Second Edition

Internetworking with TCP/IP

VOLUME I

Principles, Protocols, and Architecture

Douglas E. Comer

Library of Congress Cataloging-in-Publication Data

Comer, Douglas E.
Internetworking with TCP/IP / Douglas E. Comer. -- 2nd ed.
p. cm.
Includes bibliographical references (v. 1, p.) and index.
Contents: vol. 1. Principles, protocols, and architecture.
ISBN 0-13-468505-9 (v. 1)
1. Computer networks. 2. Computer network protocols. 3. Data transmission systems. I. Title.
TK5105.5.C59 1991
004.6--dc20 90-7829
CIP

Editorial/production supervision: Joe Scordato
Cover design: Karen Stephens
Cover illustration: Jim Kinstrey
Manufacturing buyers: Linda Behrens and Patrice Fraccio

A Paramount Communications Company
Englewood Cliffs, New Jersey 07632

Printed in the United States of America

10 9 8

ISBN 0-13-468505-9

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

anyone can build the software needed to communicate across an internet. More important, the entire technology has been designed to foster communication between machines with diverse hardware architectures, to use almost any packet switched network hardware, and to accommodate multiple computer operating systems.

To appreciate internet technology, think of how it affects research. Imagine for a minute the effects of interconnecting all the computers used by scientists. Any scientist would be able to exchange data resulting from an experiment with any other scientist. It would be possible to establish national data centers to collect data from natural phenomena and make the data available to all scientists. Computer services and programs available at one location could be used by scientists at other locations. As a result, the speed with which scientific investigations proceed would increase. In short, the changes would be dramatic.

1.2 The TCP/IP Internet

Government agencies have realized the importance and potential of internet technology for many years and have been funding research that will make possible a national internet. This book discusses principles and ideas underlying the leading internet technology, one that has resulted from research funded by the *Defense Advanced Research Projects Agency* (*DARPA*). The DARPA 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 long haul 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 a large internet that connects most major research institutions, including university, corporate, and government labs. 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*) all participate, using TCP/IP to connect many of their research sites with those of DARPA. The resulting entity, known as the *connected Internet*, the *DARPA/NSF Internet*, the *TCP/IP Internet*, or just the *Internet*†, allows researchers at connected institutions to share information with colleagues across the country as easily as they share it with researchers 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.

†We will follow the usual convention of capitalizing *Internet* when referring specifically to the connected internet, and use lower case otherwise; we will also assume the term "internet" used without further qualification refers to TCP/IP internets.

2-5

Most of the material in this book applies to any internet that uses TCP/IP, but some chapters refer specifically to the connected 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 sections of the text that describe the connected Internet completely – many corporate networks are already more complex than the connected Internet of ten years ago, and many of the problems they face have already been solved in the connected Internet.

1.3 Internet Services

One cannot appreciate the technical details underlying TCP/IP without understanding the services it provides. This chapter reviews internet services briefly, highlighting the services most users access, and leaving 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, give the formulas for passing messages, specify the details of message formats, and describe how to handle error conditions. Most important, they allow us to discuss communication standards independent of any particular vendor's network hardware. In a sense, protocols are to communication what programming languages are to computation. A programming language allows one to specify or understand a computation without knowing the details of any 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. They can create new programs quickly. Second, because programs built using higher-level abstractions are not restricted to a particular machine architecture or particular network hardware, they do not need to be changed when machines or networks are reconfigured. Third, because application programs built using higher-level protocols are independent of the underlying hardware, they can provide direct communication for an arbitrary pair of machines. Programmers do not need to build special versions of application software to move and translate data between each possible pair of machine types.

We will see that all network services are described by protocols. The next sections refer to protocols used to specify application-level services as well as those used to define network-level services. Later chapters explain each of these protocols in more detail.

1.3.1 Application Level Internet Services

From the user's point of view, a TCP/IP internet appears to be a set of application programs that use the 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. We say that internet application programs exhibit a high degree of interoperability. Most users that access the Internet do so merely by running application programs without understanding the TCP/IP technology, the structure of the underlying internet, or even the path their data travels to its destination; they rely on the application programs to handle such details. Only programmers who write such application programs view the internet as a network and need to understand the details of the technology.

The most popular and widespread Internet application services include:

- *Electronic mail.* Electronic mail allows a user to compose memos and send them to individuals or groups. Another part of the mail application allows users to read memos that they have received. Electronic mail has been so successful that many Internet users depend on it for normal business correspondence. Although many electronic mail systems exist, it is important to understand that using TCP/IP makes mail delivery more reliable. Instead of relying on intermediate machines to relay mail messages, the TCP/IP mail delivery system operates by having the sender's machine contact the receiver's machine directly. Thus, the sender knows that once the message leaves the local machine, it has been successfully received at the destination site.
- *File transfer.* Although users sometimes transfer files using electronic mail, mail is designed primarily for short, text files. The TCP/IP protocols include a file transfer application program that allows users to send or receive arbitrarily large files of programs or data. For example, using the file transfer program, one can copy from one machine to another large data banks containing satellite images, programs written in FORTRAN or Pascal, or an English dictionary. 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.* Perhaps the most interesting Internet application, 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 the user's terminal or workstation connects directly to the remote machine by sending every keystroke from the user's keyboard to the remote machine and displaying every character the remote computer prints on the user's terminal screen. When the remote login session terminates, the application returns the user to the local system.

We will return to each of these 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 writes 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 machine 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.* While TCP/IP is based on conventional packet switching technology, it is independent of any particular vendor's hardware. The connected 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 source and ultimate destination instead of between successive machines along the path, even when the two machines 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 almost universal adoption as well as the size and growth rate of the connected Internet. DARPA 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, DARPA 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 DARPA had also funded exploration of packet-switching over radio networks and satellite communication channels. Indeed, the growing diversity of network hardware technologies helped force DARPA to study network interconnection, and pushed internetworking forward.

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

The connected Internet began around 1980 when DARPA 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 *MILNET*.

To encourage university researchers to adopt and use the new protocols, DARPA 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, Inc. (BBN) to implement its TCP/IP protocols for use with UNIX, and funding Berkeley to integrate the protocols with its software distribution, DARPA was able to reach over 90% of the 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 and no others were generally available.

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 lay 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 provides a new operating system abstraction known as a *socket* that allows 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 took an active role in expanding the TCP/IP Internet to reach as many scientists as possible. Starting in 1985, it began a program to establish access networks centered around its six supercomputer centers. In 1986 it expanded networking efforts by funding a new long haul 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 connected Internet.

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

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 late 1987 it was estimated that the growth had reached 15% per month and remained high for the following two years. By 1990, the connected Internet included over 3,000 active networks and over 200,000 computers.

Adoption of TCP/IP protocols and growth of the Internet has not been limited to government-funded projects. Major computer corporations are all connected to the Internet as well as many other large corporations including: oil companies, the auto industry, electronics firms, and telephone companies. In addition, many companies use the TCP/IP protocols on their internal corporate internets even though they choose not to be part of the connected 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, requests to update the file would soon exceed the capacity of people 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 every site.

New protocols were developed and a naming system was put in place across the connected Internet that allows any user to resolve the name of a remote machine automatically. Known as the *Domain Name System*, 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 Original Internet Activities 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 Activities 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 connected Internet. It decides which protocols are a required part of the TCP/IP suite and sets official policies.

Formed in 1983 when DARPA reorganized the Internet Control and Configuration Board, the IAB inherited much of its charter from the earlier group. Its initial goals were to encourage exchange 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 a DARPA-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 applications 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 DARPA 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 New IAB Organization

By the summer of 1989, both the TCP/IP technology and the connected 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 their 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 connected 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.

The new IAB organization and names for subparts can best be explained by the diagram in Figure 1.1.

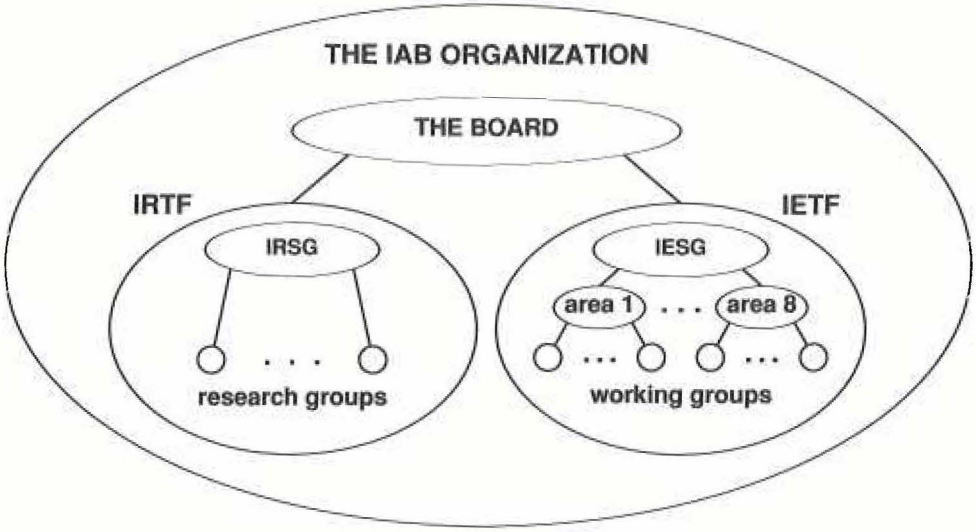


Figure 1.1 The structure of the IAB after the 1989 reorganization.

As Figure 1.1 shows, in addition to the Board itself, the IAB organization contains 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 old 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 had grown to include dozens of active members who worked on many problems concurrently. Before the reorganization, the IETF had been divided into over 20 *working groups*, each working 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.

The reorganized IAB structure retained the IETF, but split it into eight areas, each with its own manager. The IETF chairman and the eight IETF area managers comprise the *Internet Engineering Steering Group (IESG)*, the individuals responsible for coordinating all efforts of IETF working groups.

Because the IETF was widely known throughout the Internet, and because its meetings were widely recognized and attended, the name "IETF" was preserved in the reorganization and still refers to the entire body, including the chairman, area managers, and all members of working groups. Similarly, the name "IETF working group" was retained.

Created during the reorganization, the Internet Research Task Force was given a name that denotes it as 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* or *IRSG*, that sets priorities and coordinates research activities. Unlike the IETF, however, the IRTF is currently a much smaller organization. Each member of the IRSG chairs a volunteer *Internet Research Group* analogous to the IETF working groups; the IRTF is not divided into areas.

1.7 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, DCA funds a group at *SRI International* to maintain and distribute information about TCP/IP and the connected Internet. Known as the *Network Information Center* or simply *The NIC*†, the group handles many administrative details for the Internet in addition to distributing documentation.

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. (Preliminary versions of RFCs are known as *Internet drafts*.) RFCs can be short or long, can cover broad concepts or details, and can be standards or merely proposals for new protocols. The RFC editor is called the *Deputy Internet Architect*, and is a member of the IAB. While RFCs are edited, they are not refereed in the same way as academic research papers. Also, some 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 IENs throughout the text.

Both the RFC and IEN note series are numbered sequentially in the chronological order they 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 index is available to help identify the correct version.

The NIC distributes RFCs and IENs to the community. You can obtain RFCs from the NIC by postal mail, by electronic mail, or directly across the Internet using a file transfer program. Ask a local network expert how to obtain RFCs at your site, or refer to Appendix *1* for further instructions on how to retrieve them.

†Pronounced "The Nick" after its acronym.

1.8 Internet Protocols and Standardization

Readers familiar with data communication networks realize that many 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, but be prepared to migrate to international 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 international standards. It came about merely because none of the existing protocols satisfied the need. The philosophy of adopting standards when they become available also means that when international standards arise and provide the same interoperability as TCP/IP, the Internet will migrate from TCP/IP to those new standards. These ideas agree with the federal government, which has adopted an Opens Systems Profile that specifies the adoption and use of the International Standards Organization's internet technology whenever the technology offers the equivalent functionality of TCP/IP.

1.9 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 has added considerable complexity to the system by introducing its backbone network, several regional networks, and hundreds of campus networks. Other groups continue to connect to the Internet as well. The most significant change comes not from added network connections, however, but from additional traffic. Physicists, chemists, and space scientists manipulate and exchange much larger volumes of data than computer science researchers who accounted for much of the early Internet traffic. These other scientists introduced substantial load when they began using the Internet, and the load has increased steadily as they continue to find new uses.

To accommodate growth in traffic, the capacity of the NSFNET backbone has already been increased twice, making the current capacity approximately 28 times larger than the original; an additional increase by another factor of 30 is scheduled for late 1990. At the current time, it is difficult to foresee an end to the need for more capacity.

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 hardware technology plays only a minor role in the overall design, it is important to be able to distinguish between the low-level mechanisms provided by the hardware itself and the higher-level facilities that the Internet protocol software provides. It is also important to understand how the facilities supplied by 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 these physical addresses.

2.2 Two Approaches To Network Communication

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

Packet-switched networks, the type usually used to connect computers, take an entirely different approach. In a packet-switched network, traffic on the 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 computers on the network to know whether it is destined for them or how to send it on to its correct destination. For example, a file to be transmitted between two machines may 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 network 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 machines 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, packet-switched networks have become extremely popular. The motivations for adopting packet switching are cost and performance. Because multiple machines can share a network, fewer interconnections 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 packet-switching that, throughout the remainder of this text, the term *network* will refer only to packet-switched networks.

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

2.3 Wide Area, Metropolitan Area, and Local Area Networks

Packet-switched 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 three broad categories: *wide area networks* (*WAN*s), *Metropolitan Area Networks* (*MAN*s), and *Local Area Networks* (*LAN*s).

WAN technologies, sometimes called *long haul networks*, allow endpoints to be arbitrarily far apart and are intended for use over large distances. Usually, WANs operate at slower speeds than other technologies and have much greater delay between connections. Typical speeds for a WAN range from 9.6 Kbps to 45 Mbps (million bits per second).

The newest type of network hardware, MAN technologies span intermediate geographic areas and operate at medium-to-high speeds. The name is derived from the ability of a single MAN to span a large metropolitan area. MANs introduce less delay than WANs, but cannot span as large a distance. Typical MANs operate at 56 Kbps to 100 Mbps.

LAN technologies provide the highest speed connections among computers, but sacrifice the ability to span large distances. For example, a typical LAN spans a small area like a single building or a small campus and operates between 4 Mbps and 2 Gbps (billion bits per second).

We have already mentioned the general tradeoff between speed and distance: technologies that provide higher speed communication operate over shorter distances. There are other differences among technologies in the three categories as well. In LAN technologies, each computer usually contains a network interface device that connects the machine directly to the network medium (e.g., a passive copper wire or coaxial cable). Often, the network itself is passive, depending on electronic devices in the attached computers to generate and receive the necessary electrical signals. In MAN technologies, a network contains active switching elements that introduce short delays as they route data to its destination. In WAN technologies, a network usually consists of a series of complex packet switches interconnected by communication lines. The size of the network can be extended by adding a new switch and another communication line. Attaching a computer to a WAN means connecting it to one of the packet switches. The switches introduce significant delays when routing traffic. Thus, the larger the WAN becomes the longer it takes to route traffic across it.

The goal of network protocol design is to hide the technological differences between networks, making interconnection independent of the underlying hardware. The next sections present six examples of network technologies used throughout 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.

2.4 Ethernet Technology

Ethernet is the name given to a popular local area packet-switched network technology invented at Xerox PARC in the early 1970s. The version described here was standardized by Xerox Corporation, Intel Corporation, and Digital Equipment Corporation in 1978. As Figure 2.1 shows, an Ethernet consists of a coaxial cable 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. Called the *ether*, the cable itself is completely passive; all the active electronic components that make the network function are associated with computers that are attached to the network.

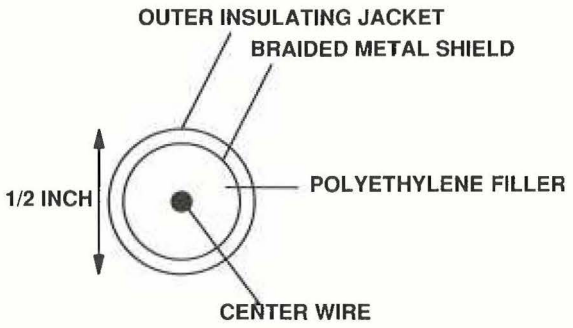


Figure 2.1 Coaxial cable used in an Ethernet

Ethernets may be extended with hardware devices called *repeaters* that relay electrical signals from one cable to another. Figure 2.2 shows a typical use of repeaters in an office building. A single backbone 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. Only two repeaters can be placed between any two machines, so the total length of a single Ethernet is still rather short (1500 meters).

Extending an Ethernet by using repeaters has advantages and disadvantages. Repeaters are less expensive than other types of interconnection hardware, making them the least costly way to extend an Ethernet. However, repeaters have two disadvantages. First, because repeaters repeat and amplify all electrical signals, they also copy electrical disturbance or errors that occur on one wire to the other. Second, because they contain active electronic components and require power, they can fail. In an office environment, the failure may occur in an inconvenient location (e.g., above the ceiling or in a wiring closet), making it difficult to find and repair.

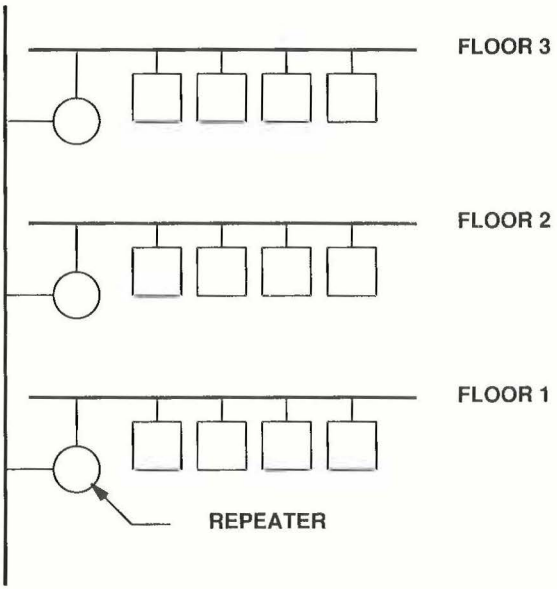


Figure 2.2 Repeaters used to join Ethernet cables in a building. At most two repeaters can be placed between a pair of communicating machines.

Connections to the ether are made by *taps* as Figure 2.3 shows. At each tap, a small hole in the outer layers of cable allows small pins to touch the center wire and the braided shield (some manufacturers' connectors require that the cable be cut and a "T" inserted). Each connection to an Ethernet has two major electronic components. A *transceiver* connects to the center wire and braided shield on the ether, sensing and sending signals on the ether. A *host interface* connects to the transceiver and communicates with the computer (usually through the computer's bus).

The transceiver is a small piece of hardware usually found physically adjacent to the ether. In addition to the analog hardware that senses and controls the ether, a transceiver contains digital circuitry that allows it to communicate with a digital computer. The transceiver can sense when the ether is in use and can translate analog electrical signals on the ether to (and from) digital form. The transceiver cable that runs between the transceiver and host interface carries power to operate the transceiver as well as signals to control its operation.

Figure 2.4 shows the interconnection between a host and a transceiver. 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 complet-

ed, and reports status information. While the transceiver is a simple hardware device, the host interface can be complex (e.g., it may contain a microprocessor used to control transfers between the computer memory and the ether).

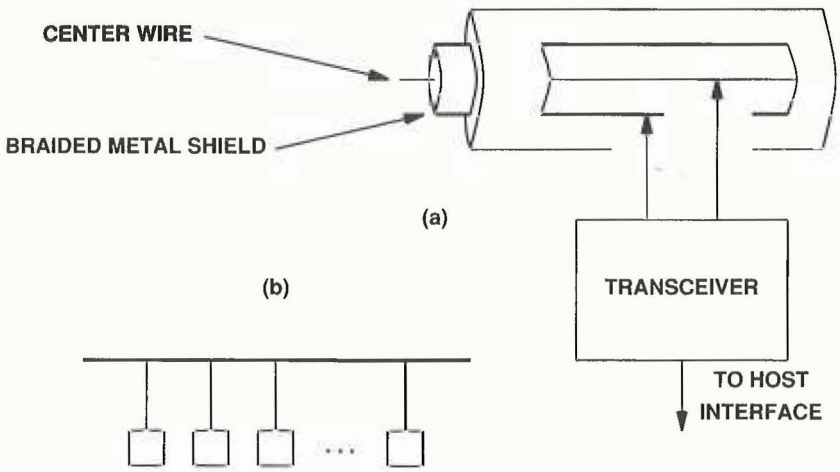


Figure 2.3 (a) A cutaway view of the cable showing the details of 2 electrical connections between a transceiver and the cable at a tap, and (b) the schematic diagram of an Ethernet with many taps.

2.4.1 Properties of an Ethernet

The Ethernet is a 10 Mbps broadcast bus technology with best-effort delivery semantics and distributed access control. It is a *bus* because all stations share a single communication channel; it is *broadcast* because all transceivers receive every transmission. 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 transceivers do not filter transmissions – they pass all packets onto the host interface, which chooses packets the host should receive and filters out all others. Ethernet is called a *best-effort delivery* mechanism because it provides no information to the sender about whether the packet was delivered. For example, if the destination machine happens to be powered down, the packet will be lost but the sender will not be notified. We will see later how the TCP/IP protocols accommodate best-effort delivery hardware.

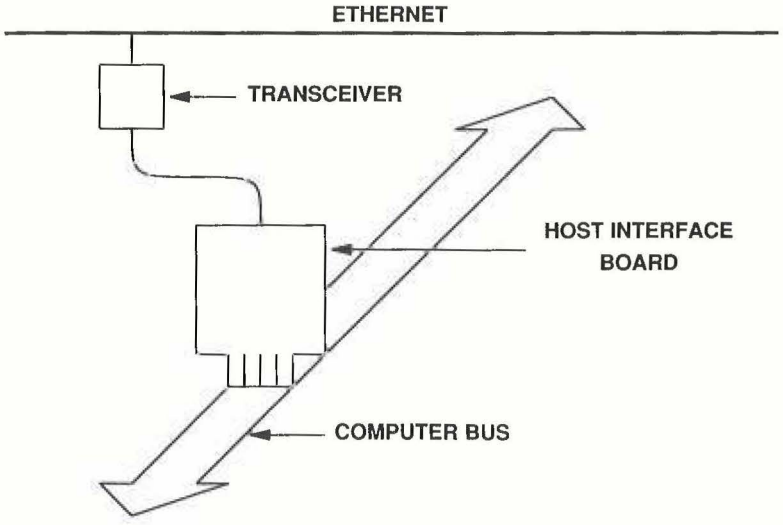


Figure 2.4 The connection between an Ethernet cable and a computer.

Ethernet access control is distributed because, unlike some network hardware, there is no central authority granting access. The Ethernet access scheme is called *Carrier Sense Multiple Access* with *Collision Detect* (*CSMA/CD*). It is *CSMA* because multiple machines can access the Ethernet simultaneously and each machine determines whether the ether is idle by sensing whether a carrier wave is present. When a host interface has a packet to transmit, it listens to the ether 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.

2.4.2 Collision Detection and Recovery

When a transceiver begins transmission, the signal does not reach all parts of the network simultaneously. Instead it travels along the cable at approximately 80% 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, such that neither is meaningful. Such incidents are called *collisions*.

The Ethernet handles collisions in an ingenious fashion. Each transceiver monitors the cable while it is transmitting to see if a foreign signal interferes with its transmission. Technically, the monitoring is called *collision detect* (*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 transceivers 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, twice as long if a second attempt to transmit also produces a collision, four times as long if a third attempt results in a collision, and so on. The idea behind 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 high probability two stations will choose random backoffs that are close together. Thus, the probability of another collision is high. By doubling 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.

2.4.3 Ethernet Capacity

The standard Ethernet is rated at 10 Mbps, which means that data can be transmitted onto the cable at 10 million bits per second. Although many recent computers 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 network total traffic capacity. Think of a network as a highway connecting multiple cities. High speeds make it possible to carry high traffic loads, while low speed 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.

2.4.4 Ethernet Variations

Recent advances in technology have made it possible to build Ethernets that do not need the electrical isolation of coaxial cable. Called *twisted pair Ethernet*, the technology allows a conventional 10 Mbps Ethernet to pass across a pair of copper wires much like the ones used to interconnect telephones. The advantage of using twisted pair is that it reduces cost and makes it possible for many groups to use existing wiring instead of adding new cable.

When high capacity is not needed, the network can still use Ethernet-like technology, but operate at slightly lower speed. The advantages are primarily economic. Lower speed means less complicated hardware and lower cost. One reason lower speed networks cost less is that the interfaces require less buffer memory and can be built from inexpensive integrated circuits.

Costs can also be reduced if high-speed digital circuits can connect directly to the cable without using a transceiver. In such cases, an Ethernet can be implemented with standard coaxial cable like that used for cable television. Called *thin-wire Ethernet*, the thin cable is inexpensive, but supports somewhat fewer connections and covers slightly shorter distances than standard Ethernet cable. Workstation manufacturers find thin

wire Ethernet an especially attractive system because they can integrate Ethernet hardware into single board computers and mount BNC-style connectors directly on the back of the machine.

Because they require no special tools, BNC connectors make it possible for users to connect workstations to Ethernets. Of course, allowing users to add their own machines to networks has disadvantages. It means that the network is susceptible to disconnection, incorrect wiring, or intentional abuse. In most situations, however, the advantages outweigh the disadvantages.

Another method of reducing costs uses a single physical cable to carry multiple, independent Ethernets. Known as *broadband*, the technology works much like broadcast radio. The transmitter multiplexes multiple Ethernets onto a single cable by assigning each Ethernet a unique frequency. Receivers must be "tuned" to the correct frequency so they receive only the desired signal and ignore others. Although the equipment needed to connect to a broadband cable is somewhat more expensive than equipment needed to connect to a conventional *baseband* cable, broadband eliminates the cost of laying multiple cables.

2.4.5 Ethernet Addressing

An Ethernet host interface provides an *addressing mechanism* that keeps unwanted packets from being passed to the host computer. Recall that each interface receives a copy of every packet – even those addressed to other machines. The hardware filters packets, ignoring those that are addressed to other machines and passing to the host only those packets addressed to it. The addressing mechanism and filter are needed to prevent a computer from being overwhelmed with incoming data.

To allow a computer to determine which packets are meant for it, each computer attached to an Ethernet is assigned a 48-bit integer known as its *Ethernet 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 Ethernet addresses belong to hardware devices, they are sometimes called *hardware addresses* or *physical 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 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 Institute for Electrical and Electronic Engineers (IEEE) manages the Ethernet address space and assigns addresses as needed.

The 48-bit Ethernet address does more than specify a single hardware interface. It can be one of three types:

- The physical address of one network interface,
- The network *broadcast* address, or
- 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 respond to a multicast address. Every computer in a multicast group can be reached simultaneously without affecting computers outside the multicast group.

To accommodate broadcast and multicast addressing, Ethernet interface hardware must recognize more than its physical address. A host interface usually accepts at least two kinds of transmissions: those addressed to the interface physical address and those addressed to the broadcast address. Some interfaces can be programmed to recognize multicast addresses or even alternate physical addresses. When the operating system starts, it initializes the Ethernet interface, giving it a set of addresses to recognize. The interface then scans each transmission, passing on to the host only those transmissions designated for one of the specified addresses.

2.4.6 Ethernet Frame Format

The 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, a frame must identify its destination. Figure 2.5 shows the Ethernet frame format that contains the physical source address as well as the physical destination address.

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 nodes 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 machine and they allow multiple protocols to be intermixed on the same physical network without interference. For example, one could have an application program using Internet protocols while another used a local experimental protocol. The operating system

†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.

‡The term *octet* refers to an 8-bit quantity, often called a *byte*.

would decide where to send incoming packets based on their frame type. We will see that the TCP/IP protocols use self-identifying Ethernet frames to distinguish among several protocols.

Preamble	Destination Address	Source Address	Frame Type	Frame Data	CRC
64 bits	48 bits	48 bits	16 bits	368-12000 bits	32 bits

Figure 2.5 The format of a frame (packet) as it travels across an Ethernet. Fields are not drawn to scale.

2.4.7 Bridges and Their Importance

We already discussed the use of Ethernet repeaters as one technique for extending a physical Ethernet to multiple physical wire segments. Although repeaters were a popular extension many years ago, most sites now use *bridges* to interconnect segments. Unlike a repeater, which replicates electrical signals, a bridge replicates packets. In fact, a bridge is a fast computer with two Ethernet interfaces and a fixed program. The bridge operates both Ethernet interfaces in *promiscuous mode*, meaning that they capture all valid packets that appear on their respective Ethernets and deliver them to the processor in the bridge. If the bridge connects two Ethernets, E_1 and E_2, the software takes each packet arriving on E_1 and transmits it on E_2, and vice versa.

Bridges are superior to repeaters because they do not replicate noise, errors, or malformed frames; a completely valid frame must be received before it will be reproduced. Furthermore, bridge interfaces follow the Ethernet CSMA/CD rules, so collisions and propagation delays on one wire remain isolated from those on the other. As a result, an (almost) arbitrary number of Ethernets can be connected together with bridges. Note that bridges hide the details of interconnection: a set of bridged segments acts like a single Ethernet. A computer can communicate across a bridge using exactly the same hardware signals it uses to communicate on its own 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 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 lists show 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 are not required to program the bridge with specific addresses. Because a bridge isolates traffic when forwarding is unnecessary, it can be used to improve the performance of an overloaded network (note that bridges work exceptionally well to partition load in a workstation environment where sets of workstations direct most of their traffic to a file 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 bridged Ethernets 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 into a cycle would produce catastrophic results because they would forward broadcast packets in both directions simultaneously.

2.5 ProNET Token Ring Technology

ProNET-10 is the name of a commercial local area network product that offers an interesting alternative to the Ethernet. Based on networking research at universities, and manufactured by Proteon Incorporated, a proNET-10 consists of a passive wiring system that interconnects computers. Like the Ethernet, the low-speed version operates at 10 Mbps†, is limited to short geographic distances, and requires attached computers to have an active host interface.

Unlike the Ethernet or related bus technologies, proNET-10 requires hosts to be wired in a one-way ring and uses an access technology known as *token passing*. The primary distinguishing feature of token-passing systems is that they achieve fair access by having all machines take turns using the network. At any time, exactly one machine

†A related Proteon product operates at 80 Mbps.

new routing tables or otherwise control the NSS. Application processors perform other tasks like network monitoring.

2.7.3 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 2.12 shows the revised connection topology, which provided redundant connections to all sites.

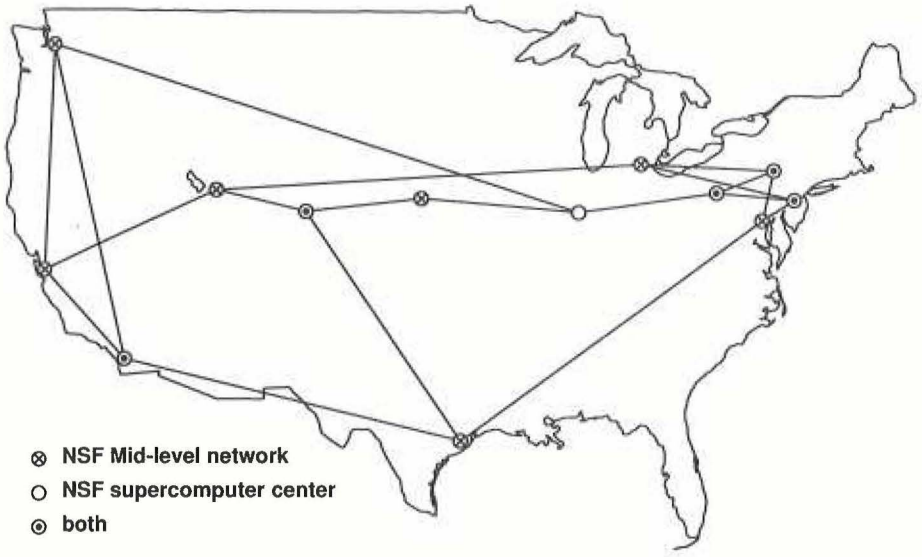


Figure 2.12 Circuits in the second NSFNET backbone from summer 1989 to 1990.

