10Base-T or even 100Base-T may not work well with 1000Base-T.

Like Fast Ethernet, the design of gigabit Ethernet was optimized for total throughput. The original packet format and maximum packet size were retained, making packets used on 10Base-T, 100Base-T and 1000Base-T networks interchangeable. Consequently, it is possible to collect traffic from ten 100Base-T Ethernets, each running at full speed, and pass the traffic across a single 1000Base-T network.

2.4.7 Properties of an Ethernet

Ethernet was designed to be a shared bus technology that supports broadcast, uses best-effort delivery semantics, and has distributed access control. The topology is called a *shared bus* because all stations connect to a single, shared communication channel; it is called a *broadcast technology* because all stations receive every transmission, making it possible to transmit a packet to all stations at the same time. The

method used to direct packets from one station to just one other station or a subset of all stations will be discussed later. For now, it is enough to understand that the lowest level hardware does not distinguish among transmissions — a hub passes all packets to each host interface, which chooses packets the computer should receive and filters out all others. Ethernet is called a *best-effort delivery* mechanism because the hardware provides no information to the sender about whether the packet was delivered. For example, if the destination machine happens to be powered down, packets sent to it will be lost, and the sender will not be notified. We will see later how the TCP/IP protocols accommodate best-effort delivery hardware.

Ethernet access control is distributed because, unlike some network technologies, Ethernet has no central authority to grant access. The Ethernet access scheme is called *Carrier Sense Multiple Access* with *Collision Detect* (*CSMA/CD*). It is *CSMA* because multiple machines can access an Ethernet simultaneously and each machine determines whether the network is idle by sensing whether a carrier wave is present. When a host interface has a packet to transmit, it listens to see if a message is being transmitted (i.e., performs carrier sensing). When no transmission is sensed, the host interface starts transmitting. Each transmission is limited in duration because there is a maximum packet size. Furthermore, the hardware must observe a minimum idle time between transmissions, which means that no single pair of communicating machines can use the network without giving other machines an opportunity for access.

2.4.8 Collision Detection And Recovery

When a station begins transmission, the signal does not reach all parts of the network simultaneously. Instead it travels along copper wires at approximately 70% of the speed of light. Thus, it is possible for two transceivers to both sense that the network is idle and begin transmission simultaneously. When the two electrical signals cross they become scrambled, meaning that neither remains meaningful. Such incidents are called *collisions*.

The Ethernet handles collisions in an ingenious fashion. Each station monitors the cable while it is transmitting to see if a foreign signal interferes with its transmission. Technically, the monitoring is called *collision detection* (*CD*), making the Ethernet a CSMA/CD network. When a collision is detected, the host interface aborts transmission, waits for activity to subside, and tries again. Care must be taken or the network could wind up with all stations busily attempting to transmit and every transmission producing a collision. To help avoid such situations, Ethernet uses a binary exponential backoff policy where a sender delays a random time after the first collision, doubles the range if a second attempt to transmit also produces a collision, quadruples the range if a third attempt results in a collision, and so on. The motivation for exponential backoff is that in the unlikely event many stations attempt to transmit simultaneously, a severe traffic jam could occur. In such a jam, there is a high probability two stations will choose random backoffs that are close together. Thus, the probability of another collision is high. By doubling the range of the random delay, the exponential backoff strategy quickly spreads the stations' attempts to retransmit over a reasonably long period of time, making the probability of further collisions extremely small.

2.4.9 Ethernet Hardware Addresses

Ethernet defines a 48-bit addressing scheme. Each computer attached to an Ethernet network is assigned a unique 48-bit number known as its *Ethernet address*. To assign an address, Ethernet hardware manufacturers purchase blocks of Ethernet addresses† and assign them in sequence as they manufacture Ethernet interface hardware. Thus, no two hardware interfaces have the same Ethernet address.

Usually, the Ethernet address is fixed in machine readable form on the host interface hardware. Because each Ethernet address belongs to a hardware device, they are sometimes called *hardware addresses*, *physical addresses*, *media access (MAC) addresses*, or *layer 2 addresses*. Note the following important property of Ethernet physical addresses:

> *Physical addresses are associated with the Ethernet interface hardware; moving the hardware interface to a new machine or replacing a hardware interface that has failed changes the machine's physical address.*

Knowing that Ethernet physical addresses can change will make it clear why higher levels of the network software are designed to accommodate such changes.

The host interface hardware examines packets and determines the packets that should be sent to the host. Recall that each interface receives a copy of every packet that passes through a hub — even those addressed to other machines. The host interface uses the destination address field in a packet as a filter. The interface ignores those packets that are addressed to other machines, and passes to the host only those packets addressed to it. The addressing mechanism and hardware filter are needed to prevent a computer from being overwhelmed with incoming data. Although the computer's central processor could perform the check, doing so in the host interface keeps traffic on the Ethernet from slowing down processing on all computers.

A 48-bit Ethernet address can do more than specify a single destination computer. An address can be one of three types:

- The physical address of one network interface (a *unicast address*)
- The network *broadcast address*
- A *multicast address*

By convention, the broadcast address (all 1s) is reserved for sending to all stations simultaneously. Multicast addresses provide a limited form of broadcast in which a subset of the computers on a network agree to listen to a given multicast address. The set of participating computers is called a *multicast group*. To join a multicast group, a computer must instruct its host interface to accept the group's multicast address. The advantage of multicasting lies in the ability to limit broadcasts: every computer in a multicast group can be reached with a single packet transmission, but computers that choose not to participate in a particular multicast group do not receive packets sent to the group.

---

†The Institute for Electrical and Electronic Engineers (IEEE) manages the Ethernet address space and assigns addresses as needed.

To accommodate broadcast and multicast addressing, Ethernet interface hardware must recognize more than its physical address. A computer interface usually accepts at least two kinds of packets: those addressed to the interface's physical (i.e., unicast) address and those addressed to the network broadcast address. Some interfaces can be programmed to recognize multicast addresses or even alternate physical addresses. When a computer boots, the operating system initializes the Ethernet interface hardware, giving it a set of addresses to recognize. The interface then examines the destination address field in each packet, passing on to the computer only those transmissions designated for one of the specified addresses.

2.4.10 Ethernet Frame Format

Ethernet should be thought of as a link-level connection among machines. Thus, it makes sense to view the data transmitted as a *frame*†. Ethernet frames are of variable length, with no frame smaller than 64 octets‡ or larger than 1518 octets (header, data, and CRC). As in all packet-switched networks, each Ethernet frame contains a field that contains the address of its destination. Figure 2.7 shows that the Ethernet frame format contains the physical source address as well as the destination address.

Preamble	Destination Address	Source Address	Frame Type	Frame Data	CRC
8 octets	6 octets	6 octets	2 octets	46–1500 octets	4 octets

Figure 2.7 The format of a frame (packet) as it travels across an Ethernet preceded by a preamble. Fields are not drawn to scale.

In addition to identifying the source and destination, each frame transmitted across the Ethernet contains a *preamble*, *type field*, *data field*, and *Cyclic Redundancy Check* (*CRC*). The preamble consists of 64 bits of alternating *0*s and *1*s to help receiving interfaces synchronize. The 32-bit CRC helps the interface detect transmission errors: the sender computes the CRC as a function of the data in the frame, and the receiver recomputes the CRC to verify that the packet has been received intact.

The frame type field contains a 16-bit integer that identifies the type of the data being carried in the frame. From the Internet point of view, the frame type field is essential because it means Ethernet frames are *self-identifying*. When a frame arrives at a given machine, the operating system uses the frame type to determine which protocol software module should process the frame. The chief advantages of self-identifying frames are that they allow multiple protocols to be used together on a single computer and they allow multiple protocols to be intermixed on the same physical network without interference. For example, one could have an application program on a computer using Internet protocols while another application on the same computer uses a local experimental protocol. The operating system examines the type field of each arriv-

†The term *frame* derives from communication over serial lines in which the sender "frames" the data by adding special characters before and after the transmitted data.

‡Technically, the term *byte* refers to a hardware-dependent character size; networking professionals use the term *octet*, because it refers to an 8-bit quantity on all computers.

ing frame to decide how to process the contents. We will see that the TCP/IP protocols use self-identifying Ethernet frames to distinguish among several protocols.

2.4.11 Extending An Ethernet With Repeaters

Although the original Ethernet cable had a maximum length, a network could be extended in two ways: using repeaters and bridges. An electronic device called a *repeater* operates on analog electrical signals. Like a hub in a twisted pair Ethernet, a repeater relays all electrical signals from one cable to another. Specifically, in the original thick Ethernet wiring scheme, a repeater can be placed between a pair of coaxial cables to double the total length. However, to preserve the CSMA/CD timing, the Ethernet standard restricts the use of repeaters — at most two repeaters can be placed between any two machines. Figure 2.8 shows a typical use of repeaters in an office building. A single cable runs vertically up the building, and a repeater attaches the backbone to an additional cable on each floor. Computers attach to the cables on each floor.

2.4.12 Extending An Ethernet With Bridges

Connecting two Ethernets with a bridge is superior to connecting them with a repeater or hub because bridges operate on packets rather than electrical signals. In particular, a bridge does not replicate noise, errors, or malformed frames; the bridge must receive a completely valid frame from one segment before the bridge will accept and transmit it on the other segment. Furthermore, each connection between a bridge and an Ethernet network follows the CSMA/CD rules, so collisions and propagation delays on one segment remain isolated from those on the other. As a result, an (almost) arbitrary number of Ethernets can be connected together with bridges. The important point is:

> *Bridges hide the details of interconnection: a set of bridged segments acts like a single Ethernet.*

Bridged networks are classified as *transparent* because a computer does not know how many bridges connect segments of the network. The computer uses exactly the same hardware, frame format, and procedures to communicate with a computer across a bridge as it uses to communicate with a computer on the local segment.

Most bridges do much more than replicate frames from one wire to another: they make intelligent decisions about which frames to forward. Such bridges are called *adaptive* or *learning* bridges. An adaptive bridge consists of a computer with two Ethernet interfaces. The software in an adaptive bridge keeps two address lists, one for each interface. When a frame arrives from Ethernet E_1, the adaptive bridge adds the 48-bit Ethernet *source* address to the list associated with E_1. Similarly, when a frame

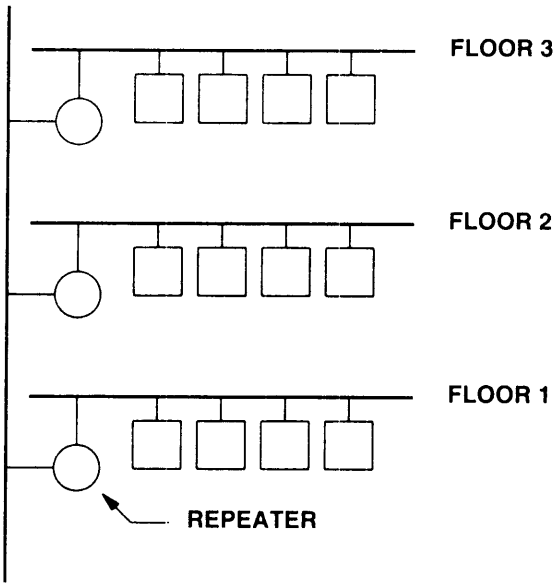

Figure 2.8 Repeaters used to join Ethernet cables in a building. At most two repeaters can be placed between a pair of communicating machines.

arrives from Ethernet E_2, the bridge adds the source address to the list associated with E_2. Thus, over time the adaptive bridge will learn which machines lie on E_1 and which lie on E_2.

After recording the source address of a frame, the adaptive bridge uses the destination address to determine whether to forward the frame. If the address list shows that the destination lies on the Ethernet from which the frame arrived, the bridge does not forward the frame. If the destination is not in the address list (i.e., the destination is a broadcast or multicast address or the bridge has not yet learned the location of the destination), the bridge forwards the frame to the other Ethernet.

The advantages of adaptive bridges should be obvious. Because the bridge uses addresses found in normal traffic, it is completely automatic — humans need not configure the bridge with specific addresses. Because it does not forward traffic unnecessarily, a bridge helps improve the performance of an overloaded network by isolating traffic on specific segments. Bridges work exceptionally well if a network can be divided physically into two segments that each contain a set of computers that communicate frequently (e.g., each segment contains a set of workstations along with a server, and the workstations direct most of their traffic to the server). To summarize:

An adaptive Ethernet bridge connects two Ethernet segments, forwarding frames from one to the other. It uses source addresses to learn which machines lie on which Ethernet segment, and it combines information learned with destination addresses to eliminate forwarding when unnecessary.

From the TCP/IP point of view, bridged Ethernets are merely another form of physical network connection. The important point is:

Because the connection among physical cables provided by bridges and repeaters is transparent to machines using the Ethernet, we think of multiple Ethernet segments connected by bridges and repeaters as a single physical network system.

Most commercial bridges are much more sophisticated and robust than our description indicates. When first powered up, they check for other bridges and learn the topology of the network. They use a distributed spanning-tree algorithm to decide how to forward frames. In particular, the bridges decide how to propagate broadcast packets so only one copy of a broadcast frame is delivered to each wire. Without such an algorithm, Ethernets and bridges connected in a cycle would produce catastrophic results because they would forward broadcast packets in both directions simultaneously.

2.5 Fiber Distributed Data Interconnect (FDDI)

FDDI is another popular local area networking technology that provides a data rate of 100 Mbps (i.e., the same data rate as Fast Ethernet). Unlike Ethernet and other LAN technologies that use copper cables to carry electrical signals, FDDI is designed to use optical fiber. Data is encoded in pulses of light†.

Optical fiber has two advantages over copper wire. First, because electrical noise does not interfere with an optical connection, the fiber can lie adjacent to powerful electrical devices. Second, because optical fibers use light, the amount of data that can be sent per unit time is much higher than cables that carry electrical signals.

It might seem that glass fibers would be difficult to install and would break if bent. However, an optical cable is surprisingly flexible. The glass fiber itself has an extremely small diameter, and the cable includes a plastic jacket that protects the fiber from breaking. Such a cable cannot bend at a ninety degree angle, but it can bend in an arc with a diameter of a few inches. Thus, installation is not difficult.

†A related technology known as *Copper Distributed Data Interface (CDDI)* works like FDDI, but uses copper cables to carry signals.

2.5.1 Properties Of An FDDI Network

An FDDI network is a 100 Mbps shared token passing ring technology with a self-healing capability. An FDDI network is *shared* because multiple computers connect to a given network and take turns sending packets. FDDI is known as a *ring* because the network forms a cycle that starts at one computer, passes through all others computers, and ends back at the source. FDDI is a *token passing ring* (or simply a *token ring*) technology because it uses token passing to control transmission. When the network is idle, a special, reserved frame called a *token* circulates around the ring from station to station. When a station has a packet to send, it waits for the token to arrive, sends its packet, and then passes the token to the next station. The circulating token guarantees fairness: it ensures that all stations have an opportunity to send a packet before any station sends a second packet.

Perhaps the most interesting property of an FDDI lies in its ability to detect and correct problems. The network is called *self-healing* because the hardware can automatically accommodate a failure.

2.5.2 Dual Counter-Rotating Rings

To provide automatic recovery from failures, FDDI hardware uses two independent rings that both connect to each computer. Figure 2.9 illustrates the topology.

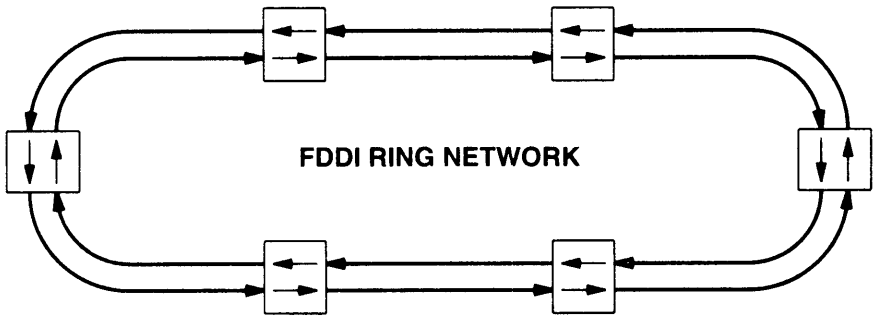

Figure 2.9 An FDDI network with optical fibers interconnecting six computers. Arrows show the direction of traffic on the fibers and through the attached computers.

FDDI rings are called *counter rotating* because traffic passes in the opposite direction on each ring. The reason for using a counter rotating scheme will become clear when we consider how FDDI handles failures.

Unless an error has occurred, an FDDI hardware does not need both rings. In fact, an FDDI interface behaves like any token passing network interface until an error occurs. The interface examines all packets that circulate around the ring, comparing the

destination address in each packet to the computer's address. The interface keeps a copy of any packet destined for the local computer, but also forwards the packet around the ring.

When a computer needs to transmit a packet, it waits for the token to arrive, temporarily stops forwarding bits, and sends its packet. After sending one packet, the interface transmits the token, and begins forwarding bits again. Even if a station has more than one packet ready to be sent when it receives the token, the station only sends one packet before passing the token.

FDDI hardware becomes more interesting when a hardware error occurs. When an interface detects that it cannot communicate with the adjacent computer, the interface uses the backup ring to bypass the failure. For example, Figure 2.10 shows an FDDI ring in which an interface has failed, and the two adjacent interfaces have eliminated it from the ring.

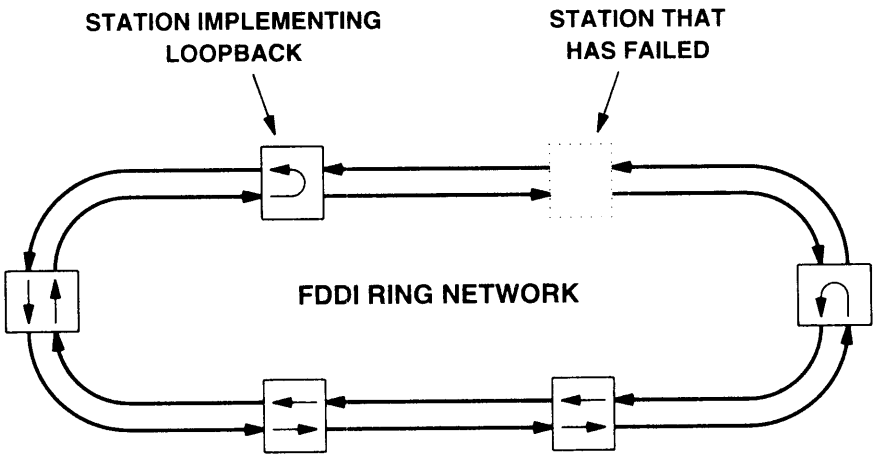

Figure 2.10 An FDDI ring after a failure. When FDDI hardware detects such a failure, it uses the second ring to bypass the failure and allows remaining stations to communicate.

The purpose of the second ring and the reason data flows in the opposite direction should now be clear: a failure can mean that the fiber has been disconnected (e.g., accidentally cut). If the fiber from both rings follows the same physical path, chances are high that the second fiber may have been disconnected as well. FDDI hardware automatically uses the counter rotating ring to form a closed loop in the direction that is still working. Doing so permits the other computers to continue communication despite the failure.

When FDDI hardware detects a failure on the network, it automatically loops data across the backup ring to permit communication among remaining stations.

2.5.3 FDDI Frame Format

FDDI standards specify the exact format of frames used on the network. The table in Figure 2.11 lists fields in an FDDI frame.

Field	Length in 4-bit units	Contents
PA	4 or more	Preamble
SD	2	Start Delimiter
FC	2	Frame Control
DA	4 or 12	Destination Address
SA	4 or 12	Source Address
RI	0 to 60	Routing Information
DATA	0 or more	Data
FCS	8	Frame Check Sequence
ED	1	End Delimiter
FS	3 or more	Frame Status

Figure 2.11 The format of frames used by FDDI, with fields measured in 4-bit units called *symbols*. The maximum frame length is 9000 symbols.

Like other technologies, each computer attached to an FDDI network is assigned an address, and each frame contains a destination address field. However, to make FDDI more flexible and to provide a standard way to interconnect two FDDI rings, the designers allowed more than one frame format. For example, the destination address field is either 4 or 12 symbols long, where a *symbol* is a 4-bit unit. The frame also includes a field used for routing. The sender can use the routing field to specify that a frame must be sent first to a connection point and then on to a destination on an attached ring.

One of the advantages of FDDI arises from its large frame size. Because a frame can contain 9000 4-bit symbols, the total frame can be 4500 octets long. Because header information occupies at most a few hundred octets, a single frame can carry 4K octets of user data. For applications that transfer large volumes of data (e.g., file transfer), the large frame size means less overhead and consequently high throughput.

2.6 Asynchronous Transfer Mode

Asynchronous Transfer Mode (*ATM*) is the name given to a connection-oriented networking technology that is intended for use in both local area and wide area networks. ATM is designed to permit extremely high speed data switching; the fastest ATM hardware can switch data at gigabit speeds†. Of course, such high speeds require complex, state-of-the-art hardware. As a result, ATM networks are more expensive than other technologies.

To achieve high transfer speeds, an ATM network uses special-purpose hardware and software techniques. First, an ATM network consists of one or more high-speed switches that each connect to computers and to other ATM switches. Second, ATM uses optical fibers for connections, including connections from a user's computer to an ATM switch. Optical fibers provide a higher transfer rate than copper wires; typically, the connection between a user's computer and an ATM switch operates at 155 Mbps. Third, the lowest layers of an ATM network use fixed-size frames called *cells*. Because each cell is exactly the same size, ATM switch hardware can process cells quickly.

2.6.1 ATM Cell Size

Surprisingly, each ATM cell is only 53 octets long. The cell contains 5 octets of header followed by 48 octets of data. Later chapters will show, however, that when using ATM to send IP traffic, the 53 octet size is irrelevant — an ATM network accepts and delivers much larger packets.

2.6.2 Connection-Oriented Networking

ATM differs from the packet-switching networks described earlier because it offers *connection-oriented* service. Before a computer connected to an ATM switch can send cells, a connection must be established manually or the host must first interact with the switch to specify a destination. The interaction is analogous to placing a telephone call‡. The requesting computer specifies the remote computer's address, and waits for the ATM switch to find a path through the network and establish a connection. If the remote computer rejects the request, does not respond, or the ATM switches between the sender and receiver cannot currently establish a path, the request to establish communication fails.

Once a connection succeeds, the local ATM switch chooses an identifier for the connection, and passes the connection identifier to the computer along with a message that informs the computer of success. The computer uses the connection identifier when sending or receiving cells.

When it finishes using a connection, the computer again communicates with the ATM switch to request that the connection be broken. The switch then disconnects the two computers. Disconnection is equivalent to hanging up a telephone at the end of a telephone call; after a disconnection, the computers cannot communicate until they es-

†Most computers cannot generate or absorb data at gigabit rates; ATM networks operate at gigabit speed to handle the traffic from many computers.

‡Because ATM was designed to carry voice as well as data, there is a strong relationship between an ATM network and a telephone system.

tablish a new connection. Furthermore, identifiers used for a connection can be recycled; once a disconnection occurs, the switch can reuse the connection identifier for a new connection.

2.7 WAN Technologies: ARPANET

We will see that wide area networks have important consequences for internet addressing and routing. The technologies discussed in the remainder of this chapter were selected because they figure prominently in both the history of the Internet and later examples in the text.

One of the oldest wide area technologies, the ARPANET, was funded by *ARPA*, the *Advanced Research Projects Agency*. ARPA awarded a contract for the development of the ARPANET to Bolt, Beranek and Newman of Cambridge, MA in the fall of 1968. By September 1969, the first pieces of the ARPANET were in place.

The ARPANET served as a testbed for much of the research in packet-switching. In addition to its use for network research, researchers in several universities, military bases, and government labs regularly used the ARPANET to exchange files and electronic mail and to provide remote login among their sites. In 1975, control of the network was transferred from ARPA to the U.S. Defense Communications Agency (DCA). The DCA made the ARPANET part of the Defense Data Network (DDN), a program that provides multiple networks as part of a world-wide communication system for the Department of Defense.

In 1983, the Department of Defense partitioned the ARPANET into two connected networks, leaving the ARPANET for experimental research and forming the *MILNET* for military use. MILNET was restricted to unclassified data because it was not considered secure. Although under normal circumstances, both ARPANET and MILNET agreed to pass traffic to each other, controls were established that allowed them to be disconnected†. Because the ARPANET and MILNET used the same hardware technology, our description of the technical details apply to both. In fact, the technology was also available commercially and was used by several corporations to establish private packet switching networks.

Because the ARPANET was already in place and used daily by many of the researchers who developed the Internet architecture, it had a profound effect on their work. They came to think of the ARPANET as a dependable wide area backbone around which the Internet could be built. The influence of a single, central wide area backbone is still painfully obvious in some of the Internet protocols that we will discuss later, and has prevented the Internet from accommodating additional backbone networks gracefully.

Physically, the ARPANET consisted of approximately 50 BBN Corporation C30 and C300 minicomputers, called *Packet Switching Nodes* or *PSNs*‡ scattered across the continental U.S. and western Europe; MILNET contained approximately 160 PSNs, including 34 in Europe and 18 in the Pacific and Far East. One PSN resided at each site participating in the network and was dedicated to the task of switching packets; it could

†Perhaps the best known example of disconnection occurred in November 1988 when a *worm* program attacked the Internet and replicated itself as quickly as possible.

‡PSNs were initially called *Interface Message Processors* or *IMPs*; some publications still use the term IMP as a synonym for packet switch.

not be used for general-purpose computation. Indeed, each PSN was considered to be part of the ARPANET, and was owned and controlled by the *Network Operations Center* (*NOC*) located at BBN in Cambridge, Massachusetts.

Point-to-point data circuits leased from common carriers connected the PSNs together to form a network. For example, leased data circuits connected the ARPANET PSN at Purdue University to the ARPANET PSNs at Carnegie Mellon and at the University of Wisconsin. Initially, most of the leased data circuits in the ARPANET operated at 56 Kbps, a speed considered fast in 1968 but extremely slow by current standards. Remember to think of the network speed as a measure of capacity rather than a measure of the time it takes to deliver packets. As more computers used the ARPANET, capacity was increased to accommodate the load. For example, during the final year the ARPANET existed, many of the cross-country links operated over megabit-speed channels.

The idea of having no single point of failure in a system is common in military applications because reliability is important. When building the ARPANET, ARPA decided to follow the military requirements for reliability, so they mandated that each PSN had to have at least two leased line connections to other PSNs, and the software had to automatically adapt to failures and choose alternate routes. As a result, the ARPANET continued to operate even if one of its data circuits failed.

In addition to connections for leased data circuits, each ARPANET PSN had up to 22 *ports* that connected it to user computers, called *hosts*. Originally, each computer that accessed the ARPANET connected directly to one of the ports on a PSN. Normally, host connections were formed with a special-purpose interface board that plugged into the computer's I/O bus.

The original PSN port hardware used a complex protocol for transferring data across the ARPANET. Known as 1822, after the number of a technical report that described it, the protocol permitted a host to send a packet across the ARPANET to a specified destination PSN and a specified port on that PSN. Performing the transfer was complicated, however, because 1822 offered reliable, flow-controlled delivery. To prevent a given host from saturating the net, 1822 limited the number of packets that could be in transit. To guarantee that each packet arrived at its destination, 1822 forced the sender to await a *Ready For Next Message* (*RFNM*) signal from the PSN before transmitting each packet. The RFNM acted as an acknowledgement. It included a buffer reservation scheme that required the sender to reserve a buffer at the destination PSN before sending a packet.

Although there are many aspects not discussed here, the key idea is that underneath all the detail, the ARPANET was merely a transfer mechanism. When a computer connected to one port sent a packet to another port, the data delivered was exactly the data sent. Because the ARPANET did not provide a network-specific frame header, packets sent across it did not have a fixed field to specify packet type. Thus, unlike some network technologies, the ARPANET did not deliver self-identifying packets. In summary:

Networks such as the ARPANET or an ATM network do not have self-identifying frames. The attached computers must agree on the format and contents of packets sent or received to a specific destination.

Unfortunately, 1822 was never an industry standard. Because few vendors manufactured 1822 interface boards, it became difficult to connect new machines to the ARPANET. To solve the problem, ARPA later revised the PSN interface to use the *X.25* standard†. The first version of an X.25 PSN implementation used only the data transfer part of the X.25 standard (known as HDLC/LAPB), but later versions made it possible to use all of X.25 when connecting to a PSN (i.e., ARPANET appeared to be an X.25 network).

Internally, of course, the ARPANET used its own set of protocols that were invisible to users. For example, there was a special protocol that allowed one PSN to request status from another, a protocol that PSNs used to send packets among themselves, and one that allowed PSNs to exchange information about link status and optimal routes.

Because the ARPANET was originally built as a single, independent network to be used for research, its protocols and addressing structure were designed without much thought given to expansion. By the mid 1970's, it became apparent no single network would solve all communication problems, and ARPA began to investigate satellite and packet radio network technologies. This experience with a variety of network technologies led to the concept of an internetwork.

2.7.1 ARPANET Addressing

While the details of ARPANET addressing are unimportant, they illustrate an alternative way in which wide area networks form physical addresses. Unlike the *flat address* schemes used by LAN technologies, wide area networks usually embed information in the address that helps the network route packets to their destination efficiently. In the ARPANET technology, each packet switch is assigned a unique integer, *P*, and each host port on the switch is numbered from *0* to *N-1*. Conceptually, a destination address consists of a pair of small integers, *(P, N)*. In practice, the hardware uses a single, large integer address, with some bits of the address used to represent *N* and others used to represent *P*.

2.8 National Science Foundation Networking

Realizing that data communication would soon be crucial to scientific research, in 1987 the National Science Foundation established a *Division of Network and Communications Research and Infrastructure* to help ensure that requisite network communications will be available for U.S. scientists and engineers. Although the division funds basic research in networking, its emphasis so far has been concentrated on providing seed funds to build extensions to the Internet.

†X.25 was standardized by the *Consultative Committee on International Telephone and Telegraph (CCITT)*, which later became the *Telecommunication Section* of the *International Telecommunication Union (ITU)*.

NSF's Internet extensions introduced a three-level hierarchy consisting of a U.S. backbone, a set of "mid-level" or "regional" networks that each span a small geographic area, and a set of "campus" or "access" networks. In the NSF model, mid-level networks attach to the backbone and campus networks attach to the mid-level nets. Each researcher had a connection from their computer to the local campus network. They used that single connection to communicate with local researchers' computers across the local campus net, and with other researchers further away. The campus network routed traffic across local nets to one of the mid-level networks, which routed it across the backbone as needed.

2.8.1 The Original NSFNET Backbone

Of all the NSF-funded networks, the NSFNET backbone has the most interesting history and used the most interesting technology. The backbone evolved in four major steps; it increased in size and capacity at the time the ARPANET declined until it became the dominant backbone in the Internet. The first version was built quickly, as a temporary measure. One early justification for the backbone was to provide scientists with access to NSF supercomputers. As a result, the first backbone consisted of six Digital Equipment Corporation LSI-11 microcomputers located at the existing NSF supercomputer centers. Geographically, the backbone spanned the continental United States from Princeton, NJ to San Diego, CA, using 56 Kbps leased lines as Figure 2.12 shows.

At each site, the LSI-11 microcomputer ran software affectionately known as *fuzzball*† code. Developed by Dave Mills, each fuzzball accessed computers at the local supercomputer center using a conventional Ethernet interface; it accessed leased lines leading to fuzzballs at other supercomputer centers using conventional link-level protocols over leased serial lines. Fuzzballs contained tables with addresses of possible destinations and used those tables to direct each incoming packet toward its destination.

The primary connection between the original NSFNET backbone and the rest of the Internet was located at Carnegie Mellon, which had both an NSFNET backbone node and an ARPANET PSN. When a user, connected to NSFNET, sent traffic to a site on the ARPANET, the packets would travel across the NSFNET to CMU where the fuzzball would route them onto the ARPANET via a local Ethernet. Similarly, the fuzzball understood that packets destined for NSFNET sites should be accepted from the Ethernet and sent across the NSF backbone to the appropriate site.

†The exact origin of the term "fuzzball" is unclear.

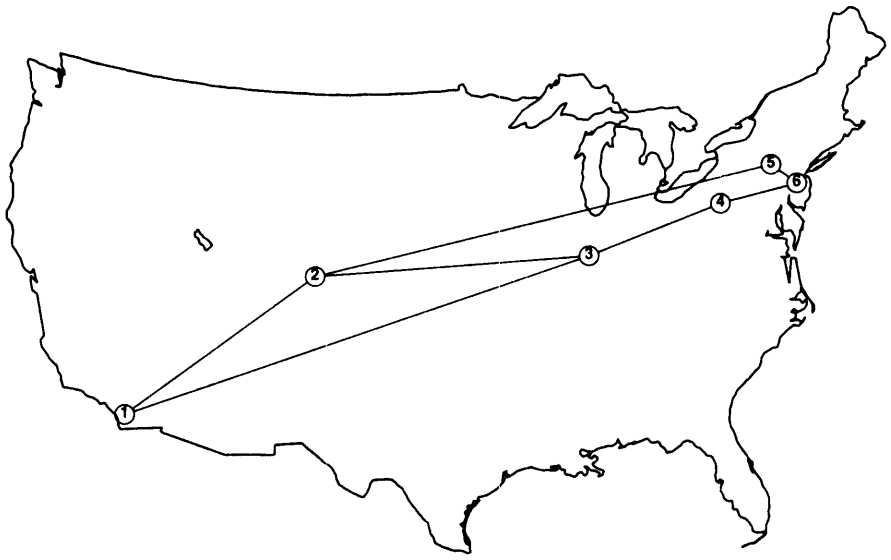

Figure 2.12 Circuits in the original NSFNET backbone with sites in (1) San Diego, CA; (2) Boulder, CO; (3) Champaign, IL; (4) Pittsburgh, PA; (5) Ithaca, NY; and (6) Princeton, NJ.

2.8.2 The Second NSFNET Backbone 1988-1989

Although users were excited about the possibilities of computer communication, the transmission and switching capacities of the original backbone were too small to provide adequate service. Within months after its inception, the backbone became overloaded and its inventor worked to engineer quick solutions for the most pressing problems, while NSF began the arduous process of planning for a second backbone.

In 1987, NSF issued a request for proposals from groups that were interested in establishing and operating a new, higher-speed backbone. Proposals were submitted in August of 1987 and evaluated that fall. On November 24, 1987 NSF announced it had selected a proposal submitted by a partnership of: MERIT Inc., the statewide computer network run out of the University of Michigan in Ann Arbor; IBM Corporation; and MCI Incorporated. The partners proposed to build a second backbone network, establish a network operation and control center in Ann Arbor, and have the system operational by the following summer. Because NSF had funded the creation of several new mid-level networks, the proposed backbone was designed to serve more sites than the original. Each additional site would provide a connection between the backbone and one of the NSF mid-level networks.

The easiest way to envision the division of labor among the three groups is to assume that MERIT was in charge of planning, establishing, and operating the network center. IBM contributed machines and manpower from its research labs to help MERIT develop, configure, and test needed hardware and software. MCI, a long-distance carrier, provided the communication bandwidth using the optical fiber already in place for its voice network. Of course, in practice there was close cooperation between all groups, including joint study projects and representatives from IBM and MCI in the project management.

By the middle of the summer of 1988, the hardware was in place and NSFNET began to use the second backbone. Shortly thereafter, the original backbone was shut down and disconnected. Figure 2.13 shows the logical topology of the second backbone after it was installed in 1988.

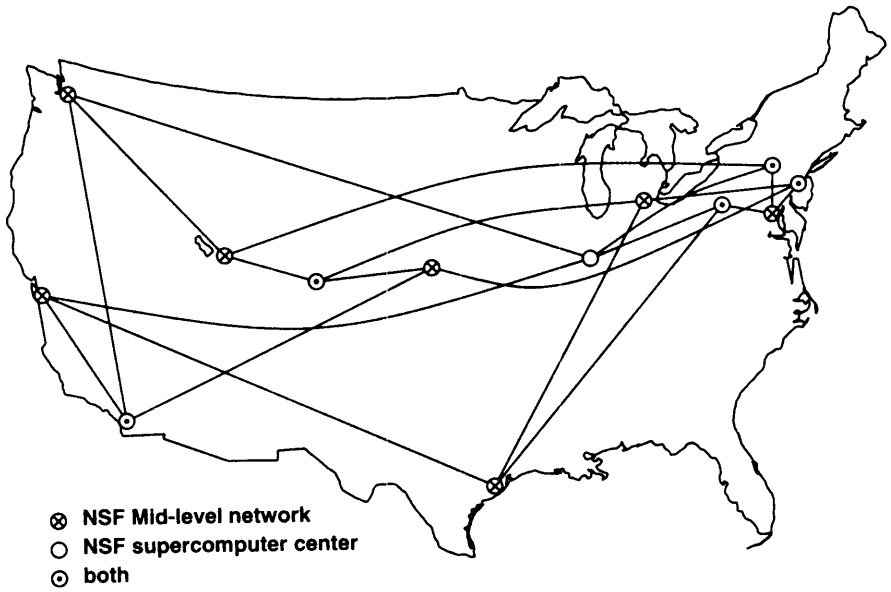

Figure 2.13 Logical circuits in the second NSFNET backbone from summer 1988 to summer 1989.

The technology chosen for the second NSFNET backbone was interesting. In essence, the backbone was a wide area network composed of packet routers interconnected by communication lines. As with the original backbone, the packet switch at each site connected to the site's local Ethernet as well as to communication lines leading to other sites.

2.8.3 NSFNET Backbone 1989-1990

After measuring traffic on the second NSFNET backbone for a year, the operations center reconfigured the network by adding some circuits and deleting others. In addition, they increased the speed of circuits to DS-1 (1.544 Mbps). Figure 2.14 shows the revised connection topology, which provided redundant connections to all sites.

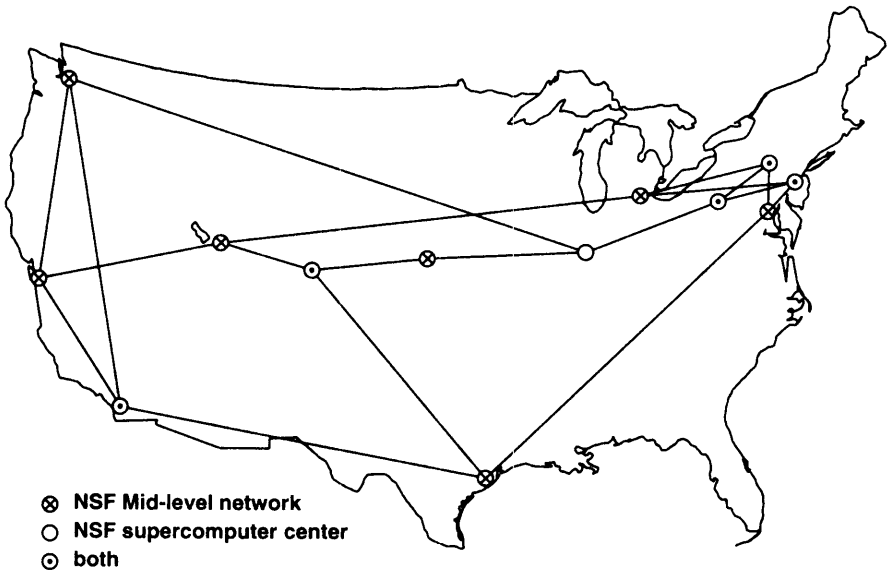

Figure 2.14 Circuits in the second NSFNET backbone from summer 1989 to 1990.

2.9 ANSNET

By 1991, NSF and other U.S. government agencies began to realize that the Internet was growing beyond its original academic and scientific domain. Companies around the world began to connect to the Internet, and nonresearch uses increased rapidly. Traffic on NSFNET had grown to almost one billion packets per day, and the 1.5 Mbps capacity was becoming insufficient for several of the circuits. A higher capacity backbone was needed. As a result, the U.S. government began a policy of commercialization and privatization. NSF decided to move the backbone to a private company and to charge institutions for connections.

Responding to the new government policy in December of 1991, IBM, MERIT, and MCI formed a not-for-profit company named *Advanced Networks and Services* (*ANS*). ANS proposed to build a new, higher speed Internet backbone. Unlike previous

wide area networks used in the Internet which had all been owned by the U.S. government, ANS would own the new backbone. By 1993, ANS had installed a new network that replaced NSFNET. Called *ANSNET*, the backbone consisted of data circuits operating at 45 Mbps†, giving it approximately 30 times more capacity than the previous NSFNET backbone. Figure 2.15 shows major circuits in ANSNET and a few of the sites connected in 1994. Each point of presence represents a location to which many sites connect.

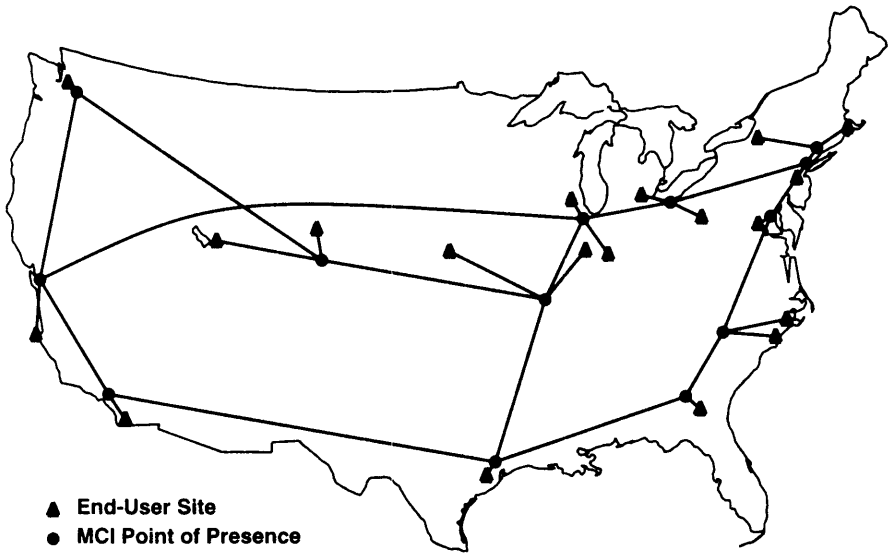

Figure 2.15 Circuits in ANSNET, the backbone of the U.S. Internet starting in 1993. Each circuit operates at 45 Mbps.

2.10 A Very High Speed Backbone (vBNS)

In 1995, NSF awarded MCI a contract to build a backbone operating at 155 Mbps (OC3 speed) to replace ANSNET. Called the *very high speed Backbone Network Service (vBNS)*, the new backbone offered a substantial increase in capacity, and required higher speed processors to route packets.

2.10.1 Commercial Internet Backbones

Since 1995, the Internet has become increasingly commercial, with the percentage of funding from the U.S. government steadily decreasing. Although vBNS still exists, it is now devoted to networking research. In its place, commercial companies have created large privately-funded backbones that carry Internet traffic. For example, public car-

†Telecommunication carriers use the term *DS3* to denote a circuit that operates at 45 Mbps; the term is often confused with *T3*, which denotes a specific encoding used over a circuit operating at DS3 speed.

riers like *AT&T* and *MCI* have each created large, high-capacity backbone networks used to carry Internet traffic from their customers. As discussed later, commercial backbones are interconnected through *peering arrangements*, making it possible for a customer of one company to send packets to a customer of another.

2.11 Other Technologies Over Which TCP/IP Has Been Used

One of the major strengths of TCP/IP lies in the variety of physical networking technologies over which it can be used. We have already discussed several widely used technologies, including local area and wide area networks. This section briefly reviews others that help illustrate an important principle:

> *Much of the success of the TCP/IP protocols lies in their ability to accommodate almost any underlying communication technology.*

2.11.1 X25NET And Tunnels

In 1980, NSF formed the *Computer Science NETwork* (*CSNET*) organization to help provide Internet services to industry and small schools. CSNET used several technologies to connect its subscribers to the Internet, including one called *X25NET*. Originally developed at Purdue University, X25NET ran TCP/IP protocols over *Public Data Networks* (*PDN*s). The motivation for building such a network arose from the economics of telecommunications: although leased serial lines were expensive, common carriers had begun to offer public packet-switched services. X25NET was designed to allow a site to use its connection to a public packet-switched service to send and receive Internet traffic.

Readers who know about public packet-switched networks may find X25NET strange because public services use the CCITT X.25 protocols exclusively while the Internet uses TCP/IP protocols. Unlike most packet switching hardware, X.25 protocols use a connection-oriented paradigm; like ATM, they were designed to provide connection-oriented service to individual applications. Thus, the use of X.25 to transport TCP/IP traffic foreshadowed the ways TCP/IP would later be transferred across ATM.

We have already stated that many underlying technologies can be used to carry Internet traffic, and X25NET illustrates how TCP/IP has been adapted to use high level facilities. The technique, sometimes called *tunneling*, simply means that TCP/IP treats a complex network system with its own protocols like any other hardware delivery system. To send TCP/IP traffic through an X.25 *tunnel*, a computer forms an X.25 connection and then sends TCP/IP packets as if they were data. The X.25 system carries the packets along its connection and delivers them to another X.25 endpoint, where they must be picked up and forwarded on to their ultimate destination. Because tunneling treats IP packets like data, the tunnel does not provide for self-identifying frames.

Thus, tunneling only works when both ends of the X.25 connection agree *a priori* that they will exchange IP packets (or agree on a format for encoding type information along with each packet).

Its connection-oriented interface makes X.25 even more unusual. Unlike connectionless networks, connection-oriented systems use a *virtual circuit* (*VC*) abstraction. Before data can be sent, switches in the network must set up a VC (i.e., a "path") between the sender and the receiver. We said that the Internet protocols were optimized to run over a connectionless packet delivery system, which means that extra effort is required to run them over a connection-oriented network.

In theory, a single connection suffices for a tunnel through a connection-oriented network — after a pair of computers has established a VC, that pair can exchange TCP/IP traffic. In practice, however, the design of the protocols used on the connection-oriented system can make a single connection inefficient. For example, because X.25 protocols limit the number of packets that can be sent on a connection before an acknowledgement is received, such networks exhibit substantially better throughput when data is sent across multiple connections simultaneously. Thus, instead of opening a single connection to a given destination, X25NET improved performance by arranging for a sender to open multiple VCs and distribute traffic among them. A receiver must accept packets arriving on all connections, and combine them together again.

Tunneling across a high-level network such as X.25 requires mapping between the addresses used by the internet and addresses used by the network. For example, consider the addressing scheme used by X.25 networks, which is given in a related standard known as *X.121*. Physical addresses each consist of a 14-digit number, with 10 digits assigned by the vendor that supplies the X.25 network service. Resembling telephone numbers, one popular vendor's assignment includes an area code based on geographic location. The addressing scheme is not surprising because it comes from an organization that determines international telephone standards. There is no mathematical relationship between such addresses and the addresses used by TCP/IP. Thus, a computer that tunnels TCP/IP data across an X.25 network must maintain a table of mappings between internet addresses and X.25 network addresses. Chapter 5 discusses the address mapping problem in detail and gives an alternative to using fixed tables. Chapter 18 shows that exactly the same problem arises for ATM networks, which use yet another alternative.

Because public X.25 networks operated independently of the Internet, a point of contact was needed between the two. Both ARPA and CSNET operated dedicated machines that provided the interconnection between X.25 and the ARPANET. The primary interconnection was known as the *VAN gateway*. The VAN agreed to accept X.25 connections and route each datagram that arrived over such a connection to its destination.

X25NET was significant because it illustrated the flexibility and adaptability of the TCP/IP protocols. In particular, it showed that tunneling makes it possible to use an extremely wide range of complex network technologies in an internet.

2.11.2 Point-To-Point Networks

We said that Wide Area Networks are usually composed of dedicated packet switches interconnected by data circuits leased from a telephone company. Phone companies originally designed such circuits to carry digitized voice calls; only later did their use in data networks become important. Consequently, the data rates of available circuits are not powers of ten. Instead, they have been chosen to carry multiples of 64 Kbps because a digitized voice call uses an encoding known as *Pulse Code Modulation* (*PCM*), which produces 8000 samples per second, where each sample is 8 bits.

The table in Figure 2.16 lists a few common data rates used in North America and Europe.

Name	Bit Rate	Voice Circuits	Location
–	0.064 Mbps	1	
T1	1.544 Mbps	24	North America
T2	6.312 Mbps	96	North America
T3	44.736 Mbps	672	North America
E1	2.048 Mbps	30	Europe
E2	8.448 Mbps	120	Europe
E3	34.368 Mbps	480	Europe

Figure 2.16 Example data rates available on digital circuits leased from a telephone company. The rates were chosen to encode multiple voice calls.

Higher rate digital circuits are also available. In addition to standards that specify the transmission of high data rates over copper, the phone companies have developed standards for transmission of the same rates over optical fiber. The table in Figure 2.17 contains examples. Of course, circuits that operate at such high data rates are considerably more expensive than circuits that operate at lower rates.

Standard Name	Optical Name	Bit Rate	Voice Circuits
STS-1	OC-1	51.840 Mbps	810
STS-3	OC-3	155.520 Mbps	2430
STS-12	OC-12	622.080 Mbps	9720
STS-24	OC-24	1,244.160 Mbps	19440
STS-48	OC-48	2,488.320 Mbps	38880

Figure 2.17 Example data rates of high-capacity circuits that can be leased from phone companies. Optical fiber is used to achieve such high rates over long distances.

From TCP/IP's point of view, any communication system that connects exactly two computers is known as a *point-to-point network*. Thus, a leased data circuit between two computers is an example of a point-to-point network. Of course, using the term "network" to describe a connection between two computers stretches the concept. However, we will learn that viewing a connection as a network helps maintain consistency. For now, we only need to note that a point-to-point network differs from conventional networks in one significant way, because only two computers attach, no hardware addresses are used. When we discuss internet address binding, the lack of hardware addresses will make point-to-point networks an exception.

2.11.3 Dial-up IP

Another interesting use of TCP/IP pioneered by CSNET involves running TCP/IP protocols over the dial-up voice network (i.e., the telephone system). CSNET member sites that used the Internet infrequently could not justify the cost of a leased line connection. For such sites, CSNET developed a dial-up IP system that worked as expected: whenever a connection was needed, software at the member's site used a modem to form a connection to the CSNET hub over the voice telephone network. A computer at the hub answered the phone call and, after obtaining valid authorization, began to forward traffic between the site and other computers on the Internet. Dialing introduced a delay after the first packet was sent. However, for automated services like electronic mail, the delay was unnoticeable.

Dialup internet access provides another example of a point-to-point network. From the TCP/IP view, dialing a telephone call is equivalent to running a wire. Once the call has been answered by a modem on the other end, there is a connection from one computer directly to another, and the connection stays in place as long as needed.

2.11.4 Other Token Ring Technologies

FDDI is not the first token ring network technology; token ring products have existed for nearly twenty years. For example, IBM produces a popular token ring LAN technology. Early versions of the IBM token ring operated at 4 Mbps; later versions operate at 16 Mbps. Like other token ring systems, an IBM token ring network consists of a loop that attaches to all computers. A station must wait for a token before transmitting, and sends the token along after transferring a packet.

An older token ring technology designed by Proteon Corporation employs a novel hardware addressing scheme that will be used in a later chapter to illustrate one of the ways TCP/IP uses hardware addresses. Called a *proNET* network, the technology permits customers to choose a hardware address for each computer. Unlike an Ethernet, in which each interface board contains a unique address assigned by the manufacturer, a proNET interface board contains eight switches that must be set before the interface is installed in a computer. The switches form a number in binary between 0 and 255, inclusive. A given proNET network could have at most 254 computers attached because address 255 was reserved for broadcast and address 0 was not used. When first instal-

ling a proNET network, a network administrator chose a unique address for each computer. Typically, addresses were assigned sequentially, starting with *1*.

A technology that permits customers to assign hardware addresses has advantages and disadvantages. The chief disadvantage arises from the potential for problems that occur if a network administrator accidentally assigns the same address to two computers. The chief advantage arises from ease of maintenance: if an interface board fails, it can be replaced without changing the computer's hardware address.

2.11.5 Wireless Network Technologies

One of the most interesting ARPA experiments in packet switching resulted in a *packet radio* technology that uses broadcast radio waves to carry packets. Designed for a military environment in which stations might be mobile, packet radio includes hardware and software that allow sites to find other sites, establish point-to-point communication, and then use the point-to-point communication to carry packets. Because sites change geographic location and may move out of communication range, the system must constantly monitor connectivity and recompute routes to reflect changes in topology. An operational packet radio system was built and used to demonstrate TCP/IP communication between a remote packet radio site and other sites on the Internet.

In recent years, a wide variety of wireless networking equipment has become available commercially. Wireless LANs use spread spectrum techniques such as direct sequencing or frequency hopping to provide data connections among a set of computers inside a building. The transmitters and antennas for such equipment are small and lightweight. The equipment can be attached to a portable notebook computer, making it convenient to move around an area such as an office building while remaining in communication.

Wireless broadband technology, originally developed as an alternative to cable television, is being used to transmit data. Known as *Multichannel Multipoint Distribution System (MMDS)*, the scheme has sufficient capacity to provide data rates as fast as those offered by the popular *Digital Subscriber Line (DSL)* technologies that deliver high data rates over copper telephone wires.

Cellular technology, which was originally designed for voice networks, has also been adapted to carry data. The chief advantage of a cellular system is the speed with which it allows users to move. Because the technology was designed to maintain voice communication even if a user travels by car, the underlying hardware can easily maintain contact with a mobile unit while transferring a stream of packets.

2.12 Summary And Conclusion

We have reviewed several network hardware technologies used by the TCP/IP protocols, ranging from inexpensive Local Area Network technologies like Ethernet and FDDI to expensive Wide Area Network technologies that use leased digital circuits to provide backbones. We have also seen that it is possible to run the TCP/IP protocols

over other general-purpose network protocols using a technique called tunneling. While the details of specific network technologies are not important, a general idea has emerged:

> *The TCP/IP protocols are extremely flexible; almost any underlying technology can be used to transfer TCP/IP traffic.*

FOR FURTHER STUDY

Early computer communication systems employed point-to-point interconnection, often using general-purpose serial line hardware that McNamara [1982] describes. Metcalf and Boggs [1976] introduces the Ethernet with a 3 Mbps prototype version. Digital *et. al.* [1980] specifies the original 10 Mbps Ethernet standard, with IEEE standard 802.3 reported in Nelson [1983]. Shoch, Dalal, and Redell [1982] provides an historical perspective of the Ethernet evolution. Related work on the ALOHA network is reported in Abramson [1970], with a survey of technologies given by Cotton [1979].

Token passing ring technology is proposed in Farmer and Newhall [1969]. Miller and Thompson [1982], as well as Andrews and Shultz [1982], provide summaries. Another alternative, the slotted ring network, is proposed by Pierce [1972]. For a comparison of technologies, see Rosenthal [1982].

For more information on the ARPANET see Cerf [1989] and BBN [1981]. The ideas behind X25NET are summarized in Comer and Korb [1983]; Lanzillo and Partridge [January 1989] describes dial-up IP. De Prycker [1993] describes Asynchronous Transfer Mode and its use for wide area services. Partridge [1994] surveys many gigabit technologies, including ATM, and describes the internal structure of high speed switches.

EXERCISES

2.1 Find out which network technologies your site uses.

2.2 What is the maximum size packet that can be sent on a high-speed network like Network System Corporation's Hyperchannel?

2.3 If your site uses Ethernet hub technology, find out how many connections can be attached to a single hub. If your site has multiple hubs (e.g., one on each floor of a building), find out how the hubs communicate.

2.4 What are the advantages and disadvantages of tunneling?

2.5 Read the Ethernet standard to find exact details of the inter-packet gap and preamble size. What is the maximum steady-state rate at which Ethernet can transport data?

2.6 What characteristic of a satellite communication channel is most desirable? Least desirable?

2.7 Find a lower bound on the time it takes to transfer a 5 megabyte file across a network that operates at: 28.8 Kbps, 1.54 Mbps, 10 Mbps, 100 Mbps, and 2.4 Gbps.

2.8 Does the processor, disk, and internal bus on your computer operate fast enough to send data from a disk file at 2 gigabits per second?

3

Internetworking Concept And Architectural Model

3.1 Introduction

So far we have looked at the low-level details of transmission across individual data networks, the foundation on which all computer communication is built. This chapter makes a giant conceptual leap by describing a scheme that allows us to collect the diverse network technologies into a coordinated whole. The primary goal is a system that hides the details of underlying network hardware while providing universal communication services. The primary result is a high-level abstraction that provides the framework for all design decisions. Succeeding chapters show how we use this abstraction to build the necessary layers of internet communication software and how the software hides the underlying physical transport mechanisms. Later chapters also show how applications use the resulting communication system.

3.2 Application-Level Interconnection

Designers have taken two different approaches to hiding network details, using application programs to handle heterogeneity or hiding details in the operating system. Early heterogeneous network interconnections provided uniformity through application-level programs called *application gateways*. In such systems, an application-level program, executing on each computer in the network, understands the details of the network connections for that computer, and interoperates across those connections with application programs on other computers. For example, some electronic mail systems

consist of mail programs that are each configured to forward a memo to a mail program on the next computer. The path from source to destination may involve many different networks, but that does not matter as long as the mail systems on all the machines cooperate by forwarding each message.

Using application programs to hide network details may seem natural at first, but such an approach results in limited, cumbersome communication. Adding new functionality to the system means building a new application program for each computer. Adding new network hardware means modifying existing programs (or creating new programs) for each possible application. On a given computer, each application program must understand the network connections for the computer, resulting in duplication of code.

Users who are experienced with networking understand that once the interconnections grow to hundreds or thousands of networks, no one can possibly build all the necessary application programs. Furthermore, success of the step-at-a-time communication scheme requires correctness of all application programs executing along the path. When an intermediate program fails, the source and destination remain unable to detect or control the problem. Thus, systems that use intermediate applications programs cannot guarantee reliable communication.

3.3 Network-Level Interconnection

The alternative to providing interconnection with application-level programs is a system based on network-level interconnection. A network-level interconnection provides a mechanism that delivers small packets of data from their original source to their ultimate destination without using intermediate application programs. Switching small units of data instead of files or large messages has several advantages. First, the scheme maps directly onto the underlying network hardware, making it extremely efficient. Second, network-level interconnection separates data communication activities from application programs, permitting intermediate computers to handle network traffic without understanding the applications that are sending or receiving it. Third, using network connections keeps the entire system flexible, making it possible to build general purpose communication facilities. Fourth, the scheme allows network managers to add new network technologies by modifying or adding a single piece of new network level software, while application programs remain unchanged.

The key to designing universal network-level interconnection can be found in an abstract communication system concept known as *internetworking*. The internetwork, or *internet*, concept is an extremely powerful one. It detaches the notions of communication from the details of network technologies and hides low-level details from the user. More important, it drives all software design decisions and explains how to handle physical addresses and routes. After reviewing basic motivations for internetworking, we will consider the properties of an internet in more detail.

We begin with two fundamental observations about the design of communication systems:

- No single network hardware technology can satisfy all constraints.
- Users desire universal interconnection.

The first observation is an economic as well as technical one. Inexpensive Local Area Networks that provide high speed communication only cover short distances; wide area networks that span long distances cannot supply local communication cheaply. Because no single network technology satisfies all needs, we are forced to consider multiple underlying hardware technologies.

The second observation is self-evident. Ultimately, users would like to be able to communicate between any two points. In particular, we desire a communication system that is not constrained by the boundaries of physical networks.

The goal is to build a unified, cooperative interconnection of networks that supports a universal communication service. Within each network, computers will use underlying technology-dependent communication facilities like those described in Chapter 2. New software, inserted between the technology-dependent communication mechanisms and application programs, will hide the low-level details and make the collection of networks appear to be a single large network. Such an interconnection scheme is called an *internetwork* or *internet*.

The idea of building an internet follows a standard pattern of system design: researchers imagine a high-level computing facility and work from available computing technology, adding layers of software until they have a system that efficiently implements the imagined high-level facility. The next section shows the first step of the design process by defining the goal more precisely.

3.4 Properties Of The Internet

The notion of universal service is important, but it alone does not capture all the ideas we have in mind for a unified internet because there can be many implementations of universal services. In our design, we want to hide the underlying internet architecture from the user. That is, we do not want to require users or application programs to understand the details of hardware interconnections to use the internet. We also do not want to mandate a network interconnection topology. In particular, adding a new network to the internet should not mean connecting to a centralized switching point, nor should it mean adding direct physical connections between the new network and all existing networks. We want to be able to send data across intermediate networks even though they are not directly connected to the source or destination computers. We want all computers in the internet to share a universal set of machine identifiers (which can be thought of as *names* or *addresses*).

Our notion of a unified internet also includes the idea of network independence in the user interface. That is, we want the set of operations used to establish communication or to transfer data to remain independent of the underlying network technologies and the destination computer. Certainly, a user should not have to understand the network interconnection topology when creating or using application programs that communicate.

3.5 Internet Architecture

We have seen how computers connect to individual networks. The question arises, "How are networks interconnected to form an internetwork?" The answer has two parts. Physically, two networks can only be connected by a computer that attaches to both of them. A physical attachment does not provide the interconnection we have in mind, however, because such a connection does not guarantee that the computer will cooperate with other machines that wish to communicate. To have a viable internet, we need special computers that are willing to transfer packets from one network to another. Computers that interconnect two networks and pass packets from one to the other are called *internet gateways* or *internet routers*[‡].

Consider an example consisting of two physical networks shown in Figure 3.1. In the figure, router *R* connects to both network *1* and network *2*. For *R* to act as a router, it must capture packets on network *1* that are bound for machines on network *2* and transfer them. Similarly, *R* must capture packets on network *2* that are destined for machines on network *1* and transfer them.

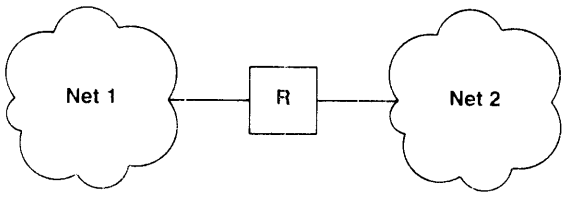

Figure 3.1 Two physical networks interconnected by *R*, a router (IP gateway).

In the figure, clouds are used to denote physical networks because the exact hardware is unimportant. Each network can be a LAN or a WAN, and each may have many computers attached or a few computers attached.

3.6 Interconnection Through IP Routers

Although it illustrates the basic connection strategy, Figure 3.1 is quite simplistic. In an actual internet that includes many networks and routers, each router needs to know about the topology of the internet beyond the networks to which it connects. For example, Figure 3.2 shows three networks interconnected by two routers.

[‡]The original literature used the term *IP gateway*. However, vendors have adopted the term *IP router* — the two terms are used interchangeably throughout this text.

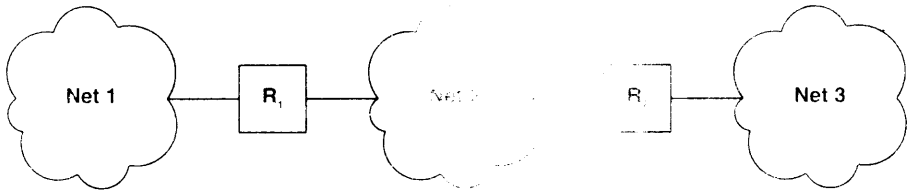

Figure 3.2 Three networks interconnected by two routers.

In this example, router R_1 must transfer from network *1* to network *2* all packets destined for computers on either network *2* or network *3*. For a large internet composed of many networks, the router's task of making decisions about where to send packets becomes more complex.

The idea of a router seems simple, but it is important because it provides a way to interconnect networks, not just computers. In fact, we have already discovered the principle of interconnection used throughout an internet:

> *In a TCP/IP internet, special computers called* IP routers *or* IP gateways *provide interconnections among physical networks.*

You might suspect that routers, which must each know how to forward packets toward their destination, are large machines with enough primary or secondary memory to hold information about every computer in the internet to which they attach. In fact, routers used with TCP/IP internets are usually small computers. They often have little disk storage and modest main memories. The trick to building a small internet router lies in the following concept:

> *Routers use the destination network, not the destination computer, when forwarding a packet.*

If packet forwarding is based on networks, the amount of information that a router needs to keep is proportional to the number of networks in the internet, not the number of computers.

Because routers play a key role in internet communication, we will return to them in later chapters and discuss the details of how they operate and how they learn about routes. For now, we will assume that it is possible and practical to have correct routes for all networks in each router in the internet. We will also assume that only routers provide connections between physical networks in an internet.

3.7 The User's View

Remember that TCP/IP is designed to provide a universal interconnection among computers independent of the particular networks to which they attach. Thus, we want a user to view an internet as a single, virtual network to which all machines connect despite their physical connections. Figure 3.3a shows how thinking of an internet instead of constituent networks simplifies the details and makes it easy for the user to conceptualize communication. In addition to routers that interconnect physical networks, software is needed on each computer to allow application programs to use an internet as if it were a single, physical network.

The advantage of providing interconnection at the network level now becomes clear. Because application programs that communicate over the internet do not know the details of underlying connections, they can be run without change on any computer. Because the details of each machine's physical network connections are hidden in the internet software, only the internet software needs to change when new physical connections are added or existing connections are removed. In fact, it is possible to optimize the internal structure of the internet by altering physical connections while application programs are executing.

A second advantage of having communication at the network level is more subtle: users do not have to understand, remember, or specify how networks connect or what traffic they carry. Application programs can be written that communicate independent of underlying physical connectivity. In fact, network managers are free to change interior parts of the underlying internet architecture without changing application software in most of the computers attached to the internet (of course, network software must be reconfigured when a computer moves to a new network).

As Figure 3.3b shows, routers do not provide direct connections among all pairs of networks. It may be necessary for traffic traveling from one computer to another to pass through several routers as the traffic crosses intermediate networks. Thus, networks participating in an internet are analogous to highways in the U.S. interstate system: each net agrees to handle transit traffic in exchange for the right to send traffic throughout the internet. Typical users are unaffected and unaware of extra traffic on their local network.

3.8 All Networks Are Equal

Chapter 2 reviewed examples of the network hardware used to build TCP/IP internets, and illustrated the great diversity of technologies. We have described an internet as a collection of cooperative, interconnected networks. It is now important to understand a fundamental concept: from the internet point of view, any communication system capable of transferring packets counts as a single network, independent of its delay and throughput characteristics, maximum packet size, or geographic scale. In particular, Figure 3.3b uses the same small cloud shape to depict each physical network because TCP/IP treats them equally despite their differences. The point is:

The TCP/IP internet protocols treat all networks equally. A Local Area Network like an Ethernet, a Wide Area Network used as a backbone, or a point-to-point link between two computers each count as one network.

Readers unaccustomed to internet architecture may find it difficult to accept such a simplistic view of networks. In essence, TCP/IP defines an abstraction of "network" that hides the details of physical networks; we will learn that such abstractions help make TCP/IP extremely powerful.

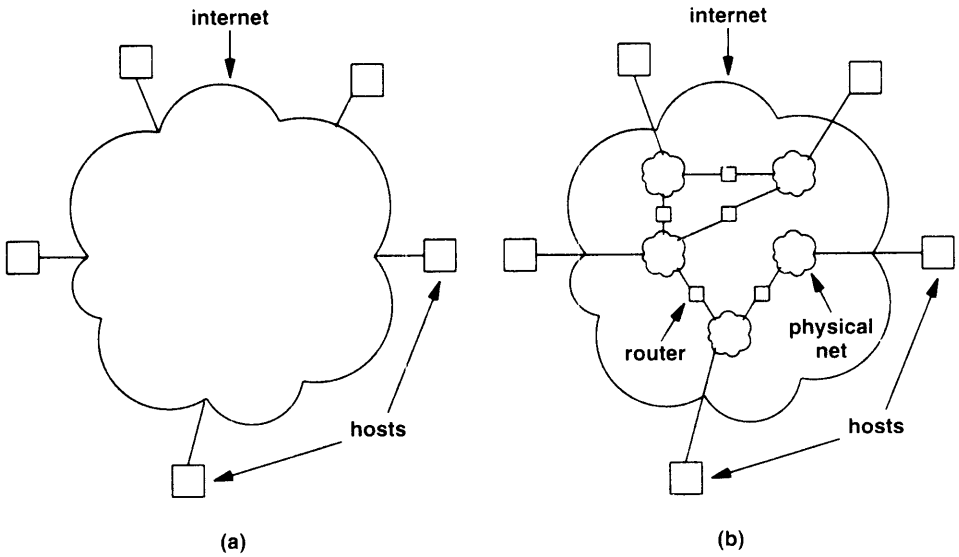

Figure 3.3 (a) The user's view of a TCP/IP internet in which each computer appears to attach to a single large network, and (b) the structure of physical networks and routers that provide interconnection.

3.9 The Unanswered Questions

Our sketch of internets leaves many unanswered questions. For example, you might wonder about the exact form of internet addresses assigned to computers or how such addresses relate to the Ethernet, FDDI, or ATM physical hardware addresses described in Chapter 2. The next three chapters confront these questions. They describe the format of IP addresses and illustrate how software on a computer maps between internet addresses and physical addresses. You might also want to know exactly what a packet looks like when it travels through an internet, or what happens when packets arrive too fast for some computer or router to handle. Chapter 7 answers these

questions. Finally, you might wonder how multiple application programs executing concurrently on a single computer can send and receive packets to multiple destinations without becoming entangled in each other's transmissions or how internet routers learn about routes. All of these questions will be answered as well.

Although it may seem vague now, the direction we are following will let us learn about both the structure and use of internet protocol software. We will examine each part, looking at the concepts and principles as well as technical details. We began by describing the physical communication layer on which an internet is built. Each of the following chapters will explore one part of the internet software, until we understand how all the pieces fit together.

3.10 Summary

An internet is more than a collection of networks interconnected by computers. Internetworking implies that the interconnected systems agree to conventions that allow each computer to communicate with every other computer. In particular, an internet will allow two computers to communicate even if the communication path between them passes across a network to which neither connects directly. Such cooperation is only possible when computers agree on a set of universal identifiers and a set of procedures for moving data to its final destination.

In an internet, interconnections among networks are formed by computers called IP routers, or IP gateways, that attach to two or more networks. A router forwards packets between networks by receiving them from one network and sending them to another.

FOR FURTHER STUDY

Our model of an internetwork comes from Cerf and Cain [1983] and Cerf and Kahn [1974], which describe an internet as a set of networks interconnected by routers and sketch an internet protocol similar to that eventually developed for the TCP/IP protocol suite. More information on the connected Internet architecture can be found in Postel [1980]; Postel, Sunshine, and Chen [1981]; and in Hinden, Haverty, and Sheltzer [1983]. Shoch [1978] presents issues in internetwork naming and addressing. Boggs *et. al.* [1980] describes the internet developed at Xerox PARC, an alternative to the TCP/IP internet we will examine. Cheriton [1983] describes internetworking as it relates to the V-system.

EXERCISES

3.1 What processors have been used as routers in the connected Internet? Does the size and speed of early router hardware surprise you? Why?

3.2 Approximately how many networks comprise the internet at your site? Approximately how many routers?

3.3 Consider the internal structure of the example internet shown in Figure 3.3b. Which routers are most crucial? Why?

3.4 Changing the information in a router can be tricky because it is impossible to change all routers simultaneously. Investigate algorithms that guarantee to either install a change on a set of computers or install it on none.

3.5 In an internet, routers periodically exchange information from their routing tables, making it possible for a new router to appear and begin routing packets. Investigate the algorithms used to exchange routing information.

3.6 Compare the organization of a TCP/IP internet to the style of internet designed by Xerox Corporation.

4

Classful Internet Addresses

4.1 Introduction

The previous chapter defines a TCP/IP internet as a virtual network built by interconnecting physical networks with routers. This chapter discusses addressing, an essential ingredient that helps TCP/IP software hide physical network details and makes the resulting internet appear to be a single, uniform entity.

4.2 Universal Identifiers

A communication system is said to supply *universal communication service* if it allows any host computer to communicate with any other host. To make our communication system universal, it needs a globally accepted method of identifying each computer that attaches to it.

Often, host identifiers are classified as *names*, *addresses*, or *routes*. Shoch [1978] suggests that a name identifies *what* an object is, an address identifies *where* it is, and a route tells *how* to get there†. Although these definitions are intuitive, they can be misleading. Names, addresses, and routes really refer to successively lower level representations of host identifiers. In general, people usually prefer pronounceable names to identify machines, while software works more efficiently with compact representations of identifiers that we think of as addresses. Either could have been chosen as the TCP/IP universal host identifiers. The decision was made to standardize on compact, binary addresses that make computations such as the selection of a route efficient. For now, we will discuss only binary addresses, postponing until later the questions of how to map between binary addresses and pronounceable names, and how to use addresses for routing.

†An identifier that specifies where an object can be found is also called a *locator*.

4.3 The Original Classful Addressing Scheme

Think of an internet as a [illegible] network like any other physical network. The difference, of course, is that [illegible] is a virtual structure, imagined by its designers, and implemented entirely in software. The designers are free to choose packet formats and sizes, addresses, delivery techniques, and so on; nothing is dictated by hardware. For addresses, the designers of TCP/IP chose a scheme analogous to physical network addressing in which each host on the internet is assigned a 32-bit integer address called its *internet address* or *IP address*. The clever part of internet addressing is that the integers are carefully chosen to make routing efficient. Specifically, an IP address encodes the identification of the network to which a host attaches as well as the identification of a unique host on that network. We can summarize:

> *Each host on a TCP/IP internet is assigned a unique 32-bit internet address that is used in all communication with that host.*

The details of IP addresses help clarify the abstract ideas. For now, we give a simplified view and expand it later. In the simplest case, each host attached to an internet is assigned a 32-bit universal identifier as its internet address. A prefix of an IP address identifies a network. That is, the IP addresses in all hosts on a given network share a common prefix.

Conceptually, each address is a pair (*netid*, *hostid*), where *netid* identifies a network, and *hostid* identifies a host on that network. In practice, however, the partition into prefix and suffix is not uniform throughout the entire internet because the designers did not specify a single boundary. In the original addressing scheme, which is known as *classful*, each IP address had one of the first three forms shown in Figure 4.1*.

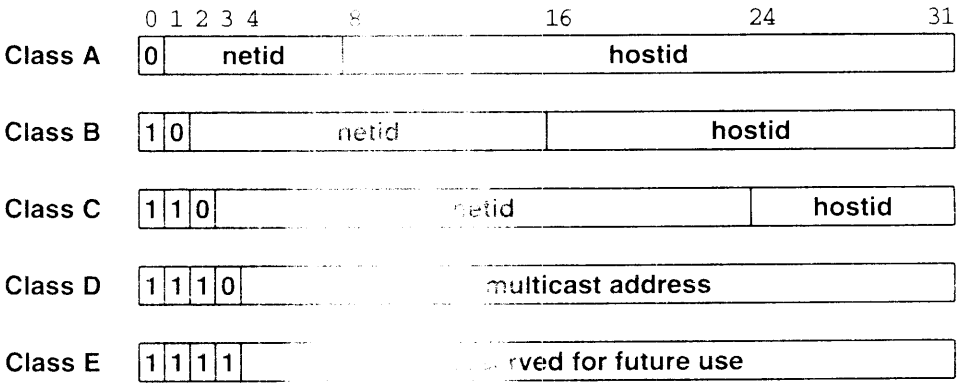

Figure 4.1 The five [illegible] (IP) addresses used with the original classful [illegible] The three primary classes, *A*, *B* and *C*, can be [illegible] first three bits.

*The fourth form, reserved [illegible] will be described later; for now, we will restrict our comments to the forms that specify

In the classful addressing scheme, each address is said to be *self-identifying* because the boundary between prefix and suffix can be computed from the address alone, without reference to external information. In particular, the class of an address can be determined from the three high-order bits, with two bits being sufficient to distinguish among the three primary classes. Class *A* addresses, used for the handful of networks that have more than 2^{16} (i.e., 65,536) hosts, devote 7 bits to netid and 24 bits to hostid. Class *B* addresses, used for intermediate size networks that have between 2^8 (i.e., 256) and 2^{16} hosts, allocate 14 bits to the netid and 16 bits to the hostid. Finally, class *C* addresses, used for networks that have less than 2^8 hosts, allocate 21 bits to the netid and only 8 bits to the hostid. Note that the IP address was originally defined in such a way that it was possible to extract the hostid or netid portions quickly. Efficiency was especially important for routers, which use the netid portion of an address when deciding where to send a packet. We will return to the discussion of efficient route lookup after examining recent changes and extensions to the addressing scheme.

4.4 Addresses Specify Network Connections

To simplify the discussion, we said that an internet address identifies a host, but that is not strictly accurate. Consider a router that attaches to two physical networks. How can we assign a single IP address if the address encodes a network identifier as well as a host identifier? In fact, we cannot. When conventional computers have two or more physical connections they are called *multi-homed hosts*. Multi-homed hosts and routers require multiple IP addresses. Each address corresponds to one of the machine's network connections. Looking at multi-homed hosts leads to the following important idea:

> *Because IP addresses encode both a network and a host on that network, they do not specify an individual computer, but a connection to a network.*

Thus, a router connecting n networks has n distinct IP addresses, one for each network connection.

4.5 Network And Directed Broadcast Addresses

We have already cited the major advantage of encoding network information in internet addresses: it makes efficient routing possible. Another advantage is that internet addresses can refer to networks as well as hosts. By convention, hostid *0* is never assigned to an individual host. Instead, an IP address with hostid portion equal to zero is used to refer to the network itself. In summary:

Internet addresses can be used to refer to networks as well as individual hosts. By convention, an address that has all bits of the hostid equal to 0 *is reserved to refer to the network.*

Another significant advantage of the internet addressing scheme is that it includes a *directed broadcast address* that refers to all hosts on the network. According to the standard, any address with the hostid consisting of all *1s* is reserved for directed broadcast†. When a packet is sent to such an address, a single copy of the packet is transferred across the internet from the source. Routers along the path use the netid portion of the address when choosing a path; they do not look at the host portion. Once the packet reaches a router attached to the final network, that router examines the host portion of the address to determine how to deliver the packet. If it finds all *1s*, the router broadcasts the packet to all hosts on the network.

On many network technologies (e.g., Ethernet), broadcasting is as efficient as unicast transmission; on others, broadcasting is supported by the network software, but requires substantially more delay than single transmission. Some network hardware does not support broadcast at all. Thus, having an IP directed broadcast address does not guarantee the availability or efficiency of broadcast delivery. In summary,

IP addresses can be used to specify a directed broadcast in which a packet is sent to all computers on a network; such addresses map to hardware broadcast, if available. By convention, a directed broadcast address has a valid netid and has a hostid with all bits set to 1.

4.6 Limited Broadcast

The broadcast address we just described is known as *directed* because it contains both a valid network ID and the broadcast hostid. A directed broadcast address can be interpreted unambiguously at any point in an internet because it uniquely identifies the target network in addition to specifying broadcast on that network. Directed broadcast addresses provide a powerful (and somewhat dangerous) mechanism that allows a remote system to send a single packet that will be broadcast on the specified network.

From an addressing point of view, the chief disadvantage of directed broadcast is that it requires knowledge of the network address. Another form of broadcast address, called a *limited broadcast address* or *local network broadcast address*, provides a broadcast address for the local network independent of the assigned IP address. The local broadcast address consists of thirty-two *1*s (hence, it is sometimes called the "all *1s*" broadcast address). A host may use the limited broadcast address as part of a start-up procedure before it learns its IP address or the IP address prefix for the local network. Once the host learns the correct IP address for the local network, however, it should use directed broadcast.

†Unfortunately, an early release of TCP/IP code that accompanied Berkeley UNIX incorrectly used all zeroes for broadcast. Because the error still survives, TCP/IP software often includes an option that allows a site to use all zeroes for directed broadcast.

As a general rule, TCP/IP protocols restrict broadcasting to the smallest possible set of machines. We will see how this rule affects multiple networks that share addresses in the chapter on subnet addressing.

4.7 Interpreting Zero To Mean "This"

We have seen that a field consisting of *1*s can be interpreted to mean "all," as in "all hosts" on a network. In general, internet software interprets fields consisting of *0*s to mean "this." The interpretation appears throughout the literature. Thus, an IP address with hostid *0* refers to "this" host, and an internet address with network ID *0* refers to "this" network. Of course, it is only meaningful to use such an address in a context where it can be interpreted unambiguously. For example, if a machine receives a packet in which the netid portion of the destination address is *0* and the hostid portion of the destination address matches its address, the receiver interprets the netid field to mean "this" network (i.e., the network over which the packet arrived).

Using netid *0* is especially important in those cases where a host wants to communicate over a network but does not yet know the network IP address. The host uses network ID *0* temporarily, and other hosts on the network interpret the address as meaning "this" network. In most cases, replies will have the network address fully specified, allowing the original sender to record it for future use. Chapters 9 and 23 will discuss in detail mechanisms a host can use to determine the network ID of the local network.

4.8 Subnet And Supernet Extensions

The addressing scheme described so far requires a unique network prefix for each physical network. Although that was, indeed, the original plan, it did not last long. In the 1980s as Local Area Network technologies became increasingly popular, it became apparent that requiring a unique prefix for each physical network would exhaust the address space quickly. Consequently, an addressing extension was developed to conserve network prefixes. Known as *subnet addressing*, the scheme allows multiple physical networks to share a prefix.

In the 1990s, a second extension was devised that ignored the classful hierarchy and allowed the division between prefix and suffix to occur at an arbitrary point. Called *classless addressing* or *supernetting*, the scheme allows more complete utilization of the address space.

Chapter 10 will consider details of the subnet and supernet addressing extensions. For now, it is only important to know that the addressing scheme has been extended, and that the original classful scheme described in this chapter is no longer the most widely used.

4.9 IP Multicast Addresses

In addition to *unicast delivery*, in which a packet is delivered to a single computer, and *broadcast delivery*, in which a packet is delivered to all computers on a given network, the IP addressing scheme supports a special form of multipoint delivery known as *multicasting*, in which a packet is delivered to a specific subset of hosts. IP multicasting is especially useful for networks where the hardware technology supports multicast delivery. Chapter 17 discusses multicast addressing and delivery in detail. For now, it is sufficient to understand that Class *D* addresses are reserved for multicasting.

4.10 Weaknesses In Internet Addressing

Encoding network information in an internet address does have some disadvantages. The most obvious disadvantage is that addresses refer to network connections, not to the host computer:

> *If a host computer moves from one network to another, its IP address must change.*

To understand the consequences, consider a traveler who wishes to disconnect his or her personal computer, carry it along on a trip, and reconnect it to the Internet after reaching the destination. The personal computer cannot be assigned a permanent IP address because an IP address identifies the network to which the machine attaches. Chapter 19 shows how the IP addressing scheme makes *mobility* a complex problem.

Another weakness of the classful addressing scheme is that when any class *C* network grows to more than 255 hosts, it must have its address changed to a class *B* address. While this may seem like a minor problem, changing network addresses can be incredibly time-consuming and difficult to debug. Because most software is not designed to handle multiple addresses for the same physical network, administrators cannot plan a smooth transition in which they introduce new addresses slowly. Instead, they must abruptly stop using one network address, change the addresses of all machines, and then resume communication using the new network address.

The most important flaw in the internet addressing scheme will not become fully apparent until we examine routing. However, its importance warrants a brief introduction here. We have suggested that routing will be based on internet addresses, with the netid portion of an address used to make routing decisions. Consider a host with two connections to the internet. We know that such a host must have more than one IP address. The following is true:

> *Because routing uses the network portion of the IP address, the path taken by packets traveling to a host with multiple IP addresses depends on the address used.*

The implications are surprising. Humans think of each host as a single entity and want to use a single name. They are often surprised to find that they must learn more than one name and even more surprised to find that packets sent using multiple names can behave differently.

Another surprising consequence of the internet addressing scheme is that merely knowing one IP address for a destination may not be sufficient; it may be impossible to reach the destination using that address. Consider the example internet shown in Figure 4.2. In the figure, two hosts, *A* and *B*, both attach to network *1*, and usually communicate directly using that network. Thus, users on host *A* should normally refer to host *B* using IP address I_3. An alternate path from *A* to *B* exists through router *R*, and is used whenever *A* sends packets to IP address I_5 (B's address on network *2*). Now suppose *B*'s connection to network *1* fails, but the machine itself remains running (e.g., a wire breaks between *B* and network *1*). Users on *A* who specify IP address I_3 cannot reach *B*, although users who specify address I_5 can. These problems with naming and addressing will arise again in later chapters when we consider routing and name binding.

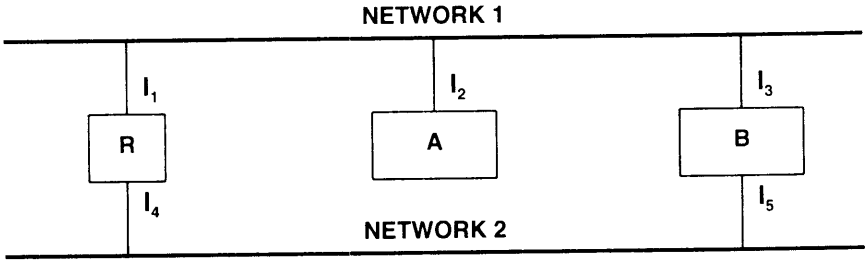

Figure 4.2 An example internet with a multi-homed host, *B*, that demonstrates a disadvantage of the IP addressing scheme. If interface I_3 becomes disconnected, *A* must use address I_5 to reach *B*, sending packets through router *R*.

4.11 Dotted Decimal Notation

When communicated to humans, either in technical documents or through application programs, IP addresses are written as four decimal integers separated by decimal points, where each integer gives the value of one octet of the IP address†. Thus, the 32-bit internet address

10000000 00001010 00000010 00011110

is written

128.10.2.30

†Dotted decimal notation is sometimes called *dotted quad notation*.

In the original classful scheme, the Internet authority chose an address appropriate to the size of the network. A class C number was assigned to a network with a small number of attached computers (less than 255); class B numbers were reserved for larger networks. Finally, a network needed to have more than 65,535 hosts before it could obtain a class A number. The address space was skewed because most networks are small, fewer are of medium size, and only a handful are gigantic.

Most organizations never interact with the central authority directly. Instead, to connect its networks to the global Internet, an organization usually contracts with a local *Internet Service Provider (ISP)*. In addition to providing a connection between the organization and the rest of the Internet, an ISP obtains a valid address prefix for each of the customer's networks. Many local ISPs are, in fact, customers of larger ISPs — when a customer requests an address prefix, the local ISP merely obtains a prefix from a larger ISP. Thus, only the largest ISPs need to contact ICANN.

Note that the central authority only assigns the network portion of an address; once an organization obtains a prefix for a network, the organization can choose how to assign a unique suffix to each host on the network without contacting the central authority. Furthermore, remember that it is only essential for the central authority to assign IP addresses for networks that are (or will be) attached to the global Internet.

4.15 Reserved Address Prefixes

We said that as long as it never connects to the outside world, an individual corporation has responsibility for assigning unique network addresses within its TCP/IP internet. Indeed, many corporate groups that use TCP/IP protocols do assign internet addresses on their own. For example, the network address 9.0.0.0 has been assigned to IBM Corporation, and address 12.0.0.0 has been assigned to AT&T. If an organization decides to use TCP/IP protocols on two of their networks with no connections to the global Internet, the organization can choose to assign addresses 9.0.0.0 and 12.0.0.0 to their local networks.

Experience has shown, however, that it is unwise to create a private internet using the same network addresses as the global Internet because most sites eventually connect to the Internet and doing so may cause problems when trying to exchange software with other sites. To avoid addressing conflicts between addresses used on private internets and addresses used on the global Internet, the IETF reserved several address prefixes, and recommends using them on private internets. Because the set of reserved prefixes includes both classful and classless values, they are described in Chapter 10.

4.16 An Example

To clarify the IP addressing scheme, consider an example of two networks in the Computer Science Department at Purdue University as they were connected to the Internet in the mid-1980s. Figure 4.5 shows the network addresses, and illustrates how routers interconnect the networks.