FOURTH EDITION

Computer Networks

ANDREW S. TANENBAUM

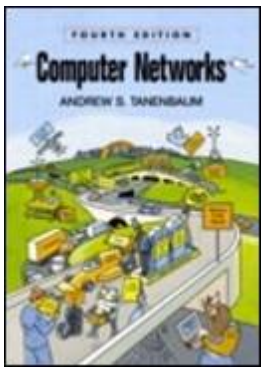


Table of Contents

Computer Networks, Fourth Edition

By Andrew S. Tanenbaum

START READING

Publisher: Prentice Hall
Pub Date: March 17, 2003
ISBN: 0-13-066102-3
Pages: 384

The world's leading introduction to networking-fully updated for tomorrow's key technologies.

Computer Networks, Fourth Edition is the ideal introduction to today's networks-and tomorrow's. This classic best seller has been thoroughly updated to reflect the newest and most important networking technologies with a special emphasis on wireless networking, including 802.11, Bluetooth, broadband wireless, ad hoc networks, i-mode, and WAP. But fixed networks have not been ignored either with coverage of ADSL, gigabit Ethernet, peer-to-peer networks, NAT, and MPLS. And there is lots of new material on applications, including over 60 pages on the Web, plus Internet radio, voice over IP, and video on demand.Finally, the coverage of network security has been revised and expanded to fill an entire chapter.

Author, educator, and researcher Andrew S. Tanenbaum, winner of the ACM Karl V. Karlstrom Outstanding Educator Award, carefully explains how networks work on the inside, from underlying hardware at the physical layer up through the top-level application layer. Tanenbaum covers all this and more:

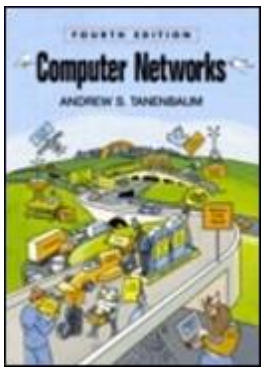
-
- Physical layer (e.g., copper, fiber, wireless, satellites, and Internet over cable)
-
- Data link layer (e.g., protocol principles, protocol verification, HDLC, and PPP)
-
- MAC Sublayer (e.g., gigabit Ethernet, 802.11, broadband wireless, and switching)
-
- Network layer (e.g., routing algorithms, congestion control, QoS, IPv4, and IPv6)

-
- Transport layer (e.g., socket programming, UDP, TCP, RTP, and network performance)
-
- Application layer (e.g., e-mail, the Web, PHP, wireless Web, MP3, and streaming audio)
-
- Network security (e.g., AES, RSA, quantum cryptography, IPsec, and Web security)

The book gives detailed descriptions of the principles associated with each layer and presents many examples drawn from the Internet and wireless networks.

[Team LiB]

NEXT



Computer Networks, Fourth Edition

By Andrew S. Tanenbaum

START READING

Publisher: Prentice Hall
Pub Date: March 17, 2003
ISBN: 0-13-066102-3
Pages: 384

Copyright

Editorial/production supervision: Patti Guerrieri

Cover design director: Jerry Votta

Cover designer: Anthony Gemmellaro

Cover design: Andrew S. Tanenbaum

Art director: Gail Cocker-Bogusz

Interior Design: Andrew S. Tanenbaum

Interior graphics: Hadel Studio

Typesetting: Andrew S. Tanenbaum

Manufacturing buyer: Maura Zaldivar

Executive editor: Mary Franz

Editorial assistant: Noreen Regina

Marketing manager: Dan DePasquale

Publishing as Prentice Hall PTR

Upper Saddle River, New Jersey 07458

Other bestselling titles by Andrew S. Tanenbaum

Distributed Systems: Principles and Paradigms

This new book, co-authored with Maarten van Steen, covers both the principles and paradigms of modern distributed systems. In the first part, it covers the principles of communication, processes, naming, synchronization, consistency and replication, fault tolerance, and security in detail. Then in the second part, it goes into different paradigms used to build distributed systems, including object-based systems, distributed file systems, document-based systems, and coordination-based systems. Numerous examples are discussed at length.

Modern Operating Systems, 2nd edition

This comprehensive text covers the principles of modern operating systems in detail and illustrates them with numerous real-world examples. After an introductory chapter, the next five chapters deal with the basic concepts: processes and threads, deadlocks, memory management, input/output, and file systems. The next six chapters deal with more advanced material, including multimedia systems, multiple processor systems, security. Finally, two detailed case studies are given: UNIX/Linux and Windows 2000.

Structured Computer Organization, 4th edition

This widely-read classic, now in its fourth edition, provides the ideal introduction to computer architecture. It covers the topic in an easy-to-understand way, bottom up. There is a chapter on digital logic for beginners, followed by chapters on microarchitecture, the instruction set architecture level, operating systems, assembly language, and parallel computer architectures.

Operating Systems: Design and Implementation, 2nd edition

This popular text on operating systems, co-authored with Albert S. Woodhull, is the only book covering both the principles of operating systems and their application to a real system. All the traditional operating systems topics are covered in detail. In addition, the principles are carefully illustrated with MINIX, a free POSIX-based UNIX-like operating system for personal computers. Each book contains a free CD-ROM containing the complete MINIX system, including all the source code. The source code is listed in an appendix to the book and explained in detail in the text.

Preface

This book is now in its fourth edition. Each edition has corresponded to a different phase in the way computer networks were used. When the first edition appeared in 1980, networks were an academic curiosity. When the second edition appeared in 1988, networks were used by universities and large businesses. When the third edition appeared in 1996, computer networks, especially the Internet, had become a daily reality for millions of people. The new item in the fourth edition is the rapid growth of wireless networking in many forms.

The networking picture has changed radically since the third edition. In the mid-1990s, numerous kinds of LANs and WANs existed, along with multiple protocol stacks. By 2003, the only wired LAN in widespread use was Ethernet, and virtually all WANs were on the Internet. Accordingly, a large amount of material about these older networks has been removed.

However, new developments are also plentiful. The most important is the huge increase in wireless networks, including 802.11, wireless local loops, 2G and 3G cellular networks, Bluetooth, WAP, i-mode, and others. Accordingly, a large amount of material has been added on wireless networks. Another newly-important topic is security, so a whole chapter on it has been added.

Although Chap. 1 has the same introductory function as it did in the third edition, the contents have been revised and brought up to date. For example, introductions to the Internet, Ethernet, and wireless LANs are given there, along with some history and background. Home networking is also discussed briefly.

Chapter 2 has been reorganized somewhat. After a brief introduction to the principles of data communication, there are three major sections on transmission (guided media, wireless, and satellite), followed by three more on important examples (the public switched telephone system, the mobile telephone system, and cable television). Among the new topics covered in this chapter are ADSL, broadband wireless, wireless MANs, and Internet access over cable and DOCSIS.

Chapter 3 has always dealt with the fundamental principles of point-to-point protocols. These ideas are essentially timeless and have not changed for decades. Accordingly, the series of detailed example protocols presented in this chapter is largely unchanged from the third edition.

In contrast, the MAC sublayer has been an area of great activity in recent years, so many changes are present in Chap. 4. The section on Ethernet has been expanded to include gigabit Ethernet. Completely new are major sections on wireless LANs, broadband wireless, Bluetooth, and data link layer switching, including MPLS.

Chapter 5 has also been updated, with the removal of all the ATM material and the addition of additional material on the Internet. Quality of service is now also a major topic, including discussions of integrated services and differentiated services. Wireless networks are also present here, with a discussion of routing in ad hoc networks. Other new topics include NAT and peer-to-peer networks.

Chap. 6 is still about the transport layer, but here, too, some changes have occurred. Among these is an example of socket programming. A one-page client and a one-page server are given in C and discussed. These programs, available on the book's Web site, can be compiled and run. Together they provide a primitive remote file or Web server available for experimentation. Other new topics include remote procedure call, RTP, and transaction/TCP.

Chap. 7, on the application layer, has been more sharply focused. After a short introduction to DNS, the rest of the chapter deals with just three topics: e-mail, the Web, and multimedia. But each topic is treated in great detail. The discussion of how the Web works is now over 60 pages, covering a vast array of topics, including static and dynamic Web pages, HTTP, CGI scripts, content delivery networks, cookies, and Web caching. Material is also present on how modern Web pages are written, including brief introductions to XML, XSL, XHTML, PHP, and more, all with

About the Author

Andrew S. Tanenbaum has an S.B. degree from M.I.T. and a Ph.D. from the University of California at Berkeley. He is currently a Professor of Computer Science at the Vrije Universiteit in Amsterdam, The Netherlands, where he heads the Computer Systems Group. He is also Dean of the Advanced School for Computing and Imaging, an interuniversity graduate school doing research on advanced parallel, distributed, and imaging systems. Nevertheless, he is trying very hard to avoid turning into a bureaucrat.

In the past, he has done research on compilers, operating systems, networking, and local-area distributed systems. His current research focuses primarily on the design and implementation of wide-area distributed systems that scales to a billion users. This research, being done together with Prof. Maarten van Steen, is described at www.cs.vu.nl/globe. Together, all these research projects have led to over 100 refereed papers in journals and conference proceedings and five books.

Prof. Tanenbaum has also produced a considerable volume of software. He was the principal architect of the Amsterdam Compiler Kit, a widely-used toolkit for writing portable compilers, as well as of MINIX, a small UNIX clone intended for use in student programming labs. This system provided the inspiration and base on which Linux was developed. Together with his Ph.D. students and programmers, he helped design the Amoeba distributed operating system, a high-performance microkernel-based distributed operating system. The MINIX and Amoeba systems are now available for free via the Internet.

His Ph.D. students have gone on to greater glory after getting their degrees. He is very proud of them. In this respect he resembles a mother hen.

Prof. Tanenbaum is a Fellow of the ACM, a Fellow of the the IEEE, and a member of the Royal Netherlands Academy of Arts and Sciences. He is also winner of the 1994 ACM Karl V. Karlstrom Outstanding Educator Award, winner of the 1997 ACM/SIGCSE Award for Outstanding Contributions to Computer Science Education, and winner of the 2002 Texty award for excellence in textbooks. He is also listed in Who's Who in the World. His home page on the World Wide Web can be found at URL http://www.cs.vu.nl/~ast/ .

Chapter 1. Introduction

Each of the past three centuries has been dominated by a single technology. The 18th century was the era of the great mechanical systems accompanying the Industrial Revolution. The 19th century was the age of the steam engine. During the 20th century, the key technology was information gathering, processing, and distribution. Among other developments, we saw the installation of worldwide telephone networks, the invention of radio and television, the birth and unprecedented growth of the computer industry, and the launching of communication satellites.

As a result of rapid technological progress, these areas are rapidly converging and the differences between collecting, transporting, storing, and processing information are quickly disappearing. Organizations with hundreds of offices spread over a wide geographical area routinely expect to be able to examine the current status of even their most remote outpost at the push of a button. As our ability to gather, process, and distribute information grows, the demand for ever more sophisticated information processing grows even faster.

Although the computer industry is still young compared to other industries (e.g., automobiles and air transportation), computers have made spectacular progress in a short time. During the first two decades of their existence, computer systems were highly centralized, usually within a single large room. Not infrequently, this room had glass walls, through which visitors could gawk at the great electronic wonder inside. A medium-sized company or university might have had one or two computers, while large institutions had at most a few dozen. The idea that within twenty years equally powerful computers smaller than postage stamps would be mass produced by the millions was pure science fiction.

The merging of computers and communications has had a profound influence on the way computer systems are organized. The concept of the "computer center" as a room with a large computer to which users bring their work for processing is now totally obsolete. The old model of a single computer serving all of the organization's computational needs has been replaced by one in which a large number of separate but interconnected computers do the job. These systems are called computer networks. The design and organization of these networks are the subjects of this book.

Throughout the book we will use the term "computer network" to mean a collection of autonomous computers interconnected by a single technology. Two computers are said to be interconnected if they are able to exchange information. The connection need not be via a copper wire; fiber optics, microwaves, infrared, and communication satellites can also be used. Networks come in many sizes, shapes and forms, as we will see later. Although it may sound strange to some people, neither the Internet nor the World Wide Web is a computer network. By the end of this book, it should be clear why. The quick answer is: the Internet is not a single network but a network of networks and the Web is a distributed system that runs on top of the Internet.

There is considerable confusion in the literature between a computer network and a distributed system. The key distinction is that in a distributed system, a collection of independent computers appears to its users as a single coherent system. Usually, it has a single model or paradigm that it presents to the users. Often a layer of software on top of the operating system, called middleware, is responsible for implementing this model. A well-known example of a distributed system is the World Wide Web, in which everything looks like a document (Web page).

In a computer network, this coherence, model, and software are absent. Users are exposed to the actual machines, without any attempt by the system to make the machines look and act in a coherent way. If the machines have different hardware and different operating systems, that is fully visible to the users. If a user wants to run a program

1.1 Uses of Computer Networks

Before we start to examine the technical issues in detail, it is worth devoting some time to pointing out why people are interested in computer networks and what they can be used for. After all, if nobody were interested in computer networks, few of them would be built. We will start with traditional uses at companies and for individuals and then move on to recent developments regarding mobile users and home networking.

1.1.1 Business Applications

Many companies have a substantial number of computers. For example, a company may have separate computers to monitor production, keep track of inventories, and do the payroll. Initially, each of these computers may have worked in isolation from the others, but at some point, management may have decided to connect them to be able to extract and correlate information about the entire company.

Put in slightly more general form, the issue here is resource sharing, and the goal is to make all programs, equipment, and especially data available to anyone on the network without regard to the physical location of the resource and the user. An obvious and widespread example is having a group of office workers share a common printer. None of the individuals really needs a private printer, and a high-volume networked printer is often cheaper, faster, and easier to maintain than a large collection of individual printers.

However, probably even more important than sharing physical resources such as printers, scanners, and CD burners, is sharing information. Every large and medium-sized company and many small companies are vitally dependent on computerized information. Most companies have customer records, inventories, accounts receivable, financial statements, tax information, and much more online. If all of its computers went down, a bank could not last more than five minutes. A modern manufacturing plant, with a computer-controlled assembly line, would not last even that long. Even a small travel agency or three-person law firm is now highly dependent on computer networks for allowing employees to access relevant information and documents instantly.

For smaller companies, all the computers are likely to be in a single office or perhaps a single building, but for larger ones, the computers and employees may be scattered over dozens of offices and plants in many countries. Nevertheless, a sales person in New York might sometimes need access to a product inventory database in Singapore. In other words, the mere fact that a user happens to be 15,000 km away from his data should not prevent him from using the data as though they were local. This goal may be summarized by saying that it is an attempt to end the "tyranny of geography."

In the simplest of terms, one can imagine a company's information system as consisting of one or more databases and some number of employees who need to access them remotely. In this model, the data are stored on powerful computers called servers. Often these are centrally housed and maintained by a system administrator. In contrast, the employees have simpler machines, called clients, on their desks, with which they access remote data, for example, to include in spreadsheets they are constructing. (Sometimes we will refer to the human user of the client machine as the "client," but it should be clear from the context whether we mean the computer or its user.) The client and server machines are connected by a network, as illustrated in Fig. 1-1. Note that we have shown the network as a simple oval, without any detail. We will use this form when we mean a network in the abstract sense. When more detail is required, it will be provided.

1.2 Network Hardware

It is now time to turn our attention from the applications and social aspects of networking (the fun stuff) to the technical issues involved in network design (the work stuff). There is no generally accepted taxonomy into which all computer networks fit, but two dimensions stand out as important: transmission technology and scale. We will now examine each of these in turn.

Broadly speaking, there are two types of transmission technology that are in widespread use. They are as follows:

1.

1. Broadcast links.
2.

2. Point-to-point links.

Broadcast networks have a single communication channel that is shared by all the machines on the network. Short messages, called packets in certain contexts, sent by any machine are received by all the others. An address field within the packet specifies the intended recipient. Upon receiving a packet, a machine checks the address field. If the packet is intended for the receiving machine, that machine processes the packet; if the packet is intended for some other machine, it is just ignored.

As an analogy, consider someone standing at the end of a corridor with many rooms off it and shouting "Watson, come here. I want you." Although the packet may actually be received (heard) by many people, only Watson responds. The others just ignore it. Another analogy is an airport announcement asking all flight 644 passengers to report to gate 12 for immediate boarding.

Broadcast systems generally also allow the possibility of addressing a packet to all destinations by using a special code in the address field. When a packet with this code is transmitted, it is received and processed by every machine on the network. This mode of operation is called broadcasting. Some broadcast systems also support transmission to a subset of the machines, something known as multicasting. One possible scheme is to reserve one bit to indicate multicasting. The remaining n - 1 address bits can hold a group number. Each machine can "subscribe" to any or all of the groups. When a packet is sent to a certain group, it is delivered to all machines subscribing to that group.

In contrast, point-to-point networks consist of many connections between individual pairs of machines. To go from the source to the destination, a packet on this type of network may have to first visit one or more intermediate machines. Often multiple routes, of different lengths, are possible, so finding good ones is important in point-to-point networks. As a general rule (although there are many exceptions), smaller, geographically localized networks tend to use broadcasting, whereas larger networks usually are point-to-point. Point-to-point transmission with one sender and one receiver is sometimes called unicasting.

An alternative criterion for classifying networks is their scale. In Fig. 1-6 we classify multiple processor systems by their physical size. At the top are the personal area networks, networks that are meant for one person. For example, a wireless network connecting a computer with its mouse, keyboard, and printer is a personal area network. Also, a PDA that controls the user's hearing aid or pacemaker fits in this category. Beyond the personal area networks come longer-range networks. These can be divided into local, metropolitan, and wide area networks. Finally, the

1.3 Network Software

The first computer networks were designed with the hardware as the main concern and the software as an afterthought. This strategy no longer works. Network software is now highly structured. In the following sections we examine the software structuring technique in some detail. The method described here forms the keystone of the entire book and will occur repeatedly later on.