2.7.4 Multiplexing and Programmable Connections

While the exact topology of NSFNET is unimportant, the technology used to reconfigure it is. As part of their proposal, MERIT, IBM, and MCI promised to explore new ways to make the network reconfigurable. What makes their proposed plan more interesting than most network plans is that it involves MCI, the vendor supplying the long-lines connection service.

To understand the possibilities for reconfiguration, consider what usually happens when a customer contacts a long-distance vendor to lease a digital data circuit. Although the customer may imagine wires hung in a direct line between the two sites, the vendor chooses a path for the circuit that takes advantage of existing cable. For ex-

ample, the vendor may connect the customer through a local office, from there to a nearby large city where the vendor has trunk capacity, across the trunk to another large city near the destination, and finally down through a local office to the specified termination point. More important, with modern technology, the vendor does not supply a separate physical circuit. Instead, electronic equipment at one end of a trunk fiber *multiplexes* (combines) multiple circuits over the fiber and equipment at the other end *demultiplexes* (separates) them, making it possible for the vendor to add or reconfigure circuits electronically. For example, Figure 2.13 shows the location of optical fiber owned by MCI. Circuits in the NSFNET backbone are multiplexed onto this fiber.

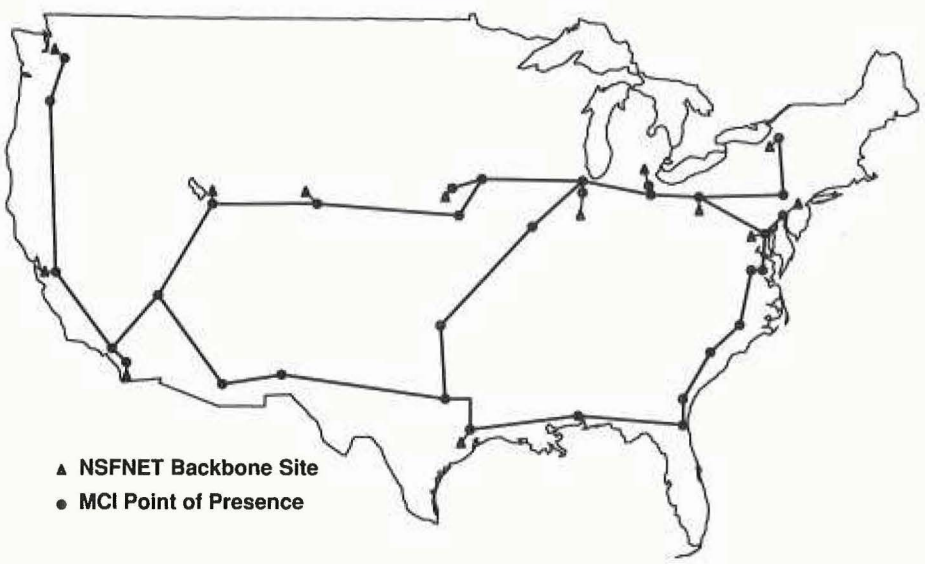


Figure 2.13 The MCI physical fiber installation over which NSFNET backbone circuits are allocated.

The MERIT/IBM/MCI proposal poses an interesting question: "If users had the ability to reconfigure circuits electronically, how could they improve networking?" The reconfiguration shown between Figures 2.11 and 2.12 illustrates one possibility. The owner of a network could watch network traffic over a long period of time and then reconfigure the circuits to provide a direct path between pairs of sites with the most traffic. In addition to adding circuits where needed, dynamic reconfiguration can allow the user to save money by eliminating costly direct paths between pairs of sites that have little or no traffic. Comparing Figures 2.11 and 2.12, we see that a direct path has

been added between the sites in Seattle and the San Francisco Bay area, while the direct path between the site at Ann Arbor, Michigan and Houston, Texas has been eliminated. Of course, one cannot reconfigure underlying circuits without recomputing routes in packet switches.

If users had access to the same reconfiguration facilities as vendors, they could do much more than merely create or delete circuits. They could adjust circuit capacities on demand. Such adjustments could be important because it could mean saving enough money on unused circuit capacity to pay for more capacity when needed. Consider NSFNET, for example. At 8 AM on the east coast, users arrive at work and begin generating traffic, so higher capacity is needed for circuits connecting to machines in the east. Meanwhile, on the west coast, most users are still asleep, so little capacity is needed for circuits that connect to west coast machines. As the day proceeds capacity should gradually shift to the west coast circuits. By late afternoon, when users are leaving their offices in the east, west coast circuits need the greatest capacity.

From the suppliers point of view, giving customers reconfigurable circuit capacity means that customers still pay for a fixed capacity in the underlying physical network, but they are free to allocate their bandwidth however they choose. Figure 2.14 illustrates the idea.

As Figure 2.14 shows, a customer who pays for capacity T in the underlying physical network can choose to divide that capacity among multiple circuits. Of course, when configuring the capacity of individual circuits, a customer must make sure that not more than capacity T is allocated at any point along the physical cable. The chief drawback of this scheme is that to make valid capacity assignments, customers must know both the topology of the physical net and the paths in that net to which their circuits have been assigned.

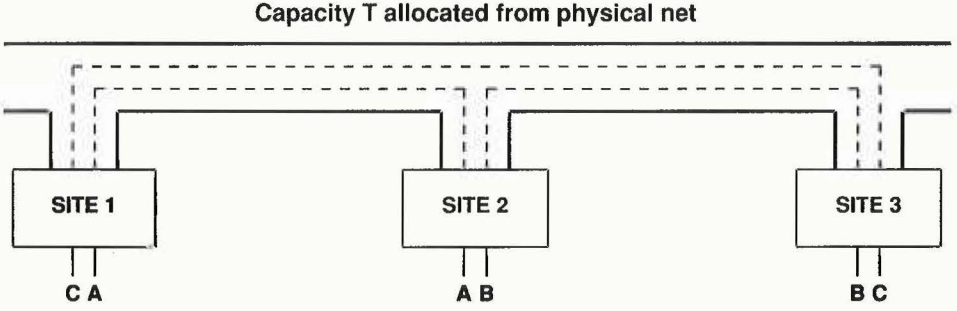


Figure 2.14 Three circuits (A, B, and C) that can be reconfigured as long as they use less than capacity T at any point in the backbone. For example, each can have capacity T/2 or, if A and B have capacity T/3, C can have capacity 2T/3.

2.7.5 NSFNET Mid-level Networks

NSF has funded many mid-level networks that span almost every state. A typical mid-level network includes 10 to 30 universities and corporations clustered in a geographic area. The original NSF goal was to cover initial costs and then encourage self-sufficiency by allowing each mid-level network to operate in fiscal and administrative autonomy. While some mid-level nets have achieved the goal of fiscal independence, others have found it difficult. Managers of mid-level networks have formed the *Federation of Academic Research Networks* (*FARNET*) to help coordinate technical work and lobby for additional government support.

Each mid-level network is free to choose whatever technology will serve it best; NSF will provide access from a mid-level net to the rest of the Internet via the NSFNET backbone. Most mid-level nets use point-to-point leased line interconnections similar to that of the NSFNET backbone; almost all plan to upgrade to higher speed lines over time.

2.7.6 NSFNET Access Networks

The family of NSF mid-level networks includes a motley collection of access networks. Some were funded as experiments using new technology (e.g., a satellite bridge), while others were funded to provide supercomputer access for a specific research individual or group. In the latter category, each supercomputer center includes a consortium of research groups that connect to it over leased lines. A consortium sometimes includes geographically distant sites, making these so-called *consortium networks* quite extensive.

2.7.7 NSFNET Campus Networks

The third tier in the NSF network family consists of campus networks that attach to mid-level nets. NSF decided to concentrate its funding on the backbone and mid-level nets, leaving universities and corporations free to choose their own local networking strategy. Most major research institutions already have a network in place at each campus; smaller corporations and schools are just beginning to consider the possibilities. The technologies used range in complexity and speed from single local area networks to complex network interconnections with backbones operating at gigabit speeds.

2.8 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.

2.8.1 X25NET

CSNET†, an organization formed in 1980 to help provide Internet services to industry and small schools, used X25NET technology to connect some of its subscribers to the Internet. Originally developed at Purdue University, X25NET runs Internet protocols over *Public Data Networks* (*PDN*s). The motivation is to allow organizations that cannot afford direct ARPANET connections to lease a network connection from a common carrier (e.g., AT&T) and use it to send Internet traffic.

Readers who know about public packet-switched networks may find X25NET strange because such networks use the CCITT X.25 protocols exclusively while the Internet uses TCP/IP protocols. When used to transport TCP/IP traffic, however, the underlying X.25 network merely provides a path over which Internet traffic can be transferred. We have already stated that many underlying technologies can be used to carry Internet traffic. The technique, sometimes called *tunneling*, simply means that a complex network with its own protocols is treated like any other hardware delivery system. To send TCP/IP traffic through an X.25 "tunnel," one makes an X.25 connection and then sends TCP/IP packets as if they were data. The X.25 system carries 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 packets like data, it does not provide for self-identifying frames. Thus, it only works when both ends of the X.25 connection agree *a priori* that they will exchange TCP/IP packets.

What makes the use of X.25 peculiar is its interface. Unlike most network hardware, X.25 protocols provide a reliable transmission stream, sometimes called a *virtual circuit*, between the sender and the receiver, while the Internet protocols have been designed for a packet delivery system, making the two (apparently) incompatible.

Viewing X.25 connections merely as delivery paths produces a strange twist. It turns out that X.25 networks exhibit substantially better throughput with multiple simultaneous connections. Thus, instead of opening a single connection to a given destination, an X25NET sender often opens multiple connections and distributes packets among them to improve performance. The receiver accepts packets from all the X.25 connections and combines them together again.

The addressing scheme used by X.25 networks is given in a related standard known as X.121. 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. It is unfortunate, however, because it makes assignment of Internet addresses difficult. Subscribers using X25NET must each maintain a table of mappings between Internet addresses and X.25 addresses.

†CSNET and BITNET have merged; the new organization is CREN.

Chapter 6 discusses the address mapping problem in detail and gives an alternative to using fixed tables.

Because public X.25 networks operate independently of the Internet, a point of contact must be provided between the two. Both DARPA and CSNET operate dedicated machines that provide the interconnection between X.25 and the ARPANET. The primary interconnection is known as the *VAN gateway*. The VAN agrees to accept X.25 connections and route incoming Internet traffic to its destination.

X25NET is significant because it illustrates the flexibility and adaptability of the TCP/IP protocols. In particular, it shows how tunneling makes it possible to use an extremely wide range of complex network technologies in an internet.

2.8.2 Cypress

Most of the network technologies we have discussed so far are expensive. But Internet access need not be limited to large institutions that connect directly to major backbones like the NSFNET; many small schools and individuals need access as well. Small institutions cannot afford high speed leased lines, or the equipment that connects to it. Cypress is designed to fill that need by providing a low cost, low volume TCP/IP technology.

Cypress consists of minicomputers interconnected by low or medium speed (9.6 Kbps to 56 Kbps) leased lines. As Figure 2.15 shows, each minicomputer resides at a subscriber's site where it connects to the local computing environment over an Ethernet local area network. It connects to the rest of Cypress over leased serial lines. At least one site on a Cypress network connects to the Internet and passes traffic between the Cypress net and the rest of the Internet.

Originally, Cypress was designed to have a ''growing vine'' topology in which each new site leased a serial line to the closest existing site. The advantage of using a vine topology is low cost; the disadvantage is delay, which becomes noticeable for traffic that passes through several intermediate machines. The Cypress topology has changed for two reasons: first, NSFNET has increased the number of potential Internet connection points dramatically, and second, most subscribers seem to be willing to pay more to avoid delays. Thus, the Cypress network evolved to a single hub located at Purdue University, where it connects to NSFNET.

Cypress is based on a few key ideas. First, to achieve low cost, Cypress consolidates functionality by using a single computer to serve several purposes. Second, like Ethernet, the Cypress protocols use best-effort delivery, with no attempt made to correct errors or recover lost packets at the link level. Later chapters will explain why best-effort delivery works well in the TCP/IP environment. Third, Cypress operates as a network, not merely as a set of point-to-point links. Fourth, Cypress connects to a network at subscriber sites, not just to a single machine. Thus, many hosts at the subscriber's site can use the Cypress connection by treating it as their path to the rest of the Internet. Fifth, Cypress allows its packet switches to be monitored from any site in the Internet because it uses IP to transport monitoring information.

The minicomputers that comprise a Cypress network are called *implets*, and each implet provides three conceptual functions in a single machine. At the lowest level, an implet operates like a packet switch, accepting packets over serial lines and routing them on to their destination using the hardware address in a frame when selecting a route. At the next level, an implet connects two networks, the local Ethernet at the subscriber's site and the Cypress network. At the highest level, an implet is a general purpose computer that executes network control and monitoring programs as user processes.

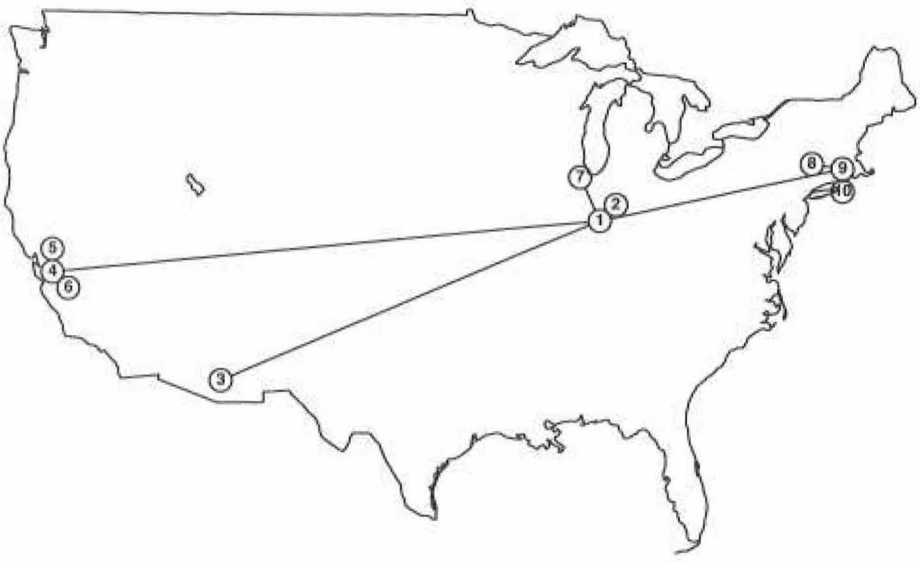


Figure 2.15 The Cypress network at its peak with sites at (1,2) Purdue, (3) Tucson AZ, (4,5,6) Palo Alto CA, (7) Chicago IL, (8) Williamstown MA, (9) Cambridge MA, and (10) Boston MA. At each site an implet connects to an Ethernet.

In addition to its technical contributions Cypress demonstrates three important ideas. First, it illustrates why network speed should be thought of as a measure of capacity. Sites with low traffic volumes perceive Cypress as an adequate, viable interconnection technology. Low speed does not mean limited functionality. Second, Cypress shows that the Internet protocols work well over a best-effort delivery system with minimum link level protocols. Third, Cypress shows that designing control and monitoring software to use the Internet protocols makes monitoring flexible and debugging easier.

2.8.3 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 use the Internet infrequently may not be able to justify leased line connections. For those sites, CSNET developed a dial-up IP system that works as expected: whenever a connection is needed, software at the member's site uses a modem to form a connection to the CSNET hub over the voice telephone network. A computer at the hub answers the phone call and, after obtaining valid authorization, forwards traffic between the site and other computers on the Internet.

2.8.4 Packet Radio

One of the most interesting DARPA experiments in packet switching resulted in a technology that used 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.

2.9 Summary And Conclusion

We have reviewed several network hardware technologies used by the TCP/IP protocols, ranging from high-speed, local area networks like Ethernet and proNET-10 to slower-speed, long haul networks like the ARPANET and Cypress. 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 in that 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] introduced the Ethernet with a 3 Mbps prototype version. Digital [1980] specifies the 10 Mbps standard adopted by most vendors, 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 was proposed in Farmer and Newhall [1969]. Miller and Thompson [1982], as well as Andrews and Shultz [1982], give recent summaries. Another alternative, the slotted ring network, was proposed by Pierce [1972]. For a comparison of technologies, see Rosenthal [1982].

Details of the proposal for the second NSFNET backbone can be found in MERIT [November 1987]. Comer, Narten and Yavatkar [1987] first suggests using the technique of building a multiprocessor packet switch around a local area network bus; apparently it was discovered independently for the second NSFNET backbone proposal.

For more information on the ARPANET see Cerf [1989] and BBN [1981]. The ideas behind X25NET are summarized in Comer and Korb [1983], while Cypress is described in Comer, Narten, and Yavatkar [April 1987]. Lanzillo and Partridge [January 1989] describes dial-up IP.

Quarterman and Hoskins [1986] provides a summary of major wide area computer networks; Quarterman [1990] contains an updated list and offers more detail. LaQuey [1990] contains a directory of computer networks.

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 NSC's Hyperchannel or Ultra Network Technologies' UltraNet?

2.3 What are the advantages and disadvantages of tunneling?

2.4 Read the Ethernet standard to find exact details of the inter-packet gap and preamble size. What is the maximum data rate Ethernet can deliver?

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

2.6 Find a lower bound on the time it takes to transfer a 5 megabyte file across a network that operates at: 9600 bps, 56 Kbps, 10 Mbps, 100 Mbps, and 2 Gbps.

3.5 Internet Architecture

We have seen how machines 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 computers that are willing to shuffle 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, machine *G* connects to both network *1* and network *2*. For *G* to act as a gateway, it must capture packets on network *1* that are bound for machines on network *2* and transfer them. Similarly, *G* must capture packets on network *2* that are destined for machines on network *1* and transfer them.

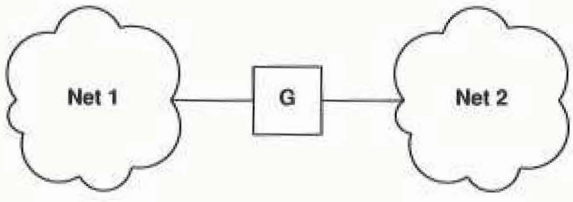


Figure 3.1 Two networks interconnected by *G*, a gateway (router).

3.6 Interconnection Through IP Gateways or Routers

When internet connections become more complex, gateways need to know about the topology of the internet beyond the networks to which they connect. For example, Figure 3.2 shows three networks interconnected by two gateways.

†The original literature used the term *gateway* but recently, vendors seem to prefer the term *IP router* – they are used interchangeably throughout this text.

Figure 3.2 Three networks interconnected by two gateways.

In this example, gateway G_1 must move from network *1* to network *2* all packets destined for machines on either network *2* or network *3*. As the size of the internet expands, the gateway's task of making decisions about where to send packets becomes more complex.

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

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

You might suspect that gateways, which must know how to route packets to their destination, are large machines with enough primary or secondary memory to hold information about every machine in the internet to which they attach. However, gateways used with TCP/IP internets are usually minicomputers; they often have little or no disk storage and limited main memories. The trick to building a small internet gateway lies in the following concept:

> *Gateways route packets based on destination network, not on destination host.*

If routing is based on networks, the amount of information that a gateway needs to keep is proportional to the number of networks in the internet, not the number of machines.

Because gateways 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 gateway in the internet. We will also assume that only gateways 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 machines independent of the particular networks to which they attach. Thus, we want the 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 gateways that interconnect physical networks, internet access software is needed on each host to allow application programs to use the internet as if it were a single, real 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 machine. Because the details of each machine's physical network connections are hidden in the internet software, only that software needs to change when new physical connections appear or old ones disappear. In fact, it is possible to optimize routing by altering physical connections without even recompiling application programs.

A second advantage of having communication at the network level is more subtle: users do not have to understand or remember how networks connect or what traffic they carry. Application programs can be written that operate 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, gateways do not provide direct connections among all pairs of networks. It may be necessary for traffic traveling from one machine to another to pass across several intermediate networks. Thus, networks participating in the 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 example 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 to depict all physical networks 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 like the NSFNET backbone, or a point-to-point link between two machines 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.

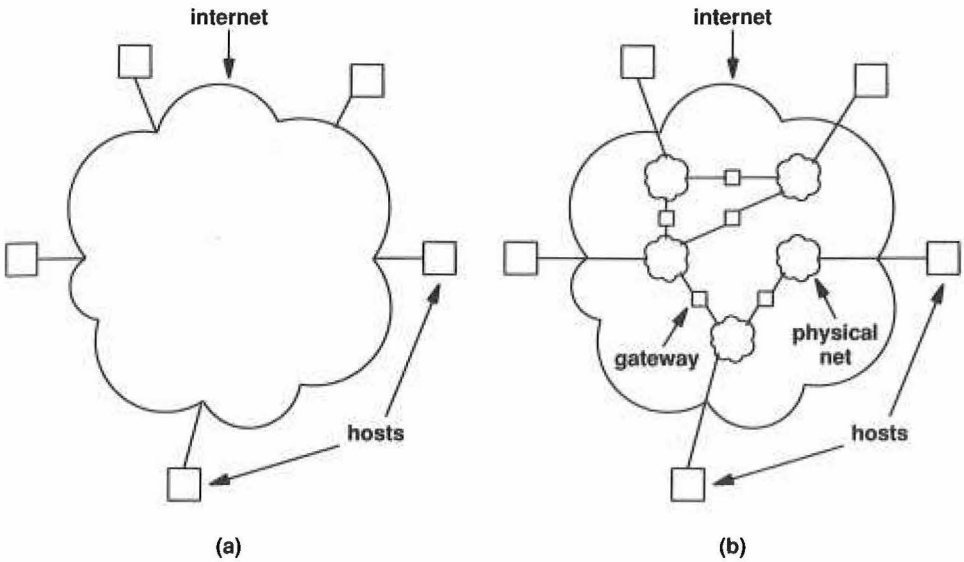


Figure 3.3 (a) The user's view of a TCP/IP internet in which each computer appears to attach to a single large network, and (b) the structure of physical networks and gateways that provide interconnection.

3.9 The Unanswered Questions

Our sketch of internets leaves many unanswered questions. For example, you might wonder about the exact form of internet machine addresses or how such addresses relate to the Ethernet, proNET-10, or ARPANET physical hardware addresses described in Chapter 2. The next three chapters confront these questions. They describe the format of IP addresses and illustrate how hosts map 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 host or gateway to handle. Chapter 7 answers these questions. Finally, you might wonder how multiple application programs executing concurrently on a single machine can send and receive packets to multiple destinations without becoming entangled in each other's transmissions or how internet gateways 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 establishing a 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 machines 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 gateways, or routers, that attach to two or more networks. Gateways route 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 gateways 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 Changing a gateway routing table can be tricky because it is impossible to change all gateways simultaneously. Investigate algorithms that guarantee to either install a change on all machines or install it on none.

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

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

3.4 What processors have been used as gateways in the connected Internet? Does the size and speed of the gateways surprise you? Why?

4

Internet Addresses

4.1 Introduction

The previous chapter defined a TCP/IP internet as a virtual network built by interconnecting physical networks with gateways. This chapter discusses addressing, an essential ingredient that helps TCP/IP software hide physical network details and makes the 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 to communicate with any other host. To make our communication system universal, we need to establish a globally accepted method of identifying computers that attach 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 better with more 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 like routing decisions 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.

4.3 Three Primary Classes Of IP Addresses

Think of an internet as a large network like any other physical network. The difference, of course, is that the internet is a virtual structure, imagined by its designers, and implemented entirely in software. Thus, 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 an 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. The bits of IP addresses for 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, each IP address must have one of the first three forms shown in Figure 4.1†.

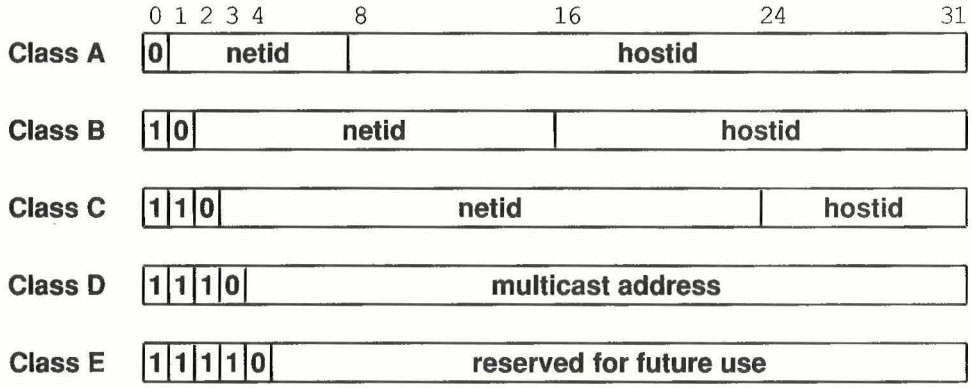


Figure 4.1 The five forms of Internet (IP) addresses. The three primary forms, Classes *A*, *B* and *C*, can be distinguished by the first two bits.

†The fourth form, reserved for internet multicasting, will be described in a later chapter; for now, we will restrict our comments to the forms that specify addresses of individual objects.

Given an IP address, its class can be determined from the three high-order bits, with two bits being sufficient to distinguish among the three primary classes. Class *A* addresses, which are 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, which are 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* networks, which have less than 2^8 hosts, allocate 21 bits to the netid and only 8 bits to the hostid. Note that the IP address has been defined in such a way that it is possible to extract the hostid or netid portions quickly. Gateways base routing on the netid and depend on such efficient extraction.

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 gateway 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 gateways 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 machine, but a connection to a network.*

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

4.5 Network And 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 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, the network address has hostid with all bits 0.*

Another significant advantage of the internet addressing scheme is that it includes a *broadcast address* that refers to all hosts on the network. According to the standard, any hostid consisting of all *1s* is reserved for broadcast†. On many network technologies (e.g., Ethernet) broadcasting can be as efficient as normal transmission; on others (e.g., Cypress) broadcasting is supported by the network software, but requires substantially more delay than single transmission. Some networks do not support broadcast at all. Thus, having an IP broadcast address does not guarantee the availability or efficiency of broadcast delivery. In summary,

> *IP addresses can be used to specify a broadcast and map to hardware broadcast if available. By convention, a broadcast address has hostid with all bits 1.*

4.6 Limited Broadcast

Technically, the broadcast address we just described is called a *directed broadcast address* 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 *1*s'' broadcast address). A host may use the limited broadcast address as part of a startup procedure before it learns its IP address or the IP address for the local network. Once the host learns the correct IP address for the local network, however, it should use 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 Chapter 16 when we discuss 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

†Unfortunately, an early release of TCP/IP code that accompanied Berkeley UNIX incorrectly used all zeroes for broadcast, and even though the Berkeley code has been repaired, the mistake still survives in some commercial systems derived from that code.

context where it can be interpreted unambiguously. For example, if a machine receives a packet in which the source address has netid set to *0* and the hostid matching its own, 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 20 will discuss in detail how a host determines its network address and how it uses network id *0*.

4.7.1 Multicast Addressing

In addition to broadcasting, the IP address scheme supports a special form of multipoint delivery known as *multicasting*. Multicasting is especially useful for networks where the hardware technology supports multicast delivery. Chapter 17 discusses multicast addressing and delivery in detail.

4.8 Weaknesses In Internet Addressing

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

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

To understand the consequences, consider travelers who wish to disconnect their personal computers, carry them along on a trip, and reconnect them to the internet after reaching their destination. The personal computer cannot be assigned a permanent IP address because an IP address identifies the network to which the machine attaches.

Another weakness of the internet 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 network id 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 multiple names 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 network 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_4. An alternate path from *A* to *B* exists through gateway *G* and is used whenever *A* sends packets to IP address I_5. 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_4 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.

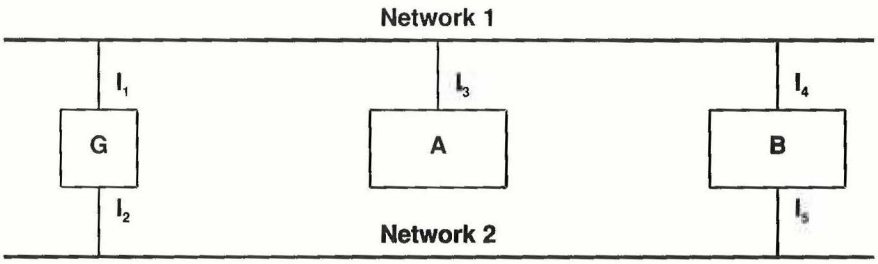


Figure 4.2 An example internet with a multi-homed host, *B*, that demonstrates a problem with the IP addressing scheme. If interface I_4 becomes disconnected, *A* must use address I_5 to reach *B*, routing packets through gateway *G*.

4.9 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

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

10000000 00001010 00000010 00011110

is written

128.10.2.30

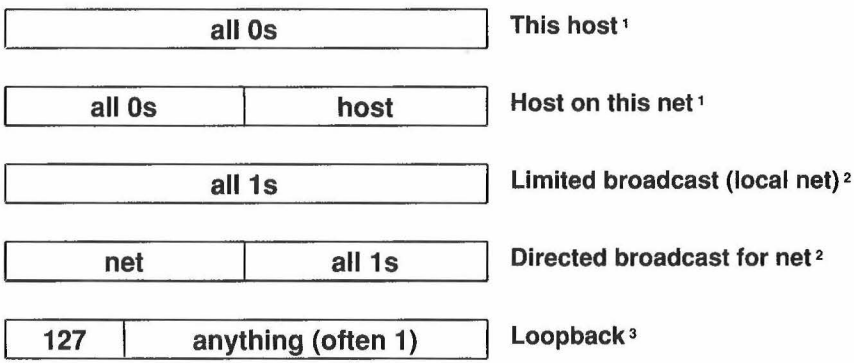
We will use dotted decimal notation when expressing IP addresses throughout the remainder of this text.

4.10 Loopback Address

The class A network address 127.0.0.0 is reserved for *loopback* and is designed for testing and inter-process communication on the local machine. When any program uses the loopback address to send data, the protocol software in the computer returns the data without sending traffic across any network. The literature explicitly states that a packet sent to a network 127 address should never appear on any network. Furthermore, a host or gateway should never propagate routing or reachability information for network number *127*; it is not a network address.

4.11 Summary Of Special Address Conventions

In practice, IP uses only a few combinations of *0*s ("this") or *1*s ("all"). Figure 4.3 lists the possibilities.



Notes: 1 Allowed only at system startup and is never a valid destination address.
2 Never a valid source address.
3 Should never appear on a network.

Figure 4.3 Special forms of IP addresses, including valid combinations of *0*s ("this"), *1*s ("all"). The length of the net portion of a directed broadcast depends on the network address class.

As the notes in the figure mention, using all *0*s for the network is only allowed during the bootstrap procedure. It allows a machine to communicate temporarily. Once the machine learns its correct network and IP address, it must not use network *0*.

4.12 Internet Addressing Authority

To insure that the network portion of an Internet addresses is unique, all Internet addresses are assigned by a central authority, the *Network Information Center* (*NIC*). The central authority only assigns the network portion of the address and delegates responsibility for assigning host addresses to the requesting organization. Local area networks with a small number of attached machines (less than 255) are usually assigned Class C numbers because many local area networks are expected. Large networks, like the ARPANET, are assigned class A numbers because only a few large networks are expected.

It is only essential for the NIC to assign IP addresses for networks that are (or will be) attached to the connected Internet. An individual corporation could take responsibility for assigning unique network addresses within its TCP/IP internet as long as it never connects that internet to the outside world. Indeed, many corporate groups that use TCP/IP protocols do assign internet addresses on their own. For example, the NIC assigned address 10.0.0.0 to the ARPANET. If a college campus decides to use TCP/IP protocols on one Ethernet with only three hosts (and no other gateway connections), that college could choose to use address 10.0.0.0 for its local network. However, experience has shown that it is unwise to create a private internet using the same network addresses as the connected Internet because it prevents future interoperability and may cause problems when trying to exchange software with other sites. Thus, everyone using TCP/IP is strongly encouraged to take the time to obtain official Internet addresses from the NIC.

4.13 An Example

To clarify the IP addressing scheme, consider the example in Figure 4.4 that shows just a few of the connections and hosts on the Internet at the Purdue University Department of Computer Science in the mid-1980s. The example shows three networks: the ARPANET (10.0.0.0), an Ethernet (128.10.0.0), and a proNET-10 token ring network (192.5.48.0). Writing out the addresses in binary shows them to be class *A*, *B*, and *C*, respectively.

In the figure, four hosts attach to these networks, labeled *Arthur*, *Merlin*, *Guenevere*, and *Lancelot*. Machine *Taliesyn* serves as a gateway between the ARPANET and the proNET-10, and machine *Glatisant* serves as a gateway between the proNET-10 and the Ethernet. Host *Merlin* has connections to both the Ethernet and the proNET-10, so it can reach hosts on either network directly. Although a multi-homed host like *Merlin* can also operate as a gateway, *Merlin* is primarily a timesharing system

and the additional work of routing packets would reduce the amount of processing available to users. Thus, a dedicated gateway, *Glatisant*, was installed to keep the gateway traffic load off the timesharing system. Traffic between these two networks was much higher than this configuration suggests because only a handful of the existing hosts are shown.

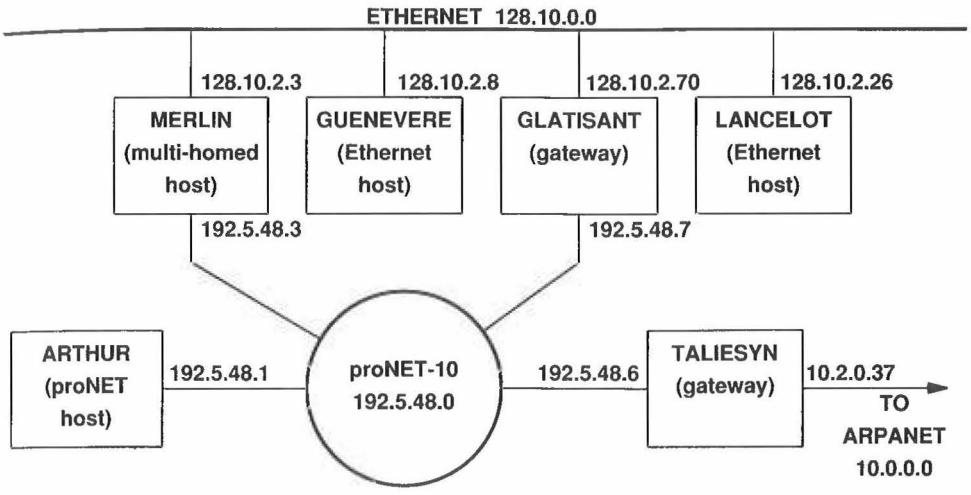


Figure 4.4 Example IP address assignments for hosts and gateways on an Ethernet, token ring network, and ARPANET.

Figure 4.4 shows the IP addresses for each network connection. *Lancelot*, which connects only to the Ethernet, has been assigned 128.10.2.26 as its only IP address. *Merlin* has address 128.10.2.3 for its connection to the Ethernet and 192.5.48.3 for its connection to the proNET-10. Choosing the same value for the low-order byte of its two addresses makes it easier for systems programmers to remember all of *Merlin's* Internet addresses.

4.14 Network Byte Order

To create an internet that is independent of any particular vendor's machine architecture or network hardware, we must define a standard representation for data. Consider what happens, for example, when one machine sends a 32-bit binary integer to another. The physical transport hardware moves the sequence of bits from the first machine to the second without changing the order. However, not all machines store 32-bit integers in the same way. On some (called *Little Endian*), the lowest memory address

contains the low-order byte of the integer. On others (called *Big Endian*), the lowest memory address holds the high-order byte of the integer. Still others store integers in groups of 16-bit words, with the lowest addresses holding the low-order word, but with bytes swapped. Thus, direct copying of bytes from one machine to another may change the value of the number.

Standardizing byte-order for integers is especially important in an internet because internet packets carry binary numbers that specify information like destination addresses and packet lengths. Such quantities must be understood by both the senders and receivers. The TCP/IP protocols solve the byte-order problem by defining a *network standard byte order* that all machines must use for binary fields in internet packets. Each host converts binary items from the local representation to network standard byte order before sending a packet; it converts from network byte order to the host-specific order when a packet is received. Naturally, the user data field in a packet is exempt from this standard – users are free to format their own data however they choose. Of course, most users rely on standard application programs and do not have to deal with the byte order problem directly.

The internet standard for byte order specifies that integers are sent most significant byte first (i.e., *Big Endian* style). If one considers the successive bytes in a packet as it travels from one machine to another, a binary integer in that packet has its most significant byte nearest the beginning of the packet and its least significant byte nearest the end of the packet. Many arguments have been offered about which data representation should be used, and the internet standard still comes under attack from time to time. However, everyone agrees that having a standard is crucial, and the exact form of the standard is far less important.

4.15 Summary

TCP/IP uses 32-bit binary addresses as universal machine identifiers. Called internet or IP addresses, the identifiers are divided into three primary classes, allowing a few hundred networks with over a million hosts each, thousands of networks with thousands of hosts each, and over a million networks with up to 254 hosts each. To make such addresses easier for humans to understand, they are written in dotted decimal notation, with the values of the four octets written in decimal, separated by decimal points.

Because the IP address encodes network identification as well as the identification of a specific host on that network, routing is efficient. An important property of IP addresses is that they refer to network connections. Hosts with multiple connections have multiple addresses. One advantage of the internet addressing scheme is that the same form of address can be used to refer to hosts, networks, and all hosts on a network (broadcast). The biggest disadvantage of the IP addressing scheme is that if a machine has multiple addresses, knowing one address may not be sufficient to reach it when some network(s) are unavailable.

To permit the exchange of binary data among machines, TCP/IP protocols enforce a standard byte ordering for integers within protocol fields. In general, a host must convert all binary data from its internal form to network standard byte order before sending a packet, and it must convert from network byte order to internal order upon receipt.

FOR FURTHER STUDY

The internet addressing scheme presented here can be found in Reynolds and Postel [RFCs 990 and 997]. Official Internet addresses are assigned by the NIC (see Appendix 1 for an address and telephone number). Chapter 16 covers an important part of the Internet address standard called *subnet addressing*. Subnet addressing allows a single network address to be used with multiple physical networks. Chapter 17 shows how Class D addresses are assigned for internet *multicast*. Cohen [1981] explains bit and byte ordering, and introduces the terms "Big Endian" and "Little Endian."

EXERCISES

4.1 Exactly how many class *A*, *B*, and *C* networks can exist? Exactly how many hosts can a network in each class have? Be careful to allow for class *D* and *E* addresses.

4.2 A machine readable list of assigned addresses is sometimes called an internet *host table*. If your site has a host table, find out how many class *A*, *B*, and *C* network numbers have been assigned.

4.3 How many hosts are attached to each of the local area networks at your site? Does your site have any local area networks for which a Class *C* address is insufficient?

4.4 What is the chief difference between the IP addressing scheme and the U.S. telephone numbering scheme?

4.5 A single central authority cannot manage to assign Internet addresses fast enough to accommodate the demand. Can you invent a scheme that allows the central authority to divide its task among several groups but still ensure that each assigned address is unique?

4.6 Does network standard byte order differ from your local machine's byte order?

Several research projects are exploring ways to find optimum segment size, but no standard currently exists.

12.14 TCP Checksum Computation

The *CHECKSUM* field in the TCP header contains a 16-bit integer checksum used to verify the integrity of the data as well as the TCP header. To compute the checksum, TCP software on the sending machine follows a procedure like the one described in Chapter 11 for UDP. It prepends a *pseudo header* to the segment, appends enough bytes containing zero to pad the segment to a multiple of 16 bits, and computes the 16-bit checksum over the entire result. TCP does not count the padded zeros in the segment length, nor does it transmit them. Also, it assumes the checksum field itself is zero for purposes of the checksum computation. As with other checksums, TCP uses 16-bit arithmetic and takes the one's complement of the one's complement sum. At the receiving site, TCP software performs the same computation to verify that the segment arrived intact.

The purpose of using a pseudo header is exactly the same as in UDP. It allows the receiver to verify that the segment has reached its correct destination, which includes both a host IP address as well as a protocol port number. Both the source and destination IP addresses are important to TCP because it must use them to identify a connection to which the segment belongs. Therefore, whenever a datagram arrives carrying a TCP segment, IP must pass to TCP the source and destination IP addresses from the datagram as well as the segment itself. Figure 12.9 shows the format of the pseudo header used in the checksum computation.

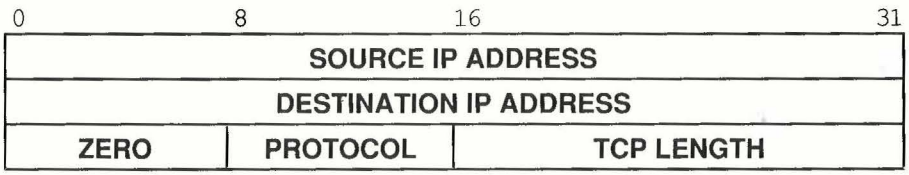


Figure 12.9 The format of the pseudo header used in TCP checksum computations. At the receiving site, this information is extracted from the IP datagram that carried the segment.

The sending TCP assigns field *PROTOCOL* the value that the underlying delivery system will use in its protocol type field. For IP datagrams carrying TCP, the value is *6*. The *TCP LENGTH* field specifies the total length of the TCP segment including the TCP header. At the receiving end, information used in the pseudo header is extracted from the IP datagram that carried the segment and included in the checksum to verify that the segment arrived at the correct destination intact.

12.15 Acknowledgements And Retransmission

Because TCP sends data in variable length segments, and because retransmitted segments can include more data than the original, acknowledgements cannot easily refer to datagrams or segments. Instead, they refer to a position in the stream using the stream sequence numbers. The receiver collects data octets from arriving segments and reconstructs an exact copy of the stream being sent. Because segments travel in IP datagrams, they can be lost or delivered out of order; the receiver uses the sequence numbers to reorder segments. At any time, the receiver will have reconstructed zero or more octets contiguously from the beginning of the stream, but may have additional pieces of the stream from datagrams that arrived out of order. The receiver always acknowledges the longest contiguous prefix of the stream that has been received correctly. Each acknowledgement specifies a sequence value one greater than the highest octet position in the contiguous prefix it received. Thus, the sender receives continuous feedback from the receiver as it progresses through the stream. We can summarize this important idea:

> *Acknowledgements always specify the sequence number of the next octet that the receiver expects to receive.*

The TCP acknowledgement scheme is called *cumulative* because it reports how much of the stream has accumulated. Cumulative acknowledgements have both advantages and disadvantages. One advantage is that acknowledgements are both easy to generate and unambiguous. Another advantage is that lost acknowledgements do not necessarily force retransmission. A major disadvantage is that the sender does not receive information about all successful transmissions, but only about a single position in the stream that has been received.

To understand why lack of information about all successful transmissions makes the protocol less efficient, think of a window that spans *5000* octets starting at position *101* in the stream, and suppose the sender has transmitted all data in the window by sending five segments. Suppose further that the first segment is lost, but all others arrive intact. The receiver continues to send acknowledgements, but they all specify octet *101*, the next highest contiguous octet it expects to receive. There is no way for the receiver to tell the sender that most of the data for the current window has arrived.

When a timeout occurs at the sender's side, the sender must choose between two potentially inefficient schemes. It may choose to retransmit all five segments instead of the one missing segment. Of course, when the retransmitted segment arrives, the receiver will have correctly received all data from the window, and will acknowledge that it expects octet *5101* next. However, that acknowledgement may not reach the sender quickly enough to prevent the unnecessary retransmission of other segments from the window. If the sender follows accepted implementation policy and retransmits only the first unacknowledged segment, it must wait for the acknowledgement before it can decide what and how much to send. Thus, it reverts to a simple positive acknowledgement protocol and may lose the advantages of having a large window.

How can TCP recover when congestion ends? You might suspect that TCP should reverse the multiplicative decrease and double the congestion window when traffic begins to flow again. However, doing so produces an unstable system that oscillates wildly between no traffic and congestion. Instead, TCP uses a technique called *slow-start*† to scale up transmission:

> *Slow-Start (Additive) Recovery: Whenever starting traffic on a new connection or increasing traffic after a period of congestion, start the congestion window at the size of a single segment and increase the congestion window by one segment each time an acknowledgement arrives.*

Slow-start avoids swamping the internet with additional traffic immediately after congestion clears or when new connections suddenly start.

The term *slow-start* may be a misnomer because under ideal conditions, the start is not very slow. TCP initializes the congestion window to *1*, sends an initial segment, and waits. When the acknowledgement arrives, it increases the congestion to *2*, sends two segments, and waits. When the two acknowledgements arrive they each increase the congestion window by *1*, so TCP can send *4* segments. Acknowledgements for those will increase the congestion window to *8*. Within four round-trip times, TCP can send *16* segments, often enough to reach the receiver's window limit. Even for extremely large windows, it takes only $\log_2 N$ round trips before TCP can send N segments.

To avoid increasing the window size too quickly and causing additional congestion, TCP adds one additional restriction. Once the congestion window reaches one half of its original size, TCP enters a *congestion avoidance* phase and slows down the rate of increment. During congestion avoidance, it increases the congestion window by *1* only if all segments in the window have been acknowledged.

Taken together, the slow-start increase, multiplicative decrease, congestion avoidance, measurement of variation, and exponential timer backoff improve the performance of TCP dramatically without adding any significant computational overhead to the protocol software. Versions of TCP that use these techniques have improved the performance of previous versions by factors of *2* to *10*.

12.21 Establishing A TCP Connection

To establish a connection, TCP uses a three-way handshake. In the simplest case, the handshake proceeds as Figure 12.11 shows.

†The term *slow-start* is attributed to John Nagle; the technique was originally called *soft-start*.

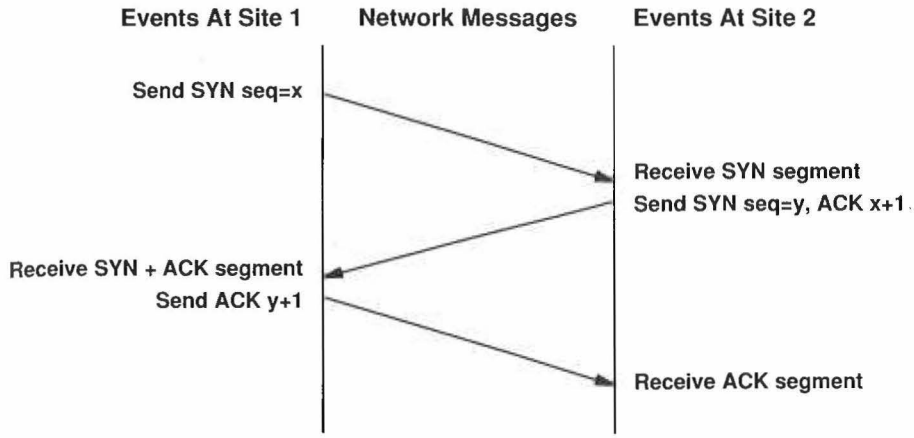


Figure 12.11 The sequence of messages in a three-way handshake. Time proceeds down the page; diagonal lines represent segments sent between sites. SYN segments carry initial sequence number information.

The first segment of a handshake can be identified because it has the SYN† bit set in the code field. The second message has both the SYN bit and ACK bits set, indicating that it acknowledges the first SYN segment as well as continuing the handshake. The final handshake message is only an acknowledgement and is merely used to inform the destination that both sides agree that a connection has been established.

Usually, the TCP software on one machine waits passively for the handshake, and the TCP software on another machine initiates it. However, the handshake is carefully designed to work even if both machines attempt to initiate a connection simultaneously. Thus, a connection can be established from either end or from both ends simultaneously. Once the connection has been established, data can flow in both directions equally well. There is no master or slave.

The three-way handshake is both necessary and sufficient for correct synchronization between the two ends of the connection. To understand why, remember that TCP builds on an unreliable packet delivery service, so messages can be lost, delayed, duplicated, or delivered out of order. Thus, the protocol must use a timeout mechanism and retransmit lost requests. Trouble arises if retransmitted and original requests arrive while the connection is being established, or if retransmitted requests are delayed until after a connection has been established, used, and terminated. A three-way handshake (plus the rule that TCP ignores additional requests for connection after a connection has been established) solves these problems.

†SYN stands for *synchronization*; it is pronounced "sin".

12.22 Initial Sequence Numbers

The three-way handshake accomplishes two important functions. It guarantees that both sides are ready to transfer data (and that they know they are both ready), and it allows both sides to agree on initial sequence numbers. Sequence numbers are sent and acknowledged during the handshake. Each machine must choose an initial sequence number at random that it will use to identify bytes in the stream it is sending. Sequence numbers cannot always start at the same value. In particular, TCP cannot merely choose sequence *1* every time it creates a connection (one of the exercises examines problems that can arise if it does). Of course, it is important that both sides agree on an initial number, so octet numbers used in acknowledgements agree with those used in data segments.

To see how machines can agree on sequence numbers for two streams after only three messages, recall that each segment contains both a sequence number field and an acknowledgement field. The machine that initiates a handshake, call it *A*, passes its initial sequence number, *x*, in the sequence field of the first SYN segment in the three-way handshake. The second machine, *B*, receives the SYN, records the sequence number, and replies by sending its initial sequence number in the sequence field as well as an acknowledgement that specifies *B* expects octet *x+1*. In the final message of the handshake, *A* ''acknowledges'' receiving from *B* all octets through *y*. In all cases, acknowledgements follow the convention of using the number of the *next* octet expected.

We have described how TCP usually carries out the three-way handshake by exchanging segments that contain a minimum amount of information. Because of the protocol design, it is possible to send data along with the initial sequence numbers in the handshake segments. In such cases, the TCP software must hold the data until the handshake completes. Once a connection has been established, the TCP software can release data being held and deliver it to a waiting application program quickly. The reader is referred to the protocol specification for the details.

12.23 Closing a TCP Connection

Two programs that use TCP to communicate can terminate the conversation gracefully using the *close* operation. Internally, TCP uses a modified three-way handshake to close connections. Recall that TCP connections are full duplex and that we view them as containing two independent stream transfers, one going in each direction. When an application program tells TCP that it has no more data to send, TCP will close the connection *in one direction*. To close its half of a connection, the sending TCP finishes transmitting the remaining data, waits for the receiver to acknowledge it, and then sends a segment with the FIN bit set. The receiving TCP acknowledges the FIN segment and informs the application program on its end that no more data is available (e.g., using the operating system's end-of-file mechanism).

Once a connection has been closed in a given direction, TCP refuses to accept more data for that direction. Meanwhile, data can continue to flow in the opposite direction until the sender closes it. Of course, acknowledgements continue to flow back to the sender even after a connection has been closed. When both directions have been closed, the TCP software at each endpoint deletes its record of the connection.

The details of closing a connection are even more subtle than suggested above because TCP uses a modified three-way handshake to close a connection. Figure 12.12 illustrates the procedure.

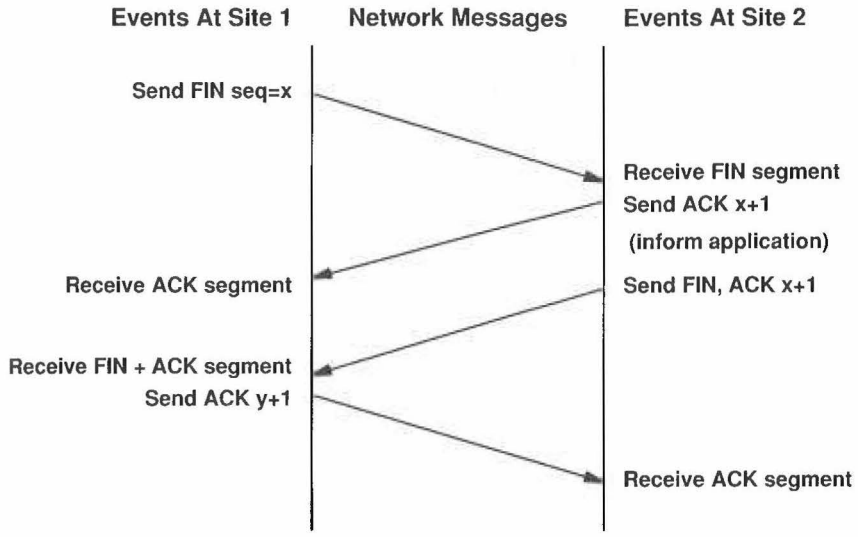


Figure 12.12 The modified three-way handshake used to close connections. The site that receives the first FIN segment acknowledges it immediately and then delays before sending the second FIN segment.

The difference between three-way handshakes used to establish and break connections occurs after a machine receives the initial FIN segment. Instead of generating a second FIN segment immediately, TCP sends an acknowledgement and then informs the application of the request to shut down. Informing the application program of the request and obtaining a response may take considerable time (e.g., it may involve human interaction). The acknowledgement prevents retransmission of the initial FIN segment during the wait. Finally, when the application program instructs TCP to shut down the connection completely, TCP sends the second FIN segment and the original site replies with the third message, an ACK.