1.3.1 Protocol Hierarchies

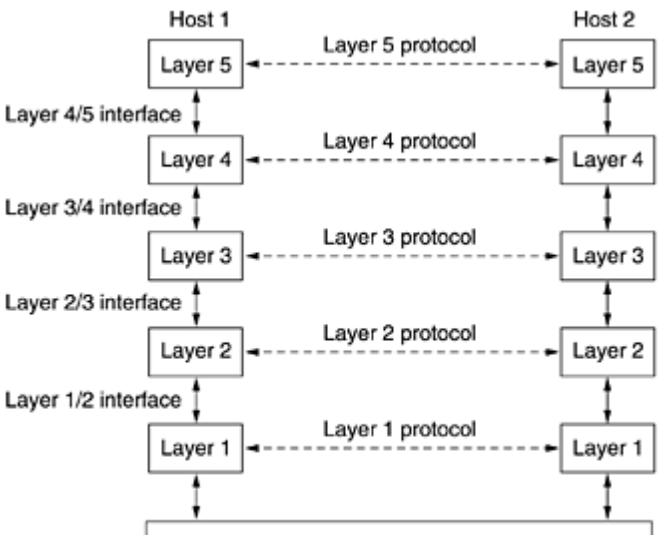
To reduce their design complexity, most networks are organized as a stack of layers or levels, each one built upon the one below it. The number of layers, the name of each layer, the contents of each layer, and the function of each layer differ from network to network. The purpose of each layer is to offer certain services to the higher layers, shielding those layers from the details of how the offered services are actually implemented. In a sense, each layer is a kind of virtual machine, offering certain services to the layer above it.

This concept is actually a familiar one and used throughout computer science, where it is variously known as information hiding, abstract data types, data encapsulation, and object-oriented programming. The fundamental idea is that a particular piece of software (or hardware) provides a service to its users but keeps the details of its internal state and algorithms hidden from them.

Layer n on one machine carries on a conversation with layer n on another machine. The rules and conventions used in this conversation are collectively known as the layer n protocol. Basically, a protocol is an agreement between the communicating parties on how communication is to proceed. As an analogy, when a woman is introduced to a man, she may choose to stick out her hand. He, in turn, may decide either to shake it or kiss it, depending, for example, on whether she is an American lawyer at a business meeting or a European princess at a formal ball. Violating the protocol will make communication more difficult, if not completely impossible.

A five-layer network is illustrated in Fig. 1-13. The entities comprising the corresponding layers on different machines are called peers. The peers may be processes, hardware devices, or even human beings. In other words, it is the peers that communicate by using the protocol.

Figure 1-13. Layers, protocols, and interfaces.



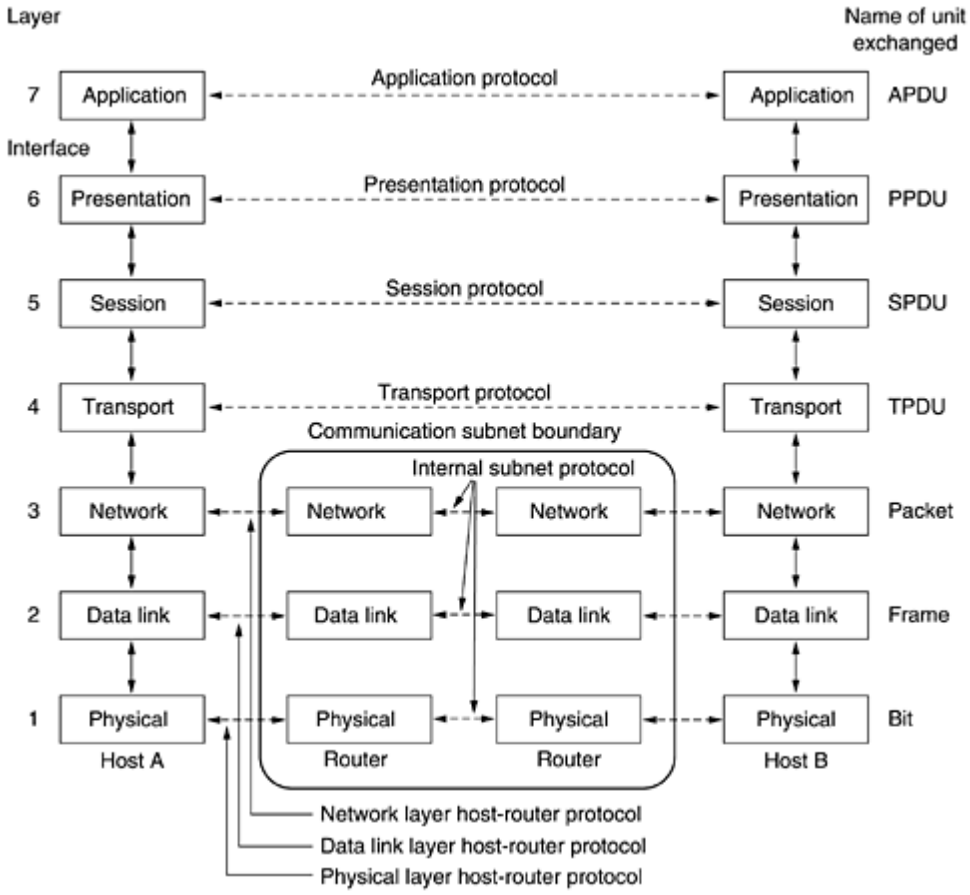
1.4 Reference Models

Now that we have discussed layered networks in the abstract, it is time to look at some examples. In the next two sections we will discuss two important network architectures, the OSI reference model and the TCP/IP reference model. Although the protocols associated with the OSI model are rarely used any more, the model itself is actually quite general and still valid, and the features discussed at each layer are still very important. The TCP/IP model has the opposite properties: the model itself is not of much use but the protocols are widely used. For this reason we will look at both of them in detail. Also, sometimes you can learn more from failures than from successes.

1.4.1 The OSI Reference Model

The OSI model (minus the physical medium) is shown in Fig. 1-20. This model is based on a proposal developed by the International Standards Organization (ISO) as a first step toward international standardization of the protocols used in the various layers (Day and Zimmermann, 1983). It was revised in 1995 (Day, 1995). The model is called the ISO OSI (Open Systems Interconnection) Reference Model because it deals with connecting open systems—that is, systems that are open for communication with other systems. We will just call it the OSI model for short.

Figure 1-20. The OSI reference model.



The OSI model has seven layers. The principles that were applied to arrive at the seven layers can be briefly summarized as follows:

1.

1. A layer should be created where a different abstraction is needed

In addition to being incomprehensible, another problem with OSI is that some functions, such as addressing, flow control, and error control, reappear again and again in each layer. Saltzer et al. (1984), for example, have pointed out that to be effective, error control must be done in the highest layer, so that repeating it over and over in each of the lower layers is often unnecessary and inefficient.

Bad Implementations

Given the enormous complexity of the model and the protocols, it will come as no surprise that the initial implementations were huge, unwieldy, and slow. Everyone who tried them got burned. It did not take long for people to associate "OSI" with "poor quality." Although the products improved in the course of time, the image stuck.

In contrast, one of the first implementations of TCP/IP was part of Berkeley UNIX and was quite good (not to mention, free). People began using it quickly, which led to a large user community, which led to improvements, which led to an even larger community. Here the spiral was upward instead of downward.

Bad Politics

On account of the initial implementation, many people, especially in academia, thought of TCP/IP as part of UNIX, and UNIX in the 1980s in academia was not unlike parenthood (then incorrectly called motherhood) and apple pie.

OSI, on the other hand, was widely thought to be the creature of the European telecommunication ministries, the European Community, and later the U.S. Government. This belief was only partly true, but the very idea of a bunch of government bureaucrats trying to shove a technically inferior standard down the throats of the poor researchers and programmers down in the trenches actually developing computer networks did not help much. Some people viewed this development in the same light as IBM announcing in the 1960s that PL/I was the language of the future, or DoD correcting this later by announcing that it was actually Ada.

1.4.5 A Critique of the TCP/IP Reference Model

The TCP/IP model and protocols have their problems too. First, the model does not clearly distinguish the concepts of service, interface, and protocol. Good software engineering practice requires differentiating between the specification and the implementation, something that OSI does very carefully, and TCP/IP does not. Consequently, the TCP/IP model is not much of a guide for designing new networks using new technologies.

Second, the TCP/IP model is not at all general and is poorly suited to describing any protocol stack other than TCP/IP. Trying to use the TCP/IP model to describe Bluetooth, for example, is completely impossible.

Third, the host-to-network layer is not really a layer at all in the normal sense of the term as used in the context of layered protocols. It is an interface (between the network and data link layers). The distinction between an interface and a layer is crucial, and one should not be sloppy about it.

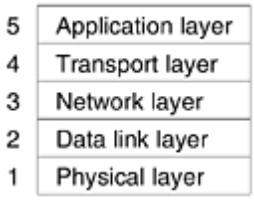
Fourth, the TCP/IP model does not distinguish (or even mention) the physical and data link layers. These are completely different. The physical layer has to do with the transmission characteristics of copper wire, fiber optics,

and wireless communication. The data link layer's job is to delimit the start and end of frames and get them from one side to the other with the desired degree of reliability. A proper model should include both as separate layers. The TCP/IP model does not do this.

Finally, although the IP and TCP protocols were carefully thought out and well implemented, many of the other protocols were ad hoc, generally produced by a couple of graduate students hacking away until they got tired. The protocol implementations were then distributed free, which resulted in their becoming widely used, deeply entrenched, and thus hard to replace. Some of them are a bit of an embarrassment now. The virtual terminal protocol, TELNET, for example, was designed for a ten-character per second mechanical Teletype terminal. It knows nothing of graphical user interfaces and mice. Nevertheless, 25 years later, it is still in widespread use.

In summary, despite its problems, the OSI model (minus the session and presentation layers) has proven to be exceptionally useful for discussing computer networks. In contrast, the OSI protocols have not become popular. The reverse is true of TCP/IP: the model is practically nonexistent, but the protocols are widely used. Since computer scientists like to have their cake and eat it, too, in this book we will use a modified OSI model but concentrate primarily on the TCP/IP and related protocols, as well as newer ones such as 802, SONET, and Bluetooth. In effect, we will use the hybrid model of Fig. 1-24 as the framework for this book.

Figure 1-24. The hybrid reference model to be used in this book.



[Team LiB]

PREVIOUS NEXT

1.5 Example Networks

The subject of computer networking covers many different kinds of networks, large and small, well known and less well known. They have different goals, scales, and technologies. In the following sections, we will look at some examples, to get an idea of the variety one finds in the area of computer networking.

We will start with the Internet, probably the best known network, and look at its history, evolution, and technology. Then we will consider ATM, which is often used within the core of large (telephone) networks. Technically, it is quite different from the Internet, contrasting nicely with it. Next we will introduce Ethernet, the dominant local area network. Finally, we will look at IEEE 802.11, the standard for wireless LANs.

1.5.1 The Internet

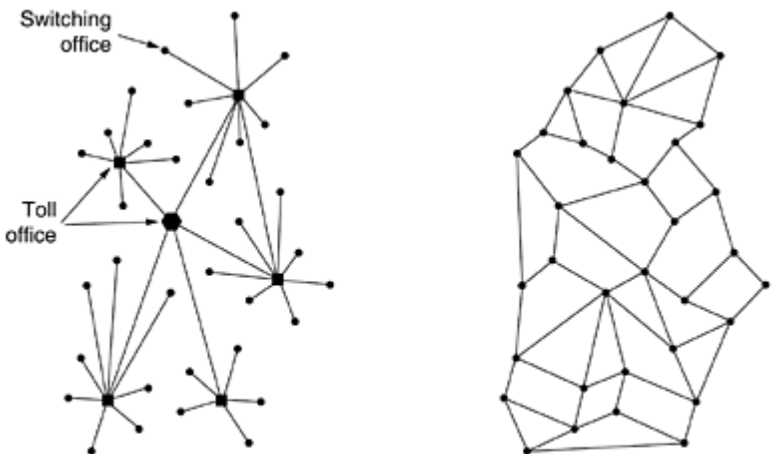
The Internet is not a network at all, but a vast collection of different networks that use certain common protocols and provide certain common services. It is an unusual system in that it was not planned by anyone and is not controlled by anyone. To better understand it, let us start from the beginning and see how it has developed and why. For a wonderful history of the Internet, John Naughton's (2000) book is highly recommended. It is one of those rare books that is not only fun to read, but also has 20 pages of ibid.'s and op. cit.'s for the serious historian. Some of the material below is based on this book.

Of course, countless technical books have been written about the Internet and its protocols as well. For more information, see, for example, (Maufer, 1999).

The ARPANET

The story begins in the late 1950s. At the height of the Cold War, the DoD wanted a command-and-control network that could survive a nuclear war. At that time, all military communications used the public telephone network, which was considered vulnerable. The reason for this belief can be gleaned from Fig. 1-25(a). Here the black dots represent telephone switching offices, each of which was connected to thousands of telephones. These switching offices were, in turn, connected to higher-level switching offices (toll offices), to form a national hierarchy with only a small amount of redundancy. The vulnerability of the system was that the destruction of a few key toll offices could fragment the system into many isolated islands.

Figure 1-25. (a) Structure of the telephone system. (b) Baran's proposed distributed switching system.



1.6 Network Standardization

Many network vendors and suppliers exist, each with its own ideas of how things should be done. Without coordination, there would be complete chaos, and users would get nothing done. The only way out is to agree on some network standards.

Not only do standards allow different computers to communicate, but they also increase the market for products adhering to the standard. A larger market leads to mass production, economies of scale in manufacturing, VLSI implementations, and other benefits that decrease price and further increase acceptance. In the following sections we will take a quick look at the important, but little-known, world of international standardization.

Standards fall into two categories: de facto and de jure. De facto (Latin for "from the fact") standards are those that have just happened, without any formal plan. The IBM PC and its successors are de facto standards for small-office and home computers because dozens of manufacturers chose to copy IBM's machines very closely. Similarly, UNIX is the de facto standard for operating systems in university computer science departments.

De jure (Latin for "by law") standards, in contrast, are formal, legal standards adopted by some authorized standardization body. International standardization authorities are generally divided into two classes: those established by treaty among national governments, and those comprising voluntary, nontreaty organizations. In the area of computer network standards, there are several organizations of each type, which are discussed below.

1.6.1 Who's Who in the Telecommunications World

The legal status of the world's telephone companies varies considerably from country to country. At one extreme is the United States, which has 1500 separate, privately owned telephone companies. Before it was broken up in 1984, AT&T, at that time the world's largest corporation, completely dominated the scene. It provided telephone service to about 80 percent of America's telephones, spread throughout half of its geographical area, with all the other companies combined servicing the remaining (mostly rural) customers. Since the breakup, AT&T continues to provide long-distance service, although now in competition with other companies. The seven Regional Bell Operating Companies that were split off from AT&T and numerous independents provide local and cellular telephone service. Due to frequent mergers and other changes, the industry is in a constant state of flux.

Companies in the United States that provide communication services to the public are called common carriers. Their offerings and prices are described by a document called a tariff, which must be approved by the Federal Communications Commission for the interstate and international traffic and by the state public utilities commissions for intrastate traffic.

At the other extreme are countries in which the national government has a complete monopoly on all communication, including the mail, telegraph, telephone, and often, radio and television. Most of the world falls in this category. In some cases the telecommunication authority is a nationalized company, and in others it is simply a branch of the government, usually known as the PTT (Post, Telegraph & Telephone administration). Worldwide, the trend is toward liberalization and competition and away from government monopoly. Most European countries have now (partially) privatized their PTTs, but elsewhere the process is still slowly gaining steam.

1.7 Metric Units

To avoid any confusion, it is worth stating explicitly that in this book, as in computer science in general, metric units are used instead of traditional English units (the furlong-stone-fortnight system). The principal metric prefixes are listed in Fig. 1-39. The prefixes are typically abbreviated by their first letters, with the units greater than 1 capitalized (KB, MB, etc.). One exception (for historical reasons) is kbps for kilobits/sec. Thus, a 1-Mbps communication line transmits 106 bits/sec and a 100 psec (or 100 ps) clock ticks every 10-10 seconds. Since milli and micro both begin with the letter "m," a choice had to be made. Normally, "m" is for milli and " " (the Greek letter mu) is for micro.

Figure 1-39. The principal metric prefixes.

Exp.	Explicit	Prefix	Exp.	Explicit	Prefix
10^{-3}	0.001	milli	10^{3}	1,000	Kilo
10^{-6}	0.000001	micro	10^{6}	1,000,000	Mega
10^{-9}	0.000000001	nano	10^{9}	1,000,000,000	Giga
10^{-12}	0.000000000001	pico	10^{12}	1,000,000,000,000	Tera
10^{-15}	0.000000000000001	femto	10^{15}	1,000,000,000,000,000	Peta
10^{-18}	0.000000000000000001	atto	10^{18}	1,000,000,000,000,000,000	Exa
10^{-21}	0.000000000000000000001	zepto	10^{21}	1,000,000,000,000,000,000,000	Zetta
10^{-24}	0.000000000000000000000001	yocto	10^{24}	1,000,000,000,000,000,000,000,000	Yotta

It is also worth pointing out that for measuring memory, disk, file, and database sizes, in common industry practice, the units have slightly different meanings. There, kilo means 210 (1024) rather than 103 (1000) because memories are always a power of two. Thus, a 1-KB memory contains 1024 bytes, not 1000 bytes. Similarly, a 1-MB memory contains 220 (1,048,576) bytes, a 1-GB memory contains 230 (1,073,741,824) bytes, and a 1-TB database contains 240 (1,099,511,627,776) bytes. However, a 1-kbps communication line transmits 1000 bits per second and a 10-Mbps LAN runs at 10,000,000 bits/sec because these speeds are not powers of two. Unfortunately, many people tend to mix up these two systems, especially for disk sizes. To avoid ambiguity, in this book, we will use the symbols KB, MB, and GB for 210, 220, and 230 bytes, respectively, and the symbols kbps, Mbps, and Gbps for 103, 106, and 109 bits/sec, respectively.

1.8 Outline of the Rest of the Book

This book discusses both the principles and practice of computer networking. Most chapters start with a discussion of the relevant principles, followed by a number of examples that illustrate these principles. These examples are usually taken from the Internet and wireless networks since these are both important and very different. Other examples will be given where relevant.

The book is structured according to the hybrid model of Fig. 1-24. Starting with Chap. 2, we begin working our way up the protocol hierarchy beginning at the bottom. The second chapter provides some background in the field of data communication. It covers wired, wireless, and satellite transmission systems. This material is concerned with the physical layer, although we cover only the architectural rather than the hardware aspects. Several examples of the physical layer, such as the public switched telephone network, mobile telephones, and the cable television network are also discussed.

Chapter 3 discusses the data link layer and its protocols by means of a number of increasingly complex examples. The analysis of these protocols is also covered. After that, some important real-world protocols are discussed, including HDLC (used in low- and medium-speed networks) and PPP (used in the Internet).

Chapter 4 concerns the medium access sublayer, which is part of the data link layer. The basic question it deals with is how to determine who may use the network next when the network consists of a single shared channel, as in most LANs and some satellite networks. Many examples are given from the areas of wired LANs, wireless LANs (especially Ethernet), wireless MANs, Bluetooth, and satellite networks. Bridges and data link switches, which are used to connect LANs, are also discussed here.

Chapter 5 deals with the network layer, especially routing, with many routing algorithms, both static and dynamic, being covered. Even with good routing algorithms though, if more traffic is offered than the network can handle, congestion can develop, so we discuss congestion and how to prevent it. Even better than just preventing congestion is guaranteeing a certain quality of service. We will discuss that topic as well here. Connecting heterogeneous networks to form internetworks leads to numerous problems that are discussed here. The network layer in the Internet is given extensive coverage.

Chapter 6 deals with the transport layer. Much of the emphasis is on connection-oriented protocols, since many applications need these. An example transport service and its implementation are discussed in detail. The actual code is given for this simple example to show how it could be implemented. Both Internet transport protocols, UDP and TCP, are covered in detail, as are their performance issues. Issues concerning wireless networks are also covered.

Chapter 7 deals with the application layer, its protocols and applications. The first topic is DNS, which is the Internet's telephone book. Next comes e-mail, including a discussion of its protocols. Then we move onto the Web, with detailed discussions of the static content, dynamic content, what happens on the client side, what happens on the server side, protocols, performance, the wireless Web, and more. Finally, we examine networked multimedia, including streaming audio, Internet radio, and video on demand.

Chapter 8 is about network security. This topic has aspects that relate to all layers, so it is easiest to treat it after all the layers have been thoroughly explained. The chapter starts with an introduction to cryptography. Later, it shows how cryptography can be used to secure communication, e-mail, and the Web. The book ends with a discussion of

1.9 Summary

Computer networks can be used for numerous services, both for companies and for individuals. For companies, networks of personal computers using shared servers often provide access to corporate information. Typically they follow the client-server model, with client workstations on employee desktops accessing powerful servers in the machine room. For individuals, networks offer access to a variety of information and entertainment resources. Individuals often access the Internet by calling up an ISP using a modem, although increasingly many people have a fixed connection at home. An up-and-coming area is wireless networking with new applications such as mobile e-mail access and m-commerce.

Roughly speaking, networks can be divided up into LANs, MANs, WANs, and internetworks, with their own characteristics, technologies, speeds, and niches. LANs cover a building and operate at high speeds. MANs cover a city, for example, the cable television system, which is now used by many people to access the Internet. WANs cover a country or continent. LANs and MANs are unswitched (i.e., do not have routers); WANs are switched. Wireless networks are becoming extremely popular, especially wireless LANs. Networks can be interconnected to form internetworks.

Network software consists of protocols, which are rules by which processes communicate. Protocols are either connectionless or connection-oriented. Most networks support protocol hierarchies, with each layer providing services to the layers above it and insulating them from the details of the protocols used in the lower layers. Protocol stacks are typically based either on the OSI model or on the TCP/IP model. Both have network, transport, and application layers, but they differ on the other layers. Design issues include multiplexing, flow control, error control, and others. Much of this book deals with protocols and their design.

Networks provide services to their users. These services can be connection-oriented or connectionless. In some networks, connectionless service is provided in one layer and connection-oriented service is provided in the layer above it.

Well-known networks include the Internet, ATM networks, Ethernet, and the IEEE 802.11 wireless LAN. The Internet evolved from the ARPANET, to which other networks were added to form an internetwork. The present Internet is actually a collection of many thousands of networks, rather than a single network. What characterizes it is the use of the TCP/IP protocol stack throughout. ATM is widely used inside the telephone system for long-haul data traffic. Ethernet is the most popular LAN and is present in most large companies and universities. Finally, wireless LANs at surprisingly high speeds (up to 54 Mbps) are beginning to be widely deployed.

To have multiple computers talk to each other requires a large amount of standardization, both in the hardware and software. Organizations such as the ITU-T, ISO, IEEE, and IAB manage different parts of the standardization process.

Problems

1.

1. Imagine that you have trained your St. Bernard, Bernie, to carry a box of three 8mm tapes instead of a flask of brandy. (When your disk fills up, you consider that an emergency.) These tapes each contain 7 gigabytes. The dog can travel to your side, wherever you may be, at 18 km/hour. For what range of distances does

Chapter 2. The Physical Layer

In this chapter we will look at the lowest layer depicted in the hierarchy of Fig. 1-24. It defines the mechanical, electrical, and timing interfaces to the network. We will begin with a theoretical analysis of data transmission, only to discover that Mother (Parent?) Nature puts some limits on what can be sent over a channel.

Then we will cover three kinds of transmission media: guided (copper wire and fiber optics), wireless (terrestrial radio), and satellite. This material will provide background information on the key transmission technologies used in modern networks.

The remainder of the chapter will be devoted to three examples of communication systems used in practice for wide area computer networks: the (fixed) telephone system, the mobile phone system, and the cable television system. All three use fiber optics in the backbone, but they are organized differently and use different technologies for the last mile.

2.1 The Theoretical Basis for Data Communication

Information can be transmitted on wires by varying some physical property such as voltage or current. By representing the value of this voltage or current as a single-valued function of time, f(t), we can model the behavior of the signal and analyze it mathematically. This analysis is the subject of the following sections.

2.1.1 Fourier Analysis

In the early 19th century, the French mathematician Jean-Baptiste Fourier proved that any reasonably behaved periodic function, g(t) with period T can be constructed as the sum of a (possibly infinite) number of sines and cosines:

Equation 2

$$g(t) = \frac{1}{2}c + \sum_{n=1}^{\infty} a_n \sin(2\pi nft) + \sum_{n=1}^{\infty} b_n \cos(2\pi nft)$$

where f = 1/T is the fundamental frequency, an and bn are the sine and cosine amplitudes of the nth harmonics (terms), and c is a constant. Such a decomposition is called a Fourier series. From the Fourier series, the function can be reconstructed; that is, if the period, T, is known and the amplitudes are given, the original function of time can be found by performing the sums of Eq. (2-1).

A data signal that has a finite duration (which all of them do) can be handled by just imagining that it repeats the entire pattern over and over forever (i.e., the interval from T to 2T is the same as from 0 to T, etc.).

The an amplitudes can be computed for any given g(t) by multiplying both sides of Eq. (2-1) by sin(2? kft) and then integrating from 0 to T. Since

$$\int_0^T \sin(2\pi kft)\,\sin(2\pi nft)\,dt = \begin{cases} 0 \text{ for } k \neq n \\ T/2 \text{ for } k = n \end{cases}$$

only one term of the summation survives: an. The bn summation vanishes completely. Similarly, by multiplying Eq. (2-1) by cos(2? kft) and integrating between 0 and T, we can derive bn. By just integrating both sides of the equation as it stands, we can find c. The results of performing these operations are as follows:

$$a_n = \frac{2}{T}\int_0^T g(t)\sin(2\pi nft)\,dt \qquad b_n = \frac{2}{T}\int_0^T g(t)\cos(2\pi nft)\,dt \qquad c = \frac{2}{T}\int_0^T g(t)\,dt$$

2.1.2 Bandwidth-Limited Signals

To see what all this has to do with data communication, let us consider a specific example: the transmission of the ASCII character "b" encoded in an 8-bit byte. The bit pattern that is to be transmitted is 01100010. The left-hand

2.2 Guided Transmission Media

The purpose of the physical layer is to transport a raw bit stream from one machine to another. Various physical media can be used for the actual transmission. Each one has its own niche in terms of bandwidth, delay, cost, and ease of installation and maintenance. Media are roughly grouped into guided media, such as copper wire and fiber optics, and unguided media, such as radio and lasers through the air. We will look at all of these in the following sections.

2.2.1 Magnetic Media

One of the most common ways to transport data from one computer to another is to write them onto magnetic tape or removable media (e.g., recordable DVDs), physically transport the tape or disks to the destination machine, and read them back in again. Although this method is not as sophisticated as using a geosynchronous communication satellite, it is often more cost effective, especially for applications in which high bandwidth or cost per bit transported is the key factor.

A simple calculation will make this point clear. An industry standard Ultrium tape can hold 200 gigabytes. A box 60 x 60 x 60 cm can hold about 1000 of these tapes, for a total capacity of 200 terabytes, or 1600 terabits (1.6 petabits). A box of tapes can be delivered anywhere in the United States in 24 hours by Federal Express and other companies. The effective bandwidth of this transmission is 1600 terabits/86,400 sec, or 19 Gbps. If the destination is only an hour away by road, the bandwidth is increased to over 400 Gbps. No computer network can even approach this.

For a bank with many gigabytes of data to be backed up daily on a second machine (so the bank can continue to function even in the face of a major flood or earthquake), it is likely that no other transmission technology can even begin to approach magnetic tape for performance. Of course, networks are getting faster, but tape densities are increasing, too.

If we now look at cost, we get a similar picture. The cost of an Ultrium tape is around $40 when bought in bulk. A tape can be reused at least ten times, so the tape cost is maybe $4000 per box per usage. Add to this another $1000 for shipping (probably much less), and we have a cost of roughly $5000 to ship 200 TB. This amounts to shipping a gigabyte for under 3 cents. No network can beat that. The moral of the story is:

Never underestimate the bandwidth of a station wagon full of tapes hurtling down the highway

2.2.2 Twisted Pair

Although the bandwidth characteristics of magnetic tape are excellent, the delay characteristics are poor. Transmission time is measured in minutes or hours, not milliseconds. For many applications an on-line connection is needed. One of the oldest and still most common transmission media is twisted pair. A twisted pair consists of two insulated copper wires, typically about 1 mm thick. The wires are twisted together in a helical form, just like a DNA molecule. Twisting is done because two parallel wires constitute a fine antenna. When the wires are twisted, the waves from different twists cancel out, so the wire radiates less effectively.

The most common application of the twisted pair is the telephone system. Nearly all telephones are connected to the

2.3 Wireless Transmission

Our age has given rise to information junkies: people who need to be on-line all the time. For these mobile users, twisted pair, coax, and fiber optics are of no use. They need to get their hits of data for their laptop, notebook, shirt pocket, palmtop, or wristwatch computers without being tethered to the terrestrial communication infrastructure. For these users, wireless communication is the answer. In the following sections, we will look at wireless communication in general, as it has many other important applications besides providing connectivity to users who want to surf the Web from the beach.

Some people believe that the future holds only two kinds of communication: fiber and wireless. All fixed (i.e., nonmobile) computers, telephones, faxes, and so on will use fiber, and all mobile ones will use wireless.

Wireless has advantages for even fixed devices in some circumstances. For example, if running a fiber to a building is difficult due to the terrain (mountains, jungles, swamps, etc.), wireless may be better. It is noteworthy that modern wireless digital communication began in the Hawaiian Islands, where large chunks of Pacific Ocean separated the users and the telephone system was inadequate.

2.3.1 The Electromagnetic Spectrum

When electrons move, they create electromagnetic waves that can propagate through space (even in a vacuum). These waves were predicted by the British physicist James Clerk Maxwell in 1865 and first observed by the German physicist Heinrich Hertz in 1887. The number of oscillations per second of a wave is called its frequency, f, and is measured in Hz (in honor of Heinrich Hertz). The distance between two consecutive maxima (or minima) is called the wavelength, which is universally designated by the Greek letter ? (lambda).

When an antenna of the appropriate size is attached to an electrical circuit, the electromagnetic waves can be broadcast efficiently and received by a receiver some distance away. All wireless communication is based on this principle.

In vacuum, all electromagnetic waves travel at the same speed, no matter what their frequency. This speed, usually called the speed of light, c, is approximately 3 x 108 m/sec, or about 1 foot (30 cm) per nanosecond. (A case could be made for redefining the foot as the distance light travels in a vacuum in 1 nsec rather than basing it on the shoe size of some long-dead king.) In copper or fiber the speed slows to about 2/3 of this value and becomes slightly frequency dependent. The speed of light is the ultimate speed limit. No object or signal can ever move faster than it.

The fundamental relation between f, ?, and c (in vacuum) is

Equation 2

$$\lambda f = c$$

Since c is a constant, if we know f, we can find ?, and vice versa. As a rule of thumb, when ? is in meters and f is in MHz, ?f ≈300. For example, 100-MHz waves are about 3 meters long, 1000-MHz waves are 0.3-meters long,

2.4 Communication Satellites

In the 1950s and early 1960s, people tried to set up communication systems by bouncing signals off metallized weather balloons. Unfortunately, the received signals were too weak to be of any practical use. Then the U.S. Navy noticed a kind of permanent weather balloon in the sky—the moon—and built an operational system for ship-to-shore communication by bouncing signals off it.

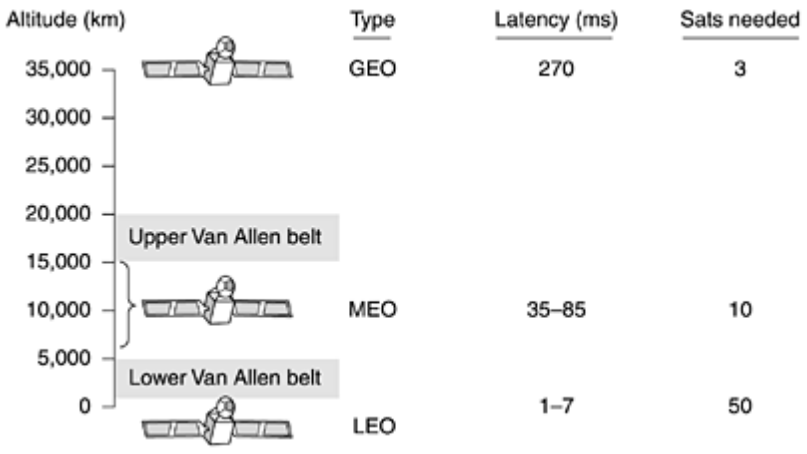
Further progress in the celestial communication field had to wait until the first communication satellite was launched. The key difference between an artificial satellite and a real one is that the artificial one can amplify the signals before sending them back, turning a strange curiosity into a powerful communication system.

Communication satellites have some interesting properties that make them attractive for many applications. In its simplest form, a communication satellite can be thought of as a big microwave repeater in the sky. It contains several transponders, each of which listens to some portion of the spectrum, amplifies the incoming signal, and then rebroadcasts it at another frequency to avoid interference with the incoming signal. The downward beams can be broad, covering a substantial fraction of the earth's surface, or narrow, covering an area only hundreds of kilometers in diameter. This mode of operation is known as a bent pipe.

According to Kepler's law, the orbital period of a satellite varies as the radius of the orbit to the 3/2 power. The higher the satellite, the longer the period. Near the surface of the earth, the period is about 90 minutes. Consequently, low-orbit satellites pass out of view fairly quickly, so many of them are needed to provide continuous coverage. At an altitude of about 35,800 km, the period is 24 hours. At an altitude of 384,000 km, the period is about one month, as anyone who has observed the moon regularly can testify.

A satellite's period is important, but it is not the only issue in determining where to place it. Another issue is the presence of the Van Allen belts, layers of highly charged particles trapped by the earth's magnetic field. Any satellite flying within them would be destroyed fairly quickly by the highly-energetic charged particles trapped there by the earth's magnetic field. These factors lead to three regions in which satellites can be placed safely. These regions and some of their properties are illustrated in Fig. 2-15. Below we will briefly describe the satellites that inhabit each of these regions.

Figure 2-15. Communication satellites and some of their properties, including altitude above the earth, round-trip delay time, and number of satellites needed for global coverage.



2.5 The Public Switched Telephone Network

When two computers owned by the same company or organization and located close to each other need to communicate, it is often easiest just to run a cable between them. LANs work this way. However, when the distances are large or there are many computers or the cables have to pass through a public road or other public right of way, the costs of running private cables are usually prohibitive. Furthermore, in just about every country in the world, stringing private transmission lines across (or underneath) public property is also illegal. Consequently, the network designers must rely on the existing telecommunication facilities.

These facilities, especially the PSTN (Public Switched Telephone Network), were usually designed many years ago, with a completely different goal in mind: transmitting the human voice in a more-or-less recognizable form. Their suitability for use in computer-computer communication is often marginal at best, but the situation is rapidly changing with the introduction of fiber optics and digital technology. In any event, the telephone system is so tightly intertwined with (wide area) computer networks, that it is worth devoting some time to studying it.

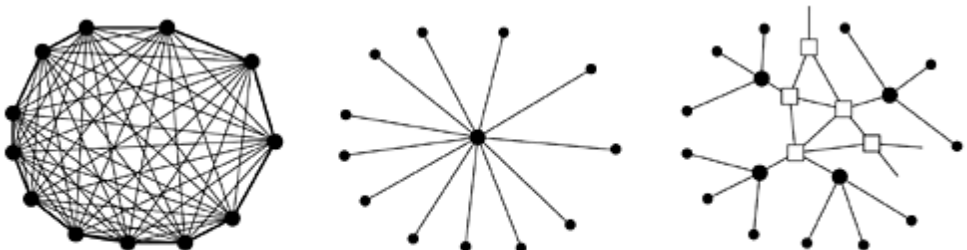
To see the order of magnitude of the problem, let us make a rough but illustrative comparison of the properties of a typical computer-computer connection via a local cable and via a dial-up telephone line. A cable running between two computers can transfer data at 109 bps, maybe more. In contrast, a dial-up line has a maximum data rate of 56 kbps, a difference of a factor of almost 20,000. That is the difference between a duck waddling leisurely through the grass and a rocket to the moon. If the dial-up line is replaced by an ADSL connection, there is still a factor of 1000–2000 difference.

The trouble, of course, is that computer systems designers are used to working with computer systems and when suddenly confronted with another system whose performance (from their point of view) is 3 or 4 orders of magnitude worse, they, not surprising, devoted much time and effort to trying to figure out how to use it efficiently. In the following sections we will describe the telephone system and show how it works. For additional information about the innards of the telephone system see (Bellamy, 2000).

2.5.1 Structure of the Telephone System

Soon after Alexander Graham Bell patented the telephone in 1876 (just a few hours ahead of his rival, Elisha Gray), there was an enormous demand for his new invention. The initial market was for the sale of telephones, which came in pairs. It was up to the customer to string a single wire between them. The electrons returned through the earth. If a telephone owner wanted to talk to n other telephone owners, separate wires had to be strung to all n houses. Within a year, the cities were covered with wires passing over houses and trees in a wild jumble. It became immediately obvious that the model of connecting every telephone to every other telephone, as shown in Fig. 2-20(a), was not going to work.

Figure 2-20. (a) Fully-interconnected network. (b) Centralized switch. (c) Two-level hierarchy.



2.6 The Mobile Telephone System

The traditional telephone system (even if it some day gets multigigabit end-to-end fiber) will still not be able to satisfy a growing group of users: people on the go. People now expect to make phone calls from airplanes, cars, swimming pools, and while jogging in the park. Within a few years they will also expect to send e-mail and surf the Web from all these locations and more. Consequently, there is a tremendous amount of interest in wireless telephony. In the following sections we will study this topic in some detail.

Wireless telephones come in two basic varieties: cordless phones and mobile phones (sometimes called cell phones). Cordless phones are devices consisting of a base station and a handset sold as a set for use within the home. These are never used for networking, so we will not examine them further. Instead we will concentrate on the mobile system, which is used for wide area voice and data communication.

Mobile phones have gone through three distinct generations, with different technologies:

1.

1. Analog voice.
2.

2. Digital voice.
3.

3. Digital voice and data (Internet, e-mail, etc.).

Although most of our discussion will be about the technology of these systems, it is interesting to note how political and tiny marketing decisions can have a huge impact. The first mobile system was devised in the U.S. by AT&T and mandated for the whole country by the FCC. As a result, the entire U.S. had a single (analog) system and a mobile phone purchased in California also worked in New York. In contrast, when mobile came to Europe, every country devised its own system, which resulted in a fiasco.

Europe learned from its mistake and when digital came around, the government-run PTTs got together and standardized on a single system (GSM), so any European mobile phone will work anywhere in Europe. By then, the U.S. had decided that government should not be in the standardization business, so it left digital to the marketplace. This decision resulted in different equipment manufacturers producing different kinds of mobile phones. As a consequence, the U.S. now has two major incompatible digital mobile phone systems in operation (plus one minor one).

Despite an initial lead by the U.S., mobile phone ownership and usage in Europe is now far greater than in the U.S. Having a single system for all of Europe is part of the reason, but there is more. A second area where the U.S. and Europe differed is in the humble matter of phone numbers. In the U.S. mobile phones are mixed in with regular (fixed) telephones. Thus, there is no way for a caller to see if, say, (212) 234-5678 is a fixed telephone (cheap or free call) or a mobile phone (expensive call). To keep people from getting nervous about using the telephone, the telephone companies decided to make the mobile phone owner pay for incoming calls. As a consequence, many people hesitated to buy a mobile phone for fear of running up a big bill by just receiving calls. In Europe, mobile phones have a special area code (analogous to 800 and 900 numbers) so they are instantly recognizable. Consequently, the usual rule of "caller pays" also applies to mobile phones in Europe (except for international calls where costs are split)

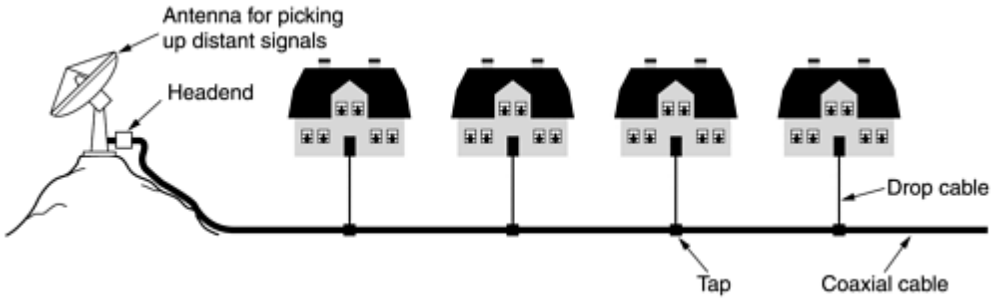
2.7 Cable Television

We have now studied both the fixed and wireless telephone systems in a fair amount of detail. Both will clearly play a major role in future networks. However, an alternative available for fixed networking is now becoming a major player: cable television networks. Many people already get their telephone and Internet service over the cable, and the cable operators are actively working to increase their market share. In the following sections we will look at cable television as a networking system in more detail and contrast it with the telephone systems we have just studied. For more information about cable, see (Laubach et al., 2001; Louis, 2002; Ovadia, 2001; and Smith, 2002).

2.7.1 Community Antenna Television

Cable television was conceived in the late 1940s as a way to provide better reception to people living in rural or mountainous areas. The system initially consisted of a big antenna on top of a hill to pluck the television signal out of the air, an amplifier, called the head end, to strengthen it, and a coaxial cable to deliver it to people's houses, as illustrated in Fig. 2-46.

Figure 2-46. An early cable television system.



In the early years, cable television was called Community Antenna Television. It was very much a mom-and-pop operation; anyone handy with electronics could set up a service for his town, and the users would chip in to pay the costs. As the number of subscribers grew, additional cables were spliced onto the original cable and amplifiers were added as needed. Transmission was one way, from the headend to the users. By 1970, thousands of independent systems existed.

In 1974, Time, Inc., started a new channel, Home Box Office, with new content (movies) and distributed only on cable. Other cable-only channels followed with news, sports, cooking, and many other topics. This development gave rise to two changes in the industry. First, large corporations began buying up existing cable systems and laying new cable to acquire new subscribers. Second, there was now a need to connect multiple systems, often in distant cities, in order to distribute the new cable channels. The cable companies began to lay cable between their cities to connect them all into a single system. This pattern was analogous to what happened in the telephone industry 80 years earlier with the connection of previously isolated end offices to make long distance calling possible.

2.7.2 Internet over Cable

Over the course of the years the cable system grew and the cables between the various cities were replaced by high-bandwidth fiber, similar to what was happening in the telephone system. A system with fiber for the long-haul runs and coaxial cable to the houses is called an HFC (Hybrid Fiber Coax) system. The electro-optical converters that interface between the optical and electrical parts of the system are called fiber nodes. Because the bandwidth of

2.8 Summary

The physical layer is the basis of all networks. Nature imposes two fundamental limits on all channels, and these determine their bandwidth. These limits are the Nyquist limit, which deals with noiseless channels, and the Shannon limit, which deals with noisy channels.

Transmission media can be guided or unguided. The principal guided media are twisted pair, coaxial cable, and fiber optics. Unguided media include radio, microwaves, infrared, and lasers through the air. An up-and-coming transmission system is satellite communication, especially LEO systems.

A key element in most wide area networks is the telephone system. Its main components are the local loops, trunks, and switches. Local loops are analog, twisted pair circuits, which require modems for transmitting digital data. ADSL offers speeds up to 50 Mbps by dividing the local loop into many virtual channels and modulating each one separately. Wireless local loops are another new development to watch, especially LMDS.

Trunks are digital, and can be multiplexed in several ways, including FDM, TDM, and WDM. Both circuit switching and packet switching are important.

For mobile applications, the fixed telephone system is not suitable. Mobile phones are currently in widespread use for voice and will soon be in widespread use for data. The first generation was analog, dominated by AMPS. The second generation was digital, with D-AMPS, GSM, and CDMA the major options. The third generation will be digital and based on broadband CDMA.

An alternative system for network access is the cable television system, which has gradually evolved from a community antenna to hybrid fiber coax. Potentially, it offers very high bandwidth, but the actual bandwidth available in practice depends heavily on the number of other users currently active and what they are doing.

Problems

1.

1. Compute the Fourier coefficients for the function f(t) = t (0 $\leq$ t $\leq$ 1).
2.

2. A noiseless 4-kHz channel is sampled every 1 msec. What is the maximum data rate?
3.

3. Television channels are 6 MHz wide. How many bits/sec can be sent if four-level digital signals are used? Assume a noiseless channel.
4.

4. If a binary signal is sent over a 3-kHz channel whose signal-to-noise ratio is 20 dB, what is the maximum achievable data rate?
5.

5. What signal-to-noise ratio is needed to put a T1 carrier on a 50-kHz line?

Chapter 3. The Data Link Layer

In this chapter we will study the design principles for layer 2, the data link layer. This study deals with the algorithms for achieving reliable, efficient communication between two adjacent machines at the data link layer. By adjacent, we mean that the two machines are connected by a communication channel that acts conceptually like a wire (e.g., a coaxial cable, telephone line, or point-to-point wireless channel). The essential property of a channel that makes it "wirelike" is that the bits are delivered in exactly the same order in which they are sent.

At first you might think this problem is so trivial that there is no software to study—machine A just puts the bits on the wire, and machine B just takes them off. Unfortunately, communication circuits make errors occasionally. Furthermore, they have only a finite data rate, and there is a nonzero propagation delay between the time a bit is sent and the time it is received. These limitations have important implications for the efficiency of the data transfer. The protocols used for communications must take all these factors into consideration. These protocols are the subject of this chapter.

After an introduction to the key design issues present in the data link layer, we will start our study of its protocols by looking at the nature of errors, their causes, and how they can be detected and corrected. Then we will study a series of increasingly complex protocols, each one solving more and more of the problems present in this layer. Finally, we will conclude with an examination of protocol modeling and correctness and give some examples of data link protocols.

3.1 Data Link Layer Design Issues

The data link layer has a number of specific functions it can carry out. These functions include

1.

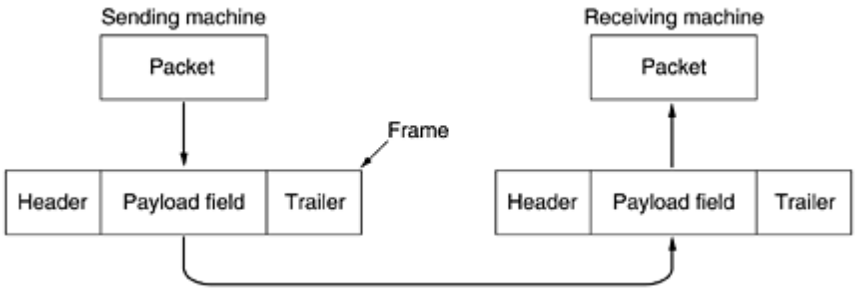
1. Providing a well-defined service interface to the network layer.
2.

2. Dealing with transmission errors.
3.

3. Regulating the flow of data so that slow receivers are not swamped by fast senders.

To accomplish these goals, the data link layer takes the packets it gets from the network layer and encapsulates them into frames for transmission. Each frame contains a frame header, a payload field for holding the packet, and a frame trailer, as illustrated in Fig. 3-1. Frame management forms the heart of what the data link layer does. In the following sections we will examine all the above-mentioned issues in detail.

Figure 3-1. Relationship between packets and frames.



Although this chapter is explicitly about the data link layer and the data link protocols, many of the principles we will study here, such as error control and flow control, are found in transport and other protocols as well. In fact, in many networks, these functions are found only in the upper layers and not in the data link layer. However, no matter where they are found, the principles are pretty much the same, so it does not really matter where we study them. In the data link layer they often show up in their simplest and purest forms, making this a good place to examine them in detail.

3.1.1 Services Provided to the Network Layer

The function of the data link layer is to provide services to the network layer. The principal service is transferring data from the network layer on the source machine to the network layer on the destination machine. On the source machine is an entity, call it a process, in the network layer that hands some bits to the data link layer for transmission to the destination. The job of the data link layer is to transmit the bits to the destination machine so they can be handed over to the network layer there, as shown in Fig. 3-2(a). The actual transmission follows the path of Fig. 3-2(b), but it is easier to think in terms of two data link layer processes communicating using a data link protocol. For this reason, we will implicitly use the model of Fig. 3-2(a) throughout this chapter.

Figure 3-2. (a) Virtual communication. (b) Actual communication.

3.2 Error Detection and Correction

As we saw in Chap. 2, the telephone system has three parts: the switches, the interoffice trunks, and the local loops. The first two are now almost entirely digital in most developed countries. The local loops are still analog twisted copper pairs and will continue to be so for years due to the enormous expense of replacing them. While errors are rare on the digital part, they are still common on the local loops. Furthermore, wireless communication is becoming more common, and the error rates here are orders of magnitude worse than on the interoffice fiber trunks. The conclusion is: transmission errors are going to be with us for many years to come. We have to learn how to deal with them.

As a result of the physical processes that generate them, errors on some media (e.g., radio) tend to come in bursts rather than singly. Having the errors come in bursts has both advantages and disadvantages over isolated single-bit errors. On the advantage side, computer data are always sent in blocks of bits. Suppose that the block size is 1000 bits and the error rate is 0.001 per bit. If errors were independent, most blocks would contain an error. If the errors came in bursts of 100 however, only one or two blocks in 100 would be affected, on average. The disadvantage of burst errors is that they are much harder to correct than are isolated errors.

3.2.1 Error-Correcting Codes

Network designers have developed two basic strategies for dealing with errors. One way is to include enough redundant information along with each block of data sent, to enable the receiver to deduce what the transmitted data must have been. The other way is to include only enough redundancy to allow the receiver to deduce that an error occurred, but not which error, and have it request a retransmission. The former strategy uses error-correcting codes and the latter uses error-detecting codes. The use of error-correcting codes is often referred to as forward error correction.

Each of these techniques occupies a different ecological niche. On channels that are highly reliable, such as fiber, it is cheaper to use an error detecting code and just retransmit the occasional block found to be faulty. However, on channels such as wireless links that make many errors, it is better to add enough redundancy to each block for the receiver to be able to figure out what the original block was, rather than relying on a retransmission, which itself may be in error.

To understand how errors can be handled, it is necessary to look closely at what an error really is. Normally, a frame consists of m data (i.e., message) bits and r redundant, or check, bits. Let the total length be n (i.e., n = m + r). An n-bit unit containing data and check bits is often referred to as an n-bit codeword.

Given any two codewords, say, 10001001 and 10110001, it is possible to determine how many corresponding bits differ. In this case, 3 bits differ. To determine how many bits differ, just exclusive OR the two codewords and count the number of 1 bits in the result, for example:

```
10001001
10110001
00111000
```

The number of bit positions in which two codewords differ is called the Hamming distance (Hamming, 1950). Its significance is that if two codewords are a Hamming distance d apart, it will require d single-bit errors to convert one

3.3 Elementary Data Link Protocols

To introduce the subject of protocols, we will begin by looking at three protocols of increasing complexity. For interested readers, a simulator for these and subsequent protocols is available via the Web (see the preface). Before we look at the protocols, it is useful to make explicit some of the assumptions underlying the model of communication. To start with, we assume that in the physical layer, data link layer, and network layer are independent processes that communicate by passing messages back and forth. In many cases, the physical and data link layer processes will be running on a processor inside a special network I/O chip and the network layer code will be running on the main CPU. However, other implementations are also possible (e.g., three processes inside a single I/O chip; or the physical and data link layers as procedures called by the network layer process). In any event, treating the three layers as separate processes makes the discussion conceptually cleaner and also serves to emphasize the independence of the layers.

Another key assumption is that machine A wants to send a long stream of data to machine B, using a reliable, connection-oriented service. Later, we will consider the case where B also wants to send data to A simultaneously. A is assumed to have an infinite supply of data ready to send and never has to wait for data to be produced. Instead, when A's data link layer asks for data, the network layer is always able to comply immediately. (This restriction, too, will be dropped later.)

We also assume that machines do not crash. That is, these protocols deal with communication errors, but not the problems caused by computers crashing and rebooting.

As far as the data link layer is concerned, the packet passed across the interface to it from the network layer is pure data, whose every bit is to be delivered to the destination's network layer. The fact that the destination's network layer may interpret part of the packet as a header is of no concern to the data link layer.

When the data link layer accepts a packet, it encapsulates the packet in a frame by adding a data link header and trailer to it (see Fig. 3-1). Thus, a frame consists of an embedded packet, some control information (in the header), and a checksum (in the trailer). The frame is then transmitted to the data link layer on the other machine. We will assume that there exist suitable library procedures to_physical_layer to send a frame and from_physical_layer to receive a frame. The transmitting hardware computes and appends the checksum (thus creating the trailer), so that the datalink layer software need not worry about it. The polynomial algorithm discussed earlier in this chapter might be used, for example.

Initially, the receiver has nothing to do. It just sits around waiting for something to happen. In the example protocols of this chapter we will indicate that the data link layer is waiting for something to happen by the procedure call wait_for_event(&event). This procedure only returns when something has happened (e.g., a frame has arrived). Upon return, the variable event tells what happened. The set of possible events differs for the various protocols to be described and will be defined separately for each protocol. Note that in a more realistic situation, the data link layer will not sit in a tight loop waiting for an event, as we have suggested, but will receive an interrupt, which will cause it to stop whatever it was doing and go handle the incoming frame. Nevertheless, for simplicity we will ignore all the details of parallel activity within the data link layer and assume that it is dedicated full time to handling just our one channel.

When a frame arrives at the receiver, the hardware computes the checksum. If the checksum is incorrect (i.e., there was a transmission error), the data link layer is so informed (event = cksum_err). If the inbound frame arrived

3.4 Sliding Window Protocols

In the previous protocols, data frames were transmitted in one direction only. In most practical situations, there is a need for transmitting data in both directions. One way of achieving full-duplex data transmission is to have two separate communication channels and use each one for simplex data traffic (in different directions). If this is done, we have two separate physical circuits, each with a "forward" channel (for data) and a "reverse" channel (for acknowledgements). In both cases the bandwidth of the reverse channel is almost entirely wasted. In effect, the user is paying for two circuits but using only the capacity of one.

A better idea is to use the same circuit for data in both directions. After all, in protocols 2 and 3 it was already being used to transmit frames both ways, and the reverse channel has the same capacity as the forward channel. In this model the data frames from A to B are intermixed with the acknowledgement frames from A to B. By looking at the kind field in the header of an incoming frame, the receiver can tell whether the frame is data or acknowledgement.

Although interleaving data and control frames on the same circuit is an improvement over having two separate physical circuits, yet another improvement is possible. When a data frame arrives, instead of immediately sending a separate control frame, the receiver restrains itself and waits until the network layer passes it the next packet. The acknowledgement is attached to the outgoing data frame (using the ack field in the frame header). In effect, the acknowledgement gets a free ride on the next outgoing data frame. The technique of temporarily delaying outgoing acknowledgements so that they can be hooked onto the next outgoing data frame is known as piggybacking.

The principal advantage of using piggybacking over having distinct acknowledgement frames is a better use of the available channel bandwidth. The ack field in the frame header costs only a few bits, whereas a separate frame would need a header, the acknowledgement, and a checksum. In addition, fewer frames sent means fewer "frame arrival" interrupts, and perhaps fewer buffers in the receiver, depending on how the receiver's software is organized. In the next protocol to be examined, the piggyback field costs only 1 bit in the frame header. It rarely costs more than a few bits.

However, piggybacking introduces a complication not present with separate acknowledgements. How long should the data link layer wait for a packet onto which to piggyback the acknowledgement? If the data link layer waits longer than the sender's timeout period, the frame will be retransmitted, defeating the whole purpose of having acknowledgements. If the data link layer were an oracle and could foretell the future, it would know when the next network layer packet was going to come in and could decide either to wait for it or send a separate acknowledgement immediately, depending on how long the projected wait was going to be. Of course, the data link layer cannot foretell the future, so it must resort to some ad hoc scheme, such as waiting a fixed number of milliseconds. If a new packet arrives quickly, the acknowledgement is piggybacked onto it; otherwise, if no new packet has arrived by the end of this time period, the data link layer just sends a separate acknowledgement frame.

The next three protocols are bidirectional protocols that belong to a class called sliding window protocols. The three differ among themselves in terms of efficiency, complexity, and buffer requirements, as discussed later. In these, as in all sliding window protocols, each outbound frame contains a sequence number, ranging from 0 up to some maximum. The maximum is usually 2n - 1 so the sequence number fits exactly in an n-bit field. The stop-and-wait sliding window protocol uses n = 1, restricting the sequence numbers to 0 and 1, but more sophisticated versions can use arbitrary n.

The essence of all sliding window protocols is that at any instant of time, the sender maintains a set of sequence

3.5 Protocol Verification

Realistic protocols and the programs that implement them are often quite complicated. Consequently, much research has been done trying to find formal, mathematical techniques for specifying and verifying protocols. In the following sections we will look at some models and techniques. Although we are looking at them in the context of the data link layer, they are also applicable to other layers.

3.5.1 Finite State Machine Models

A key concept used in many protocol models is the finite state machine. With this technique, each protocol machine (i.e., sender or receiver) is always in a specific state at every instant of time. Its state consists of all the values of its variables, including the program counter.

In most cases, a large number of states can be grouped for purposes of analysis. For example, considering the receiver in protocol 3, we could abstract out from all the possible states two important ones: waiting for frame 0 or waiting for frame 1. All other states can be thought of as transient, just steps on the way to one of the main states. Typically, the states are chosen to be those instants that the protocol machine is waiting for the next event to happen [i.e., executing the procedure call wait(event) in our examples]. At this point the state of the protocol machine is completely determined by the states of its variables. The number of states is then 2n, where n is the number of bits needed to represent all the variables combined.

The state of the complete system is the combination of all the states of the two protocol machines and the channel. The state of the channel is determined by its contents. Using protocol 3 again as an example, the channel has four possible states: a 0 frame or a 1 frame moving from sender to receiver, an acknowledgement frame going the other way, or an empty channel. If we model the sender and receiver as each having two states, the complete system has 16 distinct states.

A word about the channel state is in order. The concept of a frame being "on the channel" is an abstraction, of course. What we really mean is that a frame has possibly been received, but not yet processed at the destination. A frame remains "on the channel" until the protocol machine executes FromPhysicalLayer and processes it.

From each state, there are zero or more possible transitions to other states. Transitions occur when some event happens. For a protocol machine, a transition might occur when a frame is sent, when a frame arrives, when a timer expires, when an interrupt occurs, etc. For the channel, typical events are insertion of a new frame onto the channel by a protocol machine, delivery of a frame to a protocol machine, or loss of a frame due to noise. Given a complete description of the protocol machines and the channel characteristics, it is possible to draw a directed graph showing all the states as nodes and all the transitions as directed arcs.

One particular state is designated as the initial state. This state corresponds to the description of the system when it starts running, or at some convenient starting place shortly thereafter. From the initial state, some, perhaps all, of the other states can be reached by a sequence of transitions. Using well-known techniques from graph theory (e.g., computing the transitive closure of a graph), it is possible to determine which states are reachable and which are not. This technique is called reachability analysis (Lin et al., 1987). This analysis can be helpful in determining whether a protocol is correct.

3.6 Example Data Link Protocols

In the following sections we will examine several widely-used data link protocols. The first one, HDLC, is a classical bit-oriented protocol whose variants have been in use for decades in many applications. The second one, PPP, is the data link protocol used to connect home computers to the Internet.

3.6.1 HDLC—High-Level Data Link Control

In this section we will examine a group of closely related protocols that are a bit old but are still heavily used. They are all derived from the data link protocol first used in the IBM mainframe world: SDLC (Synchronous Data Link Control) protocol. After developing SDLC, IBM submitted it to ANSI and ISO for acceptance as U.S. and international standards, respectively. ANSI modified it to become ADCCP (Advanced Data Communication Control Procedure), and ISO modified it to become HDLC (High-level Data Link Control). CCITT then adopted and modified HDLC for its LAP (Link Access Procedure) as part of the X.25 network interface standard but later modified it again to LAPB, to make it more compatible with a later version of HDLC. The nice thing about standards is that you have so many to choose from. Furthermore, if you do not like any of them, you can just wait for next year's model.

These protocols are based on the same principles. All are bit oriented, and all use bit stuffing for data transparency. They differ only in minor, but nevertheless irritating, ways. The discussion of bit-oriented protocols that follows is intended as a general introduction. For the specific details of any one protocol, please consult the appropriate definition.

All the bit-oriented protocols use the frame structure shown in Fig. 3-24. The Address field is primarily of importance on lines with multiple terminals, where it is used to identify one of the terminals. For point-to-point lines, it is sometimes used to distinguish commands from responses.

Figure 3-24. Frame format for bit-oriented protocols.

The Control field is used for sequence numbers, acknowledgements, and other purposes, as discussed below.

The Data field may contain any information. It may be arbitrarily long, although the efficiency of the checksum falls off with increasing frame length due to the greater probability of multiple burst errors.

The Checksum field is a cyclic redundancy code using the technique we examined in Sec. 3-2.2.

The frame is delimited with another flag sequence (01111110). On idle point-to-point lines, flag sequences are transmitted continuously. The minimum frame contains three fields and totals 32 bits, excluding the flags on either end.

3.7 Summary

The task of the data link layer is to convert the raw bit stream offered by the physical layer into a stream of frames for use by the network layer. Various framing methods are used, including character count, byte stuffing, and bit stuffing. Data link protocols can provide error control to retransmit damaged or lost frames. To prevent a fast sender from overrunning a slow receiver, the data link protocol can also provide flow control. The sliding window mechanism is widely used to integrate error control and flow control in a convenient way.

Sliding window protocols can be categorized by the size of the sender's window and the size of the receiver's window. When both are equal to 1, the protocol is stop-and-wait. When the sender's window is greater than 1, for example, to prevent the sender from blocking on a circuit with a long propagation delay, the receiver can be programmed either to discard all frames other than the next one in sequence or to buffer out-of-order frames until they are needed.

We examined a series of protocols in this chapter. Protocol 1 is designed for an error-free environment in which the receiver can handle any flow sent to it. Protocol 2 still assumes an error-free environment but introduces flow control. Protocol 3 handles errors by introducing sequence numbers and using the stop-and-wait algorithm. Protocol 4 allows bidirectional communication and introduces the concept of piggybacking. Protocol 5 uses a sliding window protocol with go back n. Finally, protocol 6 uses selective repeat and negative acknowledgements.

Protocols can be modeled using various techniques to help demonstrate their correctness (or lack thereof). Finite state machine models and Petri net models are commonly used for this purpose.

Many networks use one of the bit-oriented protocols—SDLC, HDLC, ADCCP, or LAPB—at the data link level. All of these protocols use flag bytes to delimit frames, and bit stuffing to prevent flag bytes from occurring in the data. All of them also use a sliding window for flow control. The Internet uses PPP as the primary data link protocol over point-to-point lines.

Problems

1.

1. An upper-layer packet is split into 10 frames, each of which has an 80 percent chance of arriving undamaged. If no error control is done by the data link protocol, how many times must the message be sent on average to get the entire thing through?
2.

2. The following character encoding is used in a data link protocol: A: 01000111; B: 11100011; FLAG: 01111110; ESC: 11100000 Show the bit sequence transmitted (in binary) for the four-character frame: A B ESC FLAG when each of the following framing methods are used:
   a.

   a. (a) Character count.
   b.

   b. (b) Flag bytes with byte stuffing.
   c.

Chapter 4. The Medium Access Control Sublayer

As we pointed out in Chap. 1, networks can be divided into two categories: those using point-to-point connections and those using broadcast channels. This chapter deals with broadcast networks and their protocols.

In any broadcast network, the key issue is how to determine who gets to use the channel when there is competition for it. To make this point clearer, consider a conference call in which six people, on six different telephones, are all connected so that each one can hear and talk to all the others. It is very likely that when one of them stops speaking, two or more will start talking at once, leading to chaos. In a face-to-face meeting, chaos is avoided by external means, for example, at a meeting, people raise their hands to request permission to speak. When only a single channel is available, determining who should go next is much harder. Many protocols for solving the problem are known and form the contents of this chapter. In the literature, broadcast channels are sometimes referred to as multiaccess channels or random access channels.

The protocols used to determine who goes next on a multiaccess channel belong to a sublayer of the data link layer called the MAC (Medium Access Control) sublayer. The MAC sublayer is especially important in LANs, many of which use a multiaccess channel as the basis for communication. WANs, in contrast, use point-to-point links, except for satellite networks. Because multiaccess channels and LANs are so closely related, in this chapter we will discuss LANs in general, including a few issues that are not strictly part of the MAC sublayer.

Technically, the MAC sublayer is the bottom part of the data link layer, so logically we should have studied it before examining all the point-to-point protocols in Chap. 3. Nevertheless, for most people, understanding protocols involving multiple parties is easier after two-party protocols are well understood. For that reason we have deviated slightly from a strict bottom-up order of presentation.

4.1 The Channel Allocation Problem

The central theme of this chapter is how to allocate a single broadcast channel among competing users. We will first look at static and dynamic schemes in general. Then we will examine a number of specific algorithms.

4.1.1 Static Channel Allocation in LANs and MANs

The traditional way of allocating a single channel, such as a telephone trunk, among multiple competing users is Frequency Division Multiplexing (FDM). If there are N users, the bandwidth is divided into N equal-sized portions (see Fig. 2-31), each user being assigned one portion. Since each user has a private frequency band, there is no interference between users. When there is only a small and constant number of users, each of which has a heavy (buffered) load of traffic (e.g., carriers' switching offices), FDM is a simple and efficient allocation mechanism.

However, when the number of senders is large and continuously varying or the traffic is bursty, FDM presents some problems. If the spectrum is cut up into N regions and fewer than N users are currently interested in communicating, a large piece of valuable spectrum will be wasted. If more than N users want to communicate, some of them will be denied permission for lack of bandwidth, even if some of the users who have been assigned a frequency band hardly ever transmit or receive anything.

However, even assuming that the number of users could somehow be held constant at N, dividing the single available channel into static subchannels is inherently inefficient. The basic problem is that when some users are quiescent, their bandwidth is simply lost. They are not using it, and no one else is allowed to use it either. Furthermore, in most computer systems, data traffic is extremely bursty (peak traffic to mean traffic ratios of 1000:1 are common). Consequently, most of the channels will be idle most of the time.

The poor performance of static FDM can easily be seen from a simple queueing theory calculation. Let us start with the mean time delay, T, for a channel of capacity C bps, with an arrival rate of ? frames/sec, each frame having a length drawn from an exponential probability density function with mean 1/ bits/frame. With these parameters the arrival rate is ? frames/sec and the service rate is C frames/sec. From queueing theory it can be shown that for Poisson arrival and service times,

$$T = \frac{1}{\mu C - \lambda}$$

For example, if C is 100 Mbps, the mean frame length, 1/ , is 10,000 bits, and the frame arrival rate, ?, is 5000 frames/sec, then T = 200 sec. Note that if we ignored the queueing delay and just asked how long it takes to send a 10,000 bit frame on a 100-Mbps network, we would get the (incorrect) answer of 100 sec. That result only holds when there is no contention for the channel.

Now let us divide the single channel into N independent subchannels, each with capacity C/N bps. The mean input rate on each of the subchannels will now be ?/N. Recomputing T we get

Equation 4

4.2 Multiple Access Protocols

Many algorithms for allocating a multiple access channel are known. In the following sections we will study a small sample of the more interesting ones and give some examples of their use.

4.2.1 ALOHA

In the 1970s, Norman Abramson and his colleagues at the University of Hawaii devised a new and elegant method to solve the channel allocation problem. Their work has been extended by many researchers since then (Abramson, 1985). Although Abramson's work, called the ALOHA system, used ground-based radio broadcasting, the basic idea is applicable to any system in which uncoordinated users are competing for the use of a single shared channel.

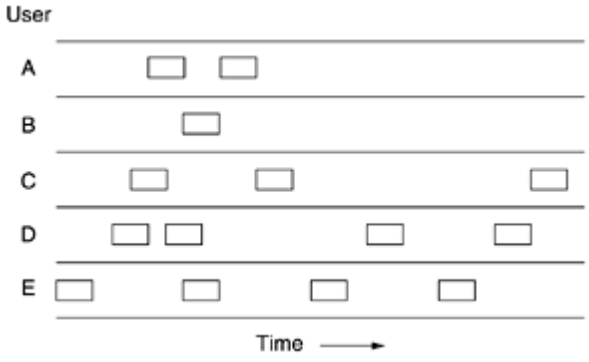
We will discuss two versions of ALOHA here: pure and slotted. They differ with respect to whether time is divided into discrete slots into which all frames must fit. Pure ALOHA does not require global time synchronization; slotted ALOHA does.

Pure ALOHA

The basic idea of an ALOHA system is simple: let users transmit whenever they have data to be sent. There will be collisions, of course, and the colliding frames will be damaged. However, due to the feedback property of broadcasting, a sender can always find out whether its frame was destroyed by listening to the channel, the same way other users do. With a LAN, the feedback is immediate; with a satellite, there is a delay of 270 msec before the sender knows if the transmission was successful. If listening while transmitting is not possible for some reason, acknowledgements are needed. If the frame was destroyed, the sender just waits a random amount of time and sends it again. The waiting time must be random or the same frames will collide over and over, in lockstep. Systems in which multiple users share a common channel in a way that can lead to conflicts are widely known as contention systems.

A sketch of frame generation in an ALOHA system is given in Fig. 4-1. We have made the frames all the same length because the throughput of ALOHA systems is maximized by having a uniform frame size rather than by allowing variable length frames.

Figure 4-1. In pure ALOHA, frames are transmitted at completely arbitrary times.



Whenever two frames try to occupy the channel at the same time, there will be a collision and both will be garbled. If the first bit of a new frame overlaps with just the last bit of a frame almost finished, both frames will be totally

4.3 Ethernet

We have now finished our general discussion of channel allocation protocols in the abstract, so it is time to see how these principles apply to real systems, in particular, LANs. As discussed in Sec. 1.5.3, the IEEE has standardized a number of local area networks and metropolitan area networks under the name of IEEE 802. A few have survived but many have not, as we saw in Fig. 1-38. Some people who believe in reincarnation think that Charles Darwin came back as a member of the IEEE Standards Association to weed out the unfit. The most important of the survivors are 802.3 (Ethernet) and 802.11 (wireless LAN). With 802.15 (Bluetooth) and 802.16 (wireless MAN), it is too early to tell. Please consult the 5th edition of this book to find out. Both 802.3 and 802.11 have different physical layers and different MAC sublayers but converge on the same logical link control sublayer (defined in 802.2), so they have the same interface to the network layer.

We introduced Ethernet in Sec. 1.5.3 and will not repeat that material here. Instead we will focus on the technical details of Ethernet, the protocols, and recent developments in high-speed (gigabit) Ethernet. Since Ethernet and IEEE 802.3 are identical except for two minor differences that we will discuss shortly, many people use the terms "Ethernet" and "IEEE 802.3" interchangeably, and we will do so, too. For more information about Ethernet, see (Breyer and Riley, 1999 ; Seifert, 1998; and Spurgeon, 2000).

4.3.1 Ethernet Cabling

Since the name "Ethernet" refers to the cable (the ether), let us start our discussion there. Four types of cabling are commonly used, as shown in Fig. 4-13.

Figure 4-13. The most common kinds of Ethernet cabling.

Name	Cable	Max. seg.	Nodes/seg.	Advantages
10Base5	Thick coax	500 m	100	Original cable; now obsolete
10Base2	Thin coax	185 m	30	No hub needed
10Base-T	Twisted pair	100 m	1024	Cheapest system
10Base-F	Fiber optics	2000 m	1024	Best between buildings

Historically, 10Base5 cabling, popularly called thick Ethernet, came first. It resembles a yellow garden hose, with markings every 2.5 meters to show where the taps go. (The 802.3 standard does not actually require the cable to be yellow, but it does suggest it.) Connections to it are generally made using vampire taps, in which a pin is very carefully forced halfway into the coaxial cable's core. The notation 10Base5 means that it operates at 10 Mbps, uses baseband signaling, and can support segments of up to 500 meters. The first number is the speed in Mbps. Then comes the word "Base" (or sometimes "BASE") to indicate baseband transmission. There used to be a broadband variant, 10Broad36, but it never caught on in the marketplace and has since vanished. Finally, if the medium is coax, its length is given rounded to units of 100 m after "Base."

Historically, the second cable type was 10Base2, or thin Ethernet, which, in contrast to the garden-hose-like thick Ethernet, bends easily. Connections to it are made using industry-standard BNC connectors to form T junctions, rather than using vampire taps. BNC connectors are easier to use and more reliable. Thin Ethernet is much cheaper and easier to install, but it can run for only 185 meters per segment, each of which can handle only 30 machines.

Detecting cable breaks, excessive length, bad taps, or loose connectors can be a major problem with both media.

4.4 Wireless LANs

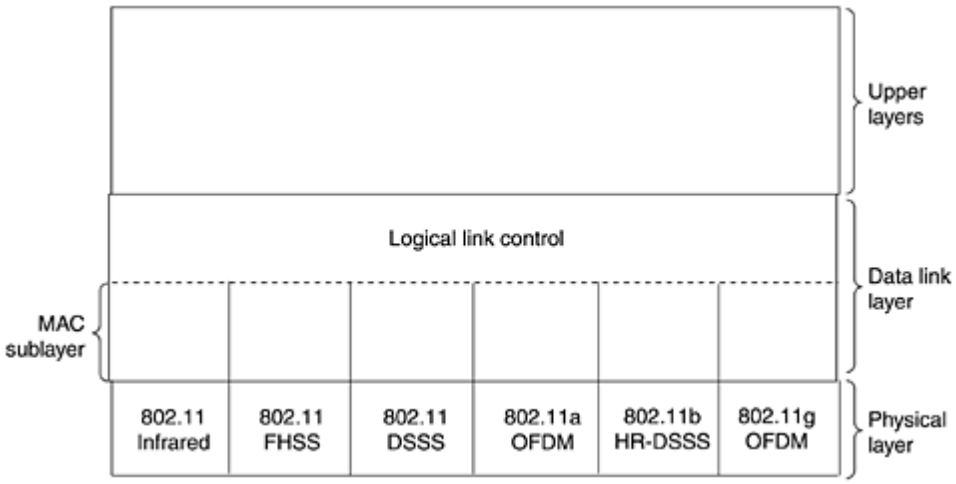
Although Ethernet is widely used, it is about to get some competition. Wireless LANs are increasingly popular, and more and more office buildings, airports, and other public places are being outfitted with them. Wireless LANs can operate in one of two configurations, as we saw in Fig. 1-35: with a base station and without a base station. Consequently, the 802.11 LAN standard takes this into account and makes provision for both arrangements, as we will see shortly.

We gave some background information on 802.11 in Sec. 1.5.4. Now is the time to take a closer look at the technology. In the following sections we will look at the protocol stack, physical layer radio transmission techniques, MAC sublayer protocol, frame structure, and services. For more information about 802.11, see (Crow et al., 1997; Geier, 2002; Heegard et al., 2001; Kapp, 2002; O'Hara and Petrick, 1999; and Severance, 1999). To hear the truth from the mouth of the horse, consult the published 802.11 standard itself.

4.4.1 The 802.11 Protocol Stack

The protocols used by all the 802 variants, including Ethernet, have a certain commonality of structure. A partial view of the 802.11 protocol stack is given in Fig. 4-25. The physical layer corresponds to the OSI physical layer fairly well, but the data link layer in all the 802 protocols is split into two or more sublayers. In 802.11, the MAC (Medium Access Control) sublayer determines how the channel is allocated, that is, who gets to transmit next. Above it is the LLC (Logical Link Control) sublayer, whose job it is to hide the differences between the different 802 variants and make them indistinguishable as far as the network layer is concerned. We studied the LLC when examining Ethernet earlier in this chapter and will not repeat that material here.

Figure 4-25. Part of the 802.11 protocol stack.



The 1997 802.11 standard specifies three transmission techniques allowed in the physical layer. The infrared method uses much the same technology as television remote controls do. The other two use short-range radio, using techniques called FHSS and DSSS. Both of these use a part of the spectrum that does not require licensing (the 2.4-GHz ISM band). Radio-controlled garage door openers also use this piece of the spectrum, so your notebook computer may find itself in competition with your garage door. Cordless telephones and microwave ovens also use this band. All of these techniques operate at 1 or 2 Mbps and at low enough power that they do not conflict too much. In 1999, two new techniques were introduced to achieve higher bandwidth. These are called OFDM and HR-DSSS. They operate at up to 54 Mbps and 11 Mbps, respectively. In 2001, a second OFDM modulation was introduced, but in a different frequency band from the first one. Now we will examine each of them briefly.

4.5 Broadband Wireless

We have been indoors too long. Let us now go outside and see if any interesting networking is going on there. It turns out that quite a bit is going on there, and some of it has to do with the so-called last mile. With the deregulation of the telephone system in many countries, competitors to the entrenched telephone company are now often allowed to offer local voice and high-speed Internet service. There is certainly plenty of demand. The problem is that running fiber, coax, or even category 5 twisted pair to millions of homes and businesses is prohibitively expensive. What is a competitor to do?

The answer is broadband wireless. Erecting a big antenna on a hill just outside of town and installing antennas directed at it on customers' roofs is much easier and cheaper than digging trenches and stringing cables. Thus, competing telecommunication companies have a great interest in providing a multimegabit wireless communication service for voice, Internet, movies on demand, etc. As we saw in Fig. 2-30, LMDS was invented for this purpose. However, until recently, every carrier devised its own system. This lack of standards meant that hardware and software could not be mass produced, which kept prices high and acceptance low.

Many people in the industry realized that having a broadband wireless standard was the key element missing, so IEEE was asked to form a committee composed of people from key companies and academia to draw up the standard. The next number available in the 802 numbering space was 802.16, so the standard got this number. Work was started in July 1999, and the final standard was approved in April 2002. Officially the standard is called "Air Interface for Fixed Broadband Wireless Access Systems." However, some people prefer to call it a wireless MAN (Metropolitan Area Network) or a wireless local loop. We regard all these terms as interchangeable.

Like some of the other 802 standards, 802.16 was heavily influenced by the OSI model, including the (sub)layers, terminology, service primitives, and more. Unfortunately, also like OSI, it is fairly complicated. In the following sections we will give a brief description of some of the highlights of 802.16, but this treatment is far from complete and leaves out many details. For additional information about broadband wireless in general, see (Bolcskei et al., 2001; and Webb, 2001). For information about 802.16 in particular, see (Eklund et al., 2002).

4.5.1 Comparison of 802.11 with 802.16

At this point you may be thinking: Why devise a new standard? Why not just use 802.11? There are some very good reasons for not using 802.11, primarily because 802.11 and 802.16 solve different problems. Before getting into the technology of 802.16, it is probably worthwhile saying a few words about why a new standard is needed at all.

The environments in which 802.11 and 802.16 operate are similar in some ways, primarily in that they were designed to provide high-bandwidth wireless communications. But they also differ in some major ways. To start with, 802.16 provides service to buildings, and buildings are not mobile. They do not migrate from cell to cell often. Much of 802.11 deals with mobility, and none of that is relevant here. Next, buildings can have more than one computer in them, a complication that does not occur when the end station is a single notebook computer. Because building owners are generally willing to spend much more money for communication gear than are notebook owners, better radios are available. This difference means that 802.16 can use full-duplex communication, something 802.11 avoids to keep the cost of the radios low.

Because 802.16 runs over part of a city, the distances involved can be several kilometers, which means that the

4.6 Bluetooth

In 1994, the L. M. Ericsson company became interested in connecting its mobile phones to other devices (e.g., PDAs) without cables. Together with four other companies (IBM, Intel, Nokia, and Toshiba), it formed a SIG (Special Interest Group, i.e., consortium) to develop a wireless standard for interconnecting computing and communication devices and accessories using short-range, low-power, inexpensive wireless radios. The project was named Bluetooth, after Harald Blaatand (Bluetooth) II (940-981), a Viking king who unified (i.e., conquered) Denmark and Norway, also without cables.

Although the original idea was just to get rid of the cables between devices, it soon began to expand in scope and encroach on the area of wireless LANs. While this move makes the standard more useful, it also creates some competition for mindshare with 802.11. To make matters worse, the two systems also interfere with each other electrically. It is also worth noting that Hewlett-Packard introduced an infrared network for connecting computer peripherals without wires some years ago, but it never really caught on in a big way.

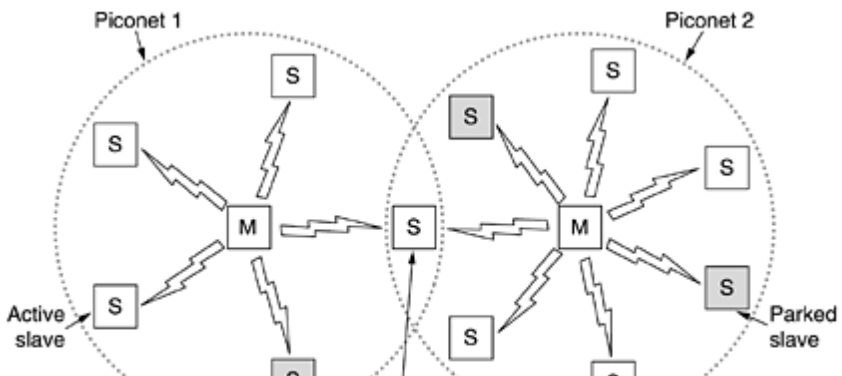
Undaunted by all this, in July 1999 the Bluetooth SIG issued a 1500-page specification of V1.0. Shortly thereafter, the IEEE standards group looking at wireless personal area networks, 802.15, adopted the Bluetooth document as a basis and began hacking on it. While it might seem strange to standardize something that already had a very detailed specification and no incompatible implementations that needed to be harmonized, history shows that having an open standard managed by a neutral body such as the IEEE often promotes the use of a technology. To be a bit more precise, it should be noted that the Bluetooth specification is for a complete system, from the physical layer to the application layer. The IEEE 802.15 committee is standardizing only the physical and data link layers; the rest of the protocol stack falls outside its charter.

Even though IEEE approved the first PAN standard, 802.15.1, in 2002, the Bluetooth SIG is still active busy with improvements. Although the Bluetooth SIG and IEEE versions are not identical, it is hoped that they will soon converge to a single standard.

4.6.1 Bluetooth Architecture

Let us start our study of the Bluetooth system with a quick overview of what it contains and what it is intended to do. The basic unit of a Bluetooth system is a piconet, which consists of a master node and up to seven active slave nodes within a distance of 10 meters. Multiple piconets can exist in the same (large) room and can even be connected via a bridge node, as shown in Fig. 4-35. An interconnected collection of piconets is called a scatternet.

Figure 4-35. Two piconets can be connected to form a scatternet.



4.7 Data Link Layer Switching

Many organizations have multiple LANs and wish to connect them. LANs can be connected by devices called bridges, which operate in the data link layer. Bridges examine the data layer link addresses to do routing. Since they are not supposed to examine the payload field of the frames they route, they can transport IPv4 (used in the Internet now), IPv6 (will be used in the Internet in the future), AppleTalk, ATM, OSI, or any other kinds of packets. In contrast, routers examine the addresses in packets and route based on them. Although this seems like a clear division between bridges and routers, some modern developments, such as the advent of switched Ethernet, have muddied the waters, as we will see later. In the following sections we will look at bridges and switches, especially for connecting different 802 LANs. For a comprehensive treatment of bridges, switches, and related topics, see (Perlman, 2000).

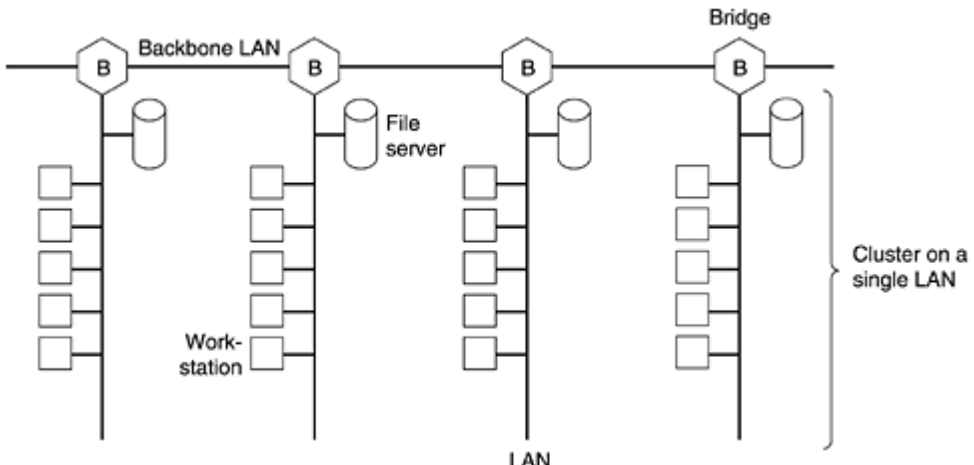
Before getting into the technology of bridges, it is worthwhile taking a look at some common situations in which bridges are used. We will mention six reasons why a single organization may end up with multiple LANs.

First, many university and corporate departments have their own LANs, primarily to connect their own personal computers, workstations, and servers. Since the goals of the various departments differ, different departments choose different LANs, without regard to what other departments are doing. Sooner or later, there is a need for interaction, so bridges are needed. In this example, multiple LANs came into existence due to the autonomy of their owners.

Second, the organization may be geographically spread over several buildings separated by considerable distances. It may be cheaper to have separate LANs in each building and connect them with bridges and laser links than to run a single cable over the entire site.

Third, it may be necessary to split what is logically a single LAN into separate LANs to accommodate the load. At many universities, for example, thousands of workstations are available for student and faculty computing. Files are normally kept on file server machines and are downloaded to users' machines upon request. The enormous scale of this system precludes putting all the workstations on a single LAN—the total bandwidth needed is far too high. Instead, multiple LANs connected by bridges are used, as shown in Fig. 4-39. Each LAN contains a cluster of workstations with its own file server so that most traffic is restricted to a single LAN and does not add load to the backbone.

Figure 4-39. Multiple LANs connected by a backbone to handle a total load higher than the capacity of a single LAN.



4.8 Summary

Some networks have a single channel that is used for all communication. In these networks, the key design issue is the allocation of this channel among the competing stations wishing to use it. Numerous channel allocation algorithms have been devised. A summary of some of the more important channel allocation methods is given in Fig. 4-52.

Figure 4-52. Channel allocation methods and systems for a common channel.

Method	Description
FDM	Dedicate a frequency band to each station
WDM	A dynamic FDM scheme for fiber
TDM	Dedicate a time slot to each station
Pure ALOHA	Unsynchronized transmission at any instant
Slotted ALOHA	Random transmission in well-defined time slots
1-persistent CSMA	Standard carrier sense multiple access
Nonpersistent CSMA	Random delay when channel is sensed busy
P-persistent CSMA	CSMA, but with a probability of p of persisting
CSMA/CD	CSMA, but abort on detecting a collision
Bit map	Round-robin scheduling using a bit map
Binary countdown	Highest-numbered ready station goes next
Tree walk	Reduced contention by selective enabling
MACA, MACAW	Wireless LAN protocols
Ethernet	CSMA/CD with binary exponential backoff
FHSS	Frequency hopping spread spectrum
DSSS	Direct sequence spread spectrum
CSMA/CA	Carrier sense multiple access with collision avoidance

The simplest allocation schemes are FDM and TDM. These are efficient when the number of stations is small and fixed and the traffic is continuous. Both are widely used under these circumstances, for example, for dividing up the bandwidth on telephone trunks.

When the number of stations is large and variable or the traffic is fairly bursty, FDM and TDM are poor choices. The ALOHA protocol, with and without slotting, has been proposed as an alternative. ALOHA and its many variants and derivatives have been widely discussed, analyzed, and used in real systems.

When the state of the channel can be sensed, stations can avoid starting a transmission while another station is transmitting. This technique, carrier sensing, has led to a variety of protocols that can be used on LANs and MANs.

A class of protocols that eliminates contention altogether, or at least reduce it considerably, is well known. Binary countdown completely eliminates contention. The tree walk protocol reduces it by dynamically dividing the stations into two disjoint groups, one of which is permitted to transmit and one of which is not. It tries to make the division in such a way that only one station that is ready to send is permitted to do so.

Wireless LANs have their own problems and solutions. The biggest problem is caused by hidden stations, so CSMA does not work. One class of solutions, typified by MACA and MACAW, attempts to stimulate transmissions around the destination, to make CSMA work better. Frequency hopping spread spectrum and direct sequence spread spectrum are also used. IEEE 802.11 combines CSMA and MACAW to produce CSMA/CA.

Chapter 5. The Network Layer

The network layer is concerned with getting packets from the source all the way to the destination. Getting to the destination may require making many hops at intermediate routers along the way. This function clearly contrasts with that of the data link layer, which has the more modest goal of just moving frames from one end of a wire to the other. Thus, the network layer is the lowest layer that deals with end-to-end transmission.

To achieve its goals, the network layer must know about the topology of the communication subnet (i.e., the set of all routers) and choose appropriate paths through it. It must also take care to choose routes to avoid overloading some of the communication lines and routers while leaving others idle. Finally, when the source and destination are in different networks, new problems occur. It is up to the network layer to deal with them. In this chapter we will study all these issues and illustrate them, primarily using the Internet and its network layer protocol, IP, although wireless networks will also be addressed.

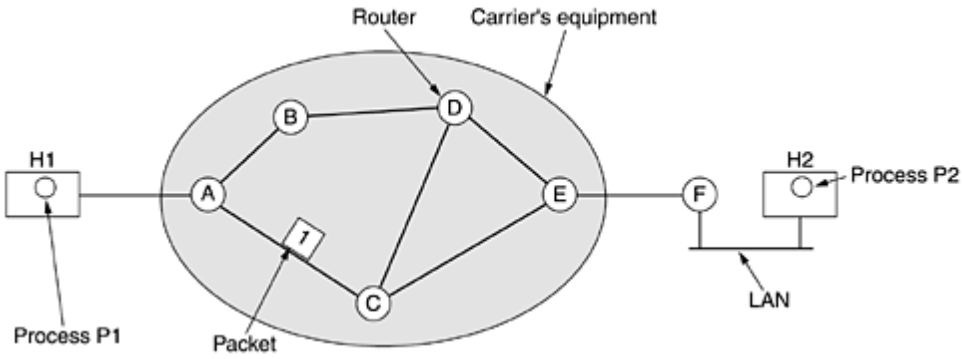
5.1 Network Layer Design Issues

In the following sections we will provide an introduction to some of the issues that the designers of the network layer must grapple with. These issues include the service provided to the transport layer and the internal design of the subnet.

5.1.1 Store-and-Forward Packet Switching

But before starting to explain the details of the network layer, it is probably worth restating the context in which the network layer protocols operate. This context can be seen in Fig. 5-1. The major components of the system are the carrier's equipment (routers connected by transmission lines), shown inside the shaded oval, and the customers' equipment, shown outside the oval. Host H1 is directly connected to one of the carrier's routers, A, by a leased line. In contrast, H2 is on a LAN with a router, F, owned and operated by the customer. This router also has a leased line to the carrier's equipment. We have shown F as being outside the oval because it does not belong to the carrier, but in terms of construction, software, and protocols, it is probably no different from the carrier's routers. Whether it belongs to the subnet is arguable, but for the purposes of this chapter, routers on customer premises are considered part of the subnet because they run the same algorithms as the carrier's routers (and our main concern here is algorithms).

Figure 5-1. The environment of the network layer protocols.



This equipment is used as follows. A host with a packet to send transmits it to the nearest router, either on its own LAN or over a point-to-point link to the carrier. The packet is stored there until it has fully arrived so the checksum can be verified. Then it is forwarded to the next router along the path until it reaches the destination host, where it is delivered. This mechanism is store-and-forward packet switching, as we have seen in previous chapters.

5.1.2 Services Provided to the Transport Layer

The network layer provides services to the transport layer at the network layer/transport layer interface. An important question is what kind of services the network layer provides to the transport layer. The network layer services have been designed with the following goals in mind.

1.

1. The services should be independent of the router technology.
2.

2. The transport layer should be shielded from the number, type, and topology of the routers present.

5.2 Routing Algorithms

The main function of the network layer is routing packets from the source machine to the destination machine. In most subnets, packets will require multiple hops to make the journey. The only notable exception is for broadcast networks, but even here routing is an issue if the source and destination are not on the same network. The algorithms that choose the routes and the data structures that they use are a major area of network layer design.

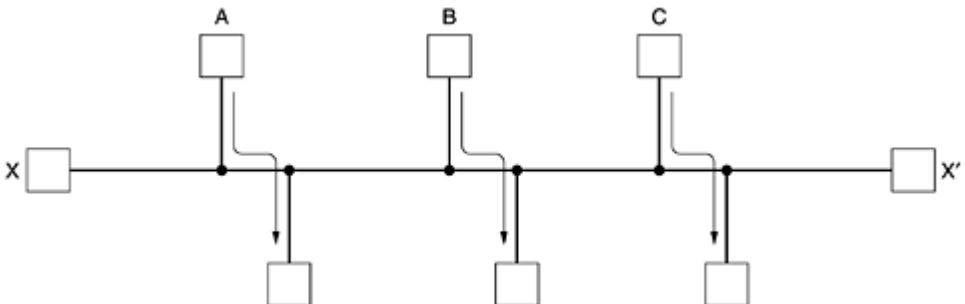
The routing algorithm is that part of the network layer software responsible for deciding which output line an incoming packet should be transmitted on. If the subnet uses datagrams internally, this decision must be made anew for every arriving data packet since the best route may have changed since last time. If the subnet uses virtual circuits internally, routing decisions are made only when a new virtual circuit is being set up. Thereafter, data packets just follow the previously-established route. The latter case is sometimes called session routing because a route remains in force for an entire user session (e.g., a login session at a terminal or a file transfer).

It is sometimes useful to make a distinction between routing, which is making the decision which routes to use, and forwarding, which is what happens when a packet arrives. One can think of a router as having two processes inside it. One of them handles each packet as it arrives, looking up the outgoing line to use for it in the routing tables. This process is forwarding. The other process is responsible for filling in and updating the routing tables. That is where the routing algorithm comes into play.

Regardless of whether routes are chosen independently for each packet or only when new connections are established, certain properties are desirable in a routing algorithm: correctness, simplicity, robustness, stability, fairness, and optimality. Correctness and simplicity hardly require comment, but the need for robustness may be less obvious at first. Once a major network comes on the air, it may be expected to run continuously for years without systemwide failures. During that period there will be hardware and software failures of all kinds. Hosts, routers, and lines will fail repeatedly, and the topology will change many times. The routing algorithm should be able to cope with changes in the topology and traffic without requiring all jobs in all hosts to be aborted and the network to be rebooted every time some router crashes.

Stability is also an important goal for the routing algorithm. There exist routing algorithms that never converge to equilibrium, no matter how long they run. A stable algorithm reaches equilibrium and stays there. Fairness and optimality may sound obvious—surely no reasonable person would oppose them—but as it turns out, they are often contradictory goals. As a simple example of this conflict, look at Fig. 5-5. Suppose that there is enough traffic between A and A', between B and B', and between C and C' to saturate the horizontal links. To maximize the total flow, the X to X' traffic should be shut off altogether. Unfortunately, X and X' may not see it that way. Evidently, some compromise between global efficiency and fairness to individual connections is needed.

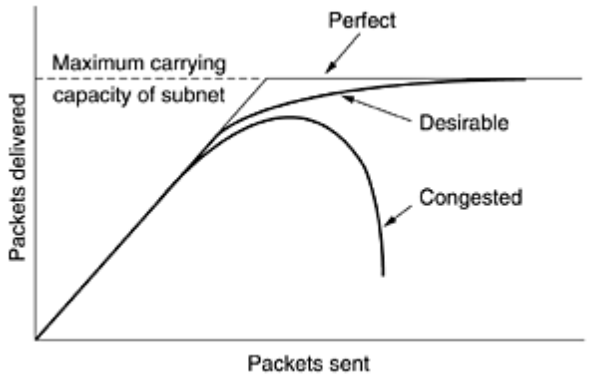
Figure 5-5. Conflict between fairness and optimality.



5.3 Congestion Control Algorithms

When too many packets are present in (a part of) the subnet, performance degrades. This situation is called congestion. Figure 5-25 depicts the symptom. When the number of packets dumped into the subnet by the hosts is within its carrying capacity, they are all delivered (except for a few that are afflicted with transmission errors) and the number delivered is proportional to the number sent. However, as traffic increases too far, the routers are no longer able to cope and they begin losing packets. This tends to make matters worse. At very high trafffic, performance collapses completely and almost no packets are delivered.

Figure 5-25. When too much traffic is offered, congestion sets in and performance degrades sharply.



Congestion can be brought on by several factors. If all of a sudden, streams of packets begin arriving on three or four input lines and all need the same output line, a queue will build up. If there is insufficient memory to hold all of them, packets will be lost. Adding more memory may help up to a point, but Nagle (1987) discovered that if routers have an infinite amount of memory, congestion gets worse, not better, because by the time packets get to the front of the queue, they have already timed out (repeatedly) and duplicates have been sent. All these packets will be dutifully forwarded to the next router, increasing the load all the way to the destination.

Slow processors can also cause congestion. If the routers' CPUs are slow at performing the bookkeeping tasks required of them (queueing buffers, updating tables, etc.), queues can build up, even though there is excess line capacity. Similarly, low-bandwidth lines can also cause congestion. Upgrading the lines but not changing the processors, or vice versa, often helps a little, but frequently just shifts the bottleneck. Also, upgrading part, but not all, of the system, often just moves the bottleneck somewhere else. The real problem is frequently a mismatch between parts of the system. This problem will persist until all the components are in balance.

It is worth explicitly pointing out the difference between congestion control and flow control, as the relationship is subtle. Congestion control has to do with making sure the subnet is able to carry the offered traffic. It is a global issue, involving the behavior of all the hosts, all the routers, the store-and-forwarding processing within the routers, and all the other factors that tend to diminish the carrying capacity of the subnet.

Flow control, in contrast, relates to the point-to-point traffic between a given sender and a given receiver. Its job is to make sure that a fast sender cannot continually transmit data faster than the receiver is able to absorb it. Flow control frequently involves some direct feedback from the receiver to the sender to tell the sender how things are doing at the other end.

To see the difference between these two concepts, consider a fiber optic network with a capacity of 1000

5.4 Quality of Service

The techniques we looked at in the previous sections are designed to reduce congestion and improve network performance. However, with the growth of multimedia networking, often these ad hoc measures are not enough. Serious attempts at guaranteeing quality of service through network and protocol design are needed. In the following sections we will continue our study of network performance, but now with a sharper focus on ways to provide a quality of service matched to application needs. It should be stated at the start, however, that many of these ideas are in flux and are subject to change.

5.4.1 Requirements

A stream of packets from a source to a destination is called a flow.Ina connection-oriented network, all the packets belonging to a flow follow the same route; in a connectionless network, they may follow different routes. The needs of each flow can be characterized by four primary parameters: reliability, delay, jitter, and bandwidth. Together these determine the QoS (Quality of Service) the flow requires. Several common applications and the stringency of their requirements are listed in Fig. 5-30.

Figure 5-30. How stringent the quality-of-service requirements are.

Application	Reliability	Delay	Jitter	Bandwidth
E-mail	High	Low	Low	Low
File transfer	High	Low	Low	Medium
Web access	High	Medium	Low	Medium
Remote login	High	Medium	Medium	Low
Audio on demand	Low	Low	High	Medium
Video on demand	Low	Low	High	High
Telephony	Low	High	High	Low
Videoconferencing	Low	High	High	High

The first four applications have stringent requirements on reliability. No bits may be delivered incorrectly. This goal is usually achieved by checksumming each packet and verifying the checksum at the destination. If a packet is damaged in transit, it is not acknowledged and will be retransmitted eventually. This strategy gives high reliability. The four final (audio/video) applications can tolerate errors, so no checksums are computed or verified.

File transfer applications, including e-mail and video, are not delay sensitive. If all packets are delayed uniformly by a few seconds, no harm is done. Interactive applications, such as Web surfing and remote login, are more delay sensitive. Real-time applications, such as telephony and videoconferencing have strict delay requirements. If all the words in a telephone call are each delayed by exactly 2.000 seconds, the users will find the connection unacceptable. On the other hand, playing audio or video files from a server does not require low delay.

The first three applications are not sensitive to the packets arriving with irregular time intervals between them. Remote login is somewhat sensitive to that, since characters on the screen will appear in little bursts if the connection suffers much jitter. Video and especially audio are extremely sensitive to jitter. If a user is watching a video over the network and the frames are all delayed by exactly 2.000 seconds, no harm is done. But if the transmission time varies randomly between 1 and 2 seconds, the result will be terrible. For audio, a jitter of even a few milliseconds is clearly audible.

5.5 Internetworking

Until now, we have implicitly assumed that there is a single homogeneous network, with each machine using the same protocol in each layer. Unfortunately, this assumption is wildly optimistic. Many different networks exist, including LANs, MANs, and WANs. Numerous protocols are in widespread use in every layer. In the following sections we will take a careful look at the issues that arise when two or more networks are connected to form an internet.

Considerable controversy exists about the question of whether today's abundance of network types is a temporary condition that will go away as soon as everyone realizes how wonderful [fill in your favorite network] is or whether it is an inevitable, but permanent, feature of the world that is here to stay. Having different networks invariably means having different protocols.

We believe that a variety of different networks (and thus protocols) will always be around, for the following reasons. First of all, the installed base of different networks is large. Nearly all personal computers run TCP/IP. Many large businesses have mainframes running IBM's SNA. A substantial number of telephone companies operate ATM networks. Some personal computer LANs still use Novell NCP/IPX or AppleTalk. Finally, wireless is an up-and-coming area with a variety of protocols. This trend will continue for years due to legacy problems, new technology, and the fact that not all vendors perceive it in their interest for their customers to be able to easily migrate to another vendor's system.

Second, as computers and networks get cheaper, the place where decisions get made moves downward in organizations. Many companies have a policy to the effect that purchases costing over a million dollars have to be approved by top management, purchases costing over 100,000 dollars have to be approved by middle management, but purchases under 100,000 dollars can be made by department heads without any higher approval. This can easily lead to the engineering department installing UNIX workstations running TCP/IP and the marketing department installing Macs with AppleTalk.

Third, different networks (e.g., ATM and wireless) have radically different technology, so it should not be surprising that as new hardware developments occur, new software will be created to fit the new hardware. For example, the average home now is like the average office ten years ago: it is full of computers that do not talk to one another. In the future, it may be commonplace for the telephone, the television set, and other appliances all to be networked together so that they can be controlled remotely. This new technology will undoubtedly bring new networks and new protocols.

As an example of how different networks might be connected, consider the example of Fig. 5-42. Here we see a corporate network with multiple locations tied together by a wide area ATM network. At one of the locations, an FDDI optical backbone is used to connect an Ethernet, an 802.11 wireless LAN, and the corporate data center's SNA mainframe network.

Figure 5-42. A collection of interconnected networks.

5.6 The Network Layer in the Internet

Before getting into the specifics of the network layer in the Internet, it is worth taking at look at the principles that drove its design in the past and made it the success that it is today. All too often, nowadays, people seem to have forgotten them. These principles are enumerated and discussed in RFC 1958, which is well worth reading (and should be mandatory for all protocol designers—with a final exam at the end). This RFC draws heavily on ideas found in (Clark, 1988; and Saltzer et al., 1984). We will now summarize what we consider to be the top 10 principles (from most important to least important).

1.

1. **Make sure it works.** Do not finalize the design or standard until multiple prototypes have successfully communicated with each other. All too often designers first write a 1000-page standard, get it approved, then discover it is deeply flawed and does not work. Then they write version 1.1 of the standard. This is not the way to go.
2.

2. **Keep it simple.** When in doubt, use the simplest solution. William of Occam stated this principle (Occam's razor) in the 14th century. Put in modern terms: fight features. If a feature is not absolutely essential, leave it out, especially if the same effect can be achieved by combining other features.
3.

3. **Make clear choices.** If there are several ways of doing the same thing, choose one. Having two or more ways to do the same thing is looking for trouble. Standards often have multiple options or modes or parameters because several powerful parties insist that their way is best. Designers should strongly resist this tendency. Just say no.
4.

4. **Exploit modularity.** This principle leads directly to the idea of having protocol stacks, each of whose layers is independent of all the other ones. In this way, if circumstances that require one module or layer to be changed, the other ones will not be affected.
5.

5. **Expect heterogeneity.** Different types of hardware, transmission facilities, and applications will occur on any large network. To handle them, the network design must be simple, general, and flexible.
6.

6. **Avoid static options and parameters.** If parameters are unavoidable (e.g., maximum packet size), it is best to have the sender and receiver negotiate a value than defining fixed choices.
7.

7. **Look for a good design; it need not be perfect.** Often the designers have a good design but it cannot handle some weird special case. Rather than messing up the design, the designers should go with the good design and put the burden of working around it on the people with the strange requirements.
8.

8. **Be strict when sending and tolerant when receiving.** In other words, only send packets that rigorously comply with the standards, but expect incoming packets that may not be fully conformant and try to deal with them.
9.

9. **Think about scalability.** If the system is to handle millions of hosts and billions of users effectively, no

5.7 Summary

The network layer provides services to the transport layer. It can be based on either virtual circuits or datagrams. In both cases, its main job is routing packets from the source to the destination. In virtual-circuit subnets, a routing decision is made when the virtual circuit is set up. In datagram subnets, it is made on every packet.

Many routing algorithms are used in computer networks. Static algorithms include shortest path routing and flooding. Dynamic algorithms include distance vector routing and link state routing. Most actual networks use one of these. Other important routing topics are hierarchical routing, routing for mobile hosts, broadcast routing, multicast routing, and routing in peer-to-peer networks.

Subnets can easily become congested, increasing the delay and lowering the throughput for packets. Network designers attempt to avoid congestion by proper design. Techniques include retransmission policy, caching, flow control, and more. If congestion does occur, it must be dealt with. Choke packets can be sent back, load can be shed, and other methods applied.

The next step beyond just dealing with congestion is to actually try to achieve a promised quality of service. The methods that can be used for this include buffering at the client, traffic shaping, resource reservation, and admission control. Approaches that have been designed for good quality of service include integrated services (including RSVP), differentiated services, and MPLS.

Networks differ in various ways, so when multiple networks are interconnected problems can occur. Sometimes the problems can be finessed by tunneling a packet through a hostile network, but if the source and destination networks are different, this approach fails. When different networks have different maximum packet sizes, fragmentation may be called for.

The Internet has a rich variety of protocols related to the network layer. These include the data transport protocol, IP, but also the control protocols ICMP, ARP, and RARP, and the routing protocols OSPF and BGP. The Internet is rapidly running out of IP addresses, so a new version of IP, IPv6, has been developed.

Problems

1.

1. Give two example computer applications for which connection-oriented service is appropriate. Now give two examples for which connectionless service is best.
2.

2. Are there any circumstances when connection-oriented service will (or at least should) deliver packets out of order? Explain.
3.

3. Datagram subnets route each packet as a separate unit, independent of all others. Virtual-circuit subnets do not have to do this, since each data packet follows a predetermined route. Does this observation mean that virtual-circuit subnets do not need the capability to route isolated packets from an arbitrary source to an arbitrary destination? Explain your answer.

Chapter 6. The Transport Layer

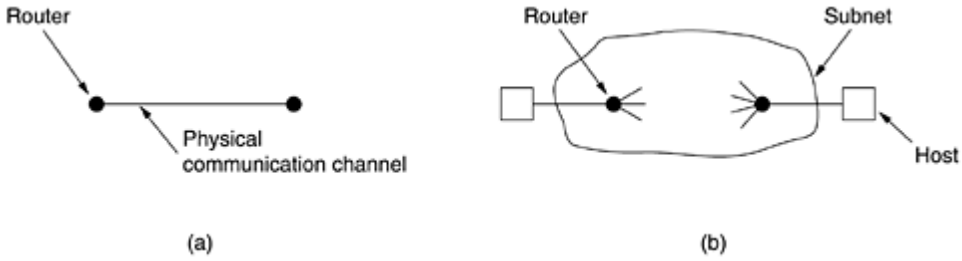
The transport layer is not just another layer. It is the heart of the whole protocol hierarchy. Its task is to provide reliable, cost-effective data transport from the source machine to the destination machine, independently of the physical network or networks currently in use. Without the transport layer, the whole concept of layered protocols would make little sense. In this chapter we will study the transport layer in detail, including its services, design, protocols, and performance.

6.2 Elements of Transport Protocols

The transport service is implemented by a transport protocol used between the two transport entities. In some ways, transport protocols resemble the data link protocols we studied in detail in Chap. 3. Both have to deal with error control, sequencing, and flow control, among other issues.

However, significant differences between the two also exist. These differences are due to major dissimilarities between the environments in which the two protocols operate, as shown in Fig. 6-7. At the data link layer, two routers communicate directly via a physical channel, whereas at the transport layer, this physical channel is replaced by the entire subnet. This difference has many important implications for the protocols, as we shall see in this chapter.

Figure 6-7. (a) Environment of the data link layer. (b) Environment of the transport layer.



For one thing, in the data link layer, it is not necessary for a router to specify which router it wants to talk to—each outgoing line uniquely specifies a particular router. In the transport layer, explicit addressing of destinations is required.

For another thing, the process of establishing a connection over the wire of Fig. 6-7(a) is simple: the other end is always there (unless it has crashed, in which case it is not there). Either way, there is not much to do. In the transport layer, initial connection establishment is more complicated, as we will see.

Another, exceedingly annoying, difference between the data link layer and the transport layer is the potential existence of storage capacity in the subnet. When a router sends a frame, it may arrive or be lost, but it cannot bounce around for a while, go into hiding in a far corner of the world, and then suddenly emerge at an inopportune moment 30 sec later. If the subnet uses datagrams and adaptive routing inside, there is a nonnegligible probability that a packet may be stored for a number of seconds and then delivered later. The consequences of the subnet's ability to store packets can sometimes be disastrous and can require the use of special protocols.

A final difference between the data link and transport layers is one of amount rather than of kind. Buffering and flow control are needed in both layers, but the presence of a large and dynamically varying number of connections in the transport layer may require a different approach than we used in the data link layer. In Chap. 3, some of the protocols allocate a fixed number of buffers to each line, so that when a frame arrives a buffer is always available. In the transport layer, the larger number of connections that must be managed make the idea of dedicating many buffers to each one less attractive. In the following sections, we will examine all of these important issues and others.

6.2.1 Addressing

When an application (e.g., a user) process wishes to set up a connection to a remote application process, it must specify which one to connect to. (Connectionless transport has the same problem: To whom should each message be

6.3 A Simple Transport Protocol

To make the ideas discussed so far more concrete, in this section we will study an example transport layer in detail. The abstract service primitives we will use are the connection-oriented primitives of Fig. 6-2. The choice of these connection-oriented primitives makes the example similar to (but simpler than) the popular TCP protocol.

6.3.1 The Example Service Primitives

Our first problem is how to express these transport primitives concretely. CONNECT is easy: we will just have a library procedure connect that can be called with the appropriate parameters necessary to establish a connection. The parameters are the local and remote TSAPs. During the call, the caller is blocked (i.e., suspended) while the transport entity tries to set up the connection. If the connection succeeds, the caller is unblocked and can start transmitting data.

When a process wants to be able to accept incoming calls, it calls listen, specifying a particular TSAP to listen to. The process then blocks until some remote process attempts to establish a connection to its TSAP.

Note that this model is highly asymmetric. One side is passive, executing a listen and waiting until something happens. The other side is active and initiates the connection. An interesting question arises of what to do if the active side begins first. One strategy is to have the connection attempt fail if there is no listener at the remote TSAP. Another strategy is to have the initiator block (possibly forever) until a listener appears.

A compromise, used in our example, is to hold the connection request at the receiving end for a certain time interval. If a process on that host calls listen before the timer goes off, the connection is established; otherwise, it is rejected and the caller is unblocked and given an error return.

To release a connection, we will use a procedure disconnect. When both sides have disconnected, the connection is released. In other words, we are using a symmetric disconnection model.

Data transmission has precisely the same problem as connection establishment: sending is active but receiving is passive. We will use the same solution for data transmission as for connection establishment: an active call send that transmits data and a passive call receive that blocks until a TPDU arrives. Our concrete service definition therefore consists of five primitives: CONNECT, LISTEN, DISCONNECT, SEND, and RECEIVE. Each primitive corresponds exactly to a library procedure that executes the primitive. The parameters for the service primitives and library procedures are as follows:

```
connum = LISTEN(local)
connum = CONNECT(local, remote)
status = SEND(connum, buffer, bytes)
status = RECEIVE(connum, buffer, bytes)
status = DISCONNECT(connum)
```

The LISTEN primitive announces the caller's willingness to accept connection requests directed at the indicated TSAP. The user of the primitive is blocked until an attempt is made to connect to it. There is no timeout.

6.4 The Internet Transport Protocols: UDP

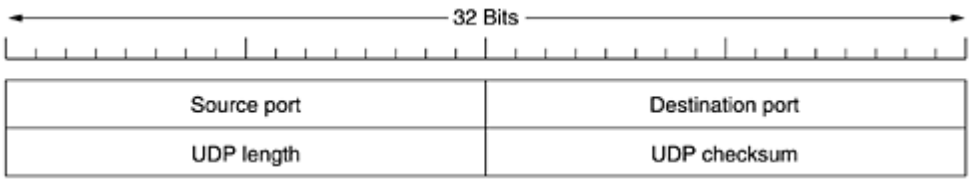
The Internet has two main protocols in the transport layer, a connectionless protocol and a connection-oriented one. In the following sections we will study both of them. The connectionless protocol is UDP. The connection-oriented protocol is TCP. Because UDP is basically just IP with a short header added, we will start with it. We will also look at two applications of UDP.

6.4.1 Introduction to UDP

The Internet protocol suite supports a connectionless transport protocol, UDP (User Datagram Protocol). UDP provides a way for applications to send encapsulated IP datagrams and send them without having to establish a connection. UDP is described in RFC 768.

UDP transmits segments consisting of an 8-byte header followed by the payload. The header is shown in Fig. 6-23. The two ports serve to identify the end points within the source and destination machines. When a UDP packet arrives, its payload is handed to the process attached to the destination port. This attachment occurs when BIND primitive or something similar is used, as we saw in Fig. 6-6 for TCP (the binding process is the same for UDP). In fact, the main value of having UDP over just using raw IP is the addition of the source and destination ports. Without the port fields, the transport layer would not know what to do with the packet. With them, it delivers segments correctly.

Figure 6-23. The UDP header.



The source port is primarily needed when a reply must be sent back to the source. By copying the source port field from the incoming segment into the destination port field of the outgoing segment, the process sending the reply can specify which process on the sending machine is to get it.

The UDP length field includes the 8-byte header and the data. The UDP checksum is optional and stored as 0 if not computed (a true computed 0 is stored as all 1s). Turning it off is foolish unless the quality of the data does not matter (e.g., digitized speech).

It is probably worth mentioning explicitly some of the things that UDP does not do. It does not do flow control, error control, or retransmission upon receipt of a bad segment. All of that is up to the user processes. What it does do is provide an interface to the IP protocol with the added feature of demultiplexing multiple processes using the ports. That is all it does. For applications that need to have precise control over the packet flow, error control, or timing, UDP provides just what the doctor ordered.

One area where UDP is especially useful is in client-server situations. Often, the client sends a short request to the server and expects a short reply back. If either the request or reply is lost, the client can just time out and try again. Not only is the code simple, but fewer messages are required (one in each direction) than with a protocol requiring an

6.5 The Internet Transport Protocols: TCP

UDP is a simple protocol and it has some niche uses, such as client-server interactions and multimedia, but for most Internet applications, reliable, sequenced delivery is needed. UDP cannot provide this, so another protocol is required. It is called TCP and is the main workhorse of the Internet. Let us now study it in detail.

6.5.1 Introduction to TCP

TCP (Transmission Control Protocol) was specifically designed to provide a reliable end-to-end byte stream over an unreliable internetwork. An internetwork differs from a single network because different parts may have wildly different topologies, bandwidths, delays, packet sizes, and other parameters. TCP was designed to dynamically adapt to properties of the internetwork and to be robust in the face of many kinds of failures.

TCP was formally defined in RFC 793. As time went on, various errors and inconsistencies were detected, and the requirements were changed in some areas. These clarifications and some bug fixes are detailed in RFC 1122. Extensions are given in RFC 1323.

Each machine supporting TCP has a TCP transport entity, either a library procedure, a user process, or part of the kernel. In all cases, it manages TCP streams and interfaces to the IP layer. A TCP entity accepts user data streams from local processes, breaks them up into pieces not exceeding 64 KB (in practice, often 1460 data bytes in order to fit in a single Ethernet frame with the IP and TCP headers), and sends each piece as a separate IP datagram. When datagrams containing TCP data arrive at a machine, they are given to the TCP entity, which reconstructs the original byte streams. For simplicity, we will sometimes use just "TCP" to mean the TCP transport entity (a piece of software) or the TCP protocol (a set of rules). From the context it will be clear which is meant. For example, in "The user gives TCP the data," the TCP transport entity is clearly intended.

The IP layer gives no guarantee that datagrams will be delivered properly, so it is up to TCP to time out and retransmit them as need be. Datagrams that do arrive may well do so in the wrong order; it is also up to TCP to reassemble them into messages in the proper sequence. In short, TCP must furnish the reliability that most users want and that IP does not provide.

6.5.2 The TCP Service Model

TCP service is obtained by both the sender and receiver creating end points, called sockets, as discussed in Sec. 6.1.3. Each socket has a socket number (address) consisting of the IP address of the host and a 16-bit number local to that host, called a port. A port is the TCP name for a TSAP. For TCP service to be obtained, a connection must be explicitly established between a socket on the sending machine and a socket on the receiving machine. The socket calls are listed in Fig. 6-5.

A socket may be used for multiple connections at the same time. In other words, two or more connections may terminate at the same socket. Connections are identified by the socket identifiers at both ends, that is, (socket1, socket2). No virtual circuit numbers or other identifiers are used.

Port numbers below 1024 are called well-known ports and are reserved for standard services. For example, any

6.6 Performance Issues

Performance issues are very important in computer networks. When hundreds or thousands of computers are interconnected, complex interactions, with unforeseen consequences, are common. Frequently, this complexity leads to poor performance and no one knows why. In the following sections, we will examine many issues related to network performance to see what kinds of problems exist and what can be done about them.

Unfortunately, understanding network performance is more an art than a science. There is little underlying theory that is actually of any use in practice. The best we can do is give rules of thumb gained from hard experience and present examples taken from the real world. We have intentionally delayed this discussion until we studied the transport layer in TCP in order to be able to use TCP as an example in various places.

The transport layer is not the only place performance issues arise. We saw some of them in the network layer in the previous chapter. Nevertheless, the network layer tends to be largely concerned with routing and congestion control. The broader, system-oriented issues tend to be transport related, so this chapter is an appropriate place to examine them.

In the next five sections, we will look at five aspects of network performance:

1.

1. Performance problems.
2.

2. Measuring network performance.
3.

3. System design for better performance.
4.

4. Fast TPDU processing.
5.

5. Protocols for future high-performance networks.

As an aside, we need a generic name for the units exchanged by transport entities. The TCP term, segment, is confusing at best and is never used outside the TCP world in this context. The ATM terms (CS-PDU, SAR-PDU, and CPCS-PDU) are specific to ATM. Packets clearly refer to the network layer, and messages belong to the application layer. For lack of a standard term, we will go back to calling the units exchanged by transport entities TPDUs. When we mean both TPDU and packet together, we will use packet as the collective term, as in "The CPU must be fast enough to process incoming packets in real time." By this we mean both the network layer packet and the TPDU encapsulated in it.

6.6.1 Performance Problems in Computer Networks

Some performance problems, such as congestion, are caused by temporary resource overloads. If more traffic

6.7 Summary

The transport layer is the key to understanding layered protocols. It provides various services, the most important of which is an end-to-end, reliable, connection-oriented byte stream from sender to receiver. It is accessed through service primitives that permit the establishment, use, and release of connections. A common transport layer interface is the one provided by Berkeley sockets.

Transport protocols must be able to do connection management over unreliable networks. Connection establishment is complicated by the existence of delayed duplicate packets that can reappear at inopportune moments. To deal with them, three-way handshakes are needed to establish connections. Releasing a connection is easier than establishing one but is still far from trivial due to the two-army problem.

Even when the network layer is completely reliable, the transport layer has plenty of work to do. It must handle all the service primitives, manage connections and timers, and allocate and utilize credits.

The Internet has two main transport protocols: UDP and TCP. UDP is a connectionless protocol that is mainly a wrapper for IP packets with the additional feature of multiplexing and demultiplexing multiple processes using a single IP address. UDP can be used for client-server interactions, for example, using RPC. It can also be used for building real-time protocols such as RTP.

The main Internet transport protocol is TCP. It provides a reliable bidirectional byte stream. It uses a 20-byte header on all segments. Segments can be fragmented by routers within the Internet, so hosts must be prepared to do reassembly. A great deal of work has gone into optimizing TCP performance, using algorithms from Nagle, Clark, Jacobson, Karn, and others. Wireless links add a variety of complications to TCP. Transactional TCP is an extension to TCP that handles client-server interactions with a reduced number of packets.

Network performance is typically dominated by protocol and TPDU processing overhead, and this situation gets worse at higher speeds. Protocols should be designed to minimize the number of TPDUs, context switches, and times each TPDU is copied. For gigabit networks, simple protocols are called for.

Problems

1.

1. In our example transport primitives of Fig. 6-2, LISTEN is a blocking call. Is this strictly necessary? If not, explain how a nonblocking primitive could be used. What advantage would this have over the scheme described in the text?
2.

2. In the model underlying Fig. 6-4, it is assumed that packets may be lost by the network layer and thus must be individually acknowledged. Suppose that the network layer is 100 percent reliable and never loses packets. What changes, if any, are needed to Fig. 6-4?
3.

3. In both parts of Fig. 6-6, there is a comment that the value of SERVER_PORT must be the same in both client and server. Why is this so important?

Chapter 7. The Application Layer

Having finished all the preliminaries, we now come to the layer where all the applications are found. The layers below the application layer are there to provide reliable transport, but they do not do real work for users. In this chapter we will study some real network applications.

However, even in the application layer there is a need for support protocols, to allow the applications to function. Accordingly, we will look at one of these before starting with the applications themselves. The item in question is DNS, which handles naming within the Internet. After that, we will examine three real applications: electronic mail, the World Wide Web, and finally, multimedia.

7.1 DNS—The Domain Name System

Although programs theoretically could refer to hosts, mailboxes, and other resources by their network (e.g., IP) addresses, these addresses are hard for people to remember. Also, sending e-mail to tana@128.111.24.41 means that if Tana's ISP or organization moves the mail server to a different machine with a different IP address, her e-mail address has to change. Consequently, ASCII names were introduced to decouple machine names from machine addresses. In this way, Tana's address might be something like tana@art.ucsb.edu. Nevertheless, the network itself understands only numerical addresses, so some mechanism is required to convert the ASCII strings to network addresses. In the following sections we will study how this mapping is accomplished in the Internet.

Way back in the ARPANET, there was simply a file, hosts.txt, that listed all the hosts and their IP addresses. Every night, all the hosts would fetch it from the site at which it was maintained. For a network of a few hundred large timesharing machines, this approach worked reasonably well.

However, when thousands of minicomputers and PCs were connected to the net, everyone realized that this approach could not continue to work forever. For one thing, the size of the file would become too large. However, even more important, host name conflicts would occur constantly unless names were centrally managed, something unthinkable in a huge international network due to the load and latency. To solve these problems, DNS (the Domain Name System) was invented.

The essence of DNS is the invention of a hierarchical, domain-based naming scheme and a distributed database system for implementing this naming scheme. It is primarily used for mapping host names and e-mail destinations to IP addresses but can also be used for other purposes. DNS is defined in RFCs 1034 and 1035.

Very briefly, the way DNS is used is as follows. To map a name onto an IP address, an application program calls a library procedure called the resolver, passing it the name as a parameter. We saw an example of a resolver, gethostbyname, in Fig. 6-6. The resolver sends a UDP packet to a local DNS server, which then looks up the name and returns the IP address to the resolver, which then returns it to the caller. Armed with the IP address, the program can then establish a TCP connection with the destination or send it UDP packets.

7.1.1 The DNS Name Space

Managing a large and constantly changing set of names is a nontrivial problem. In the postal system, name management is done by requiring letters to specify (implicitly or explicitly) the country, state or province, city, and street address of the addressee. By using this kind of hierarchical addressing, there is no confusion between the Marvin Anderson on Main St. in White Plains, N.Y. and the Marvin Anderson on Main St. in Austin, Texas. DNS works the same way.

Conceptually, the Internet is divided into over 200 top-level domains, where each domain covers many hosts. Each domain is partitioned into subdomains, and these are further partitioned, and so on. All these domains can be represented by a tree, as shown in Fig. 7-1. The leaves of the tree represent domains that have no subdomains (but do contain machines, of course). A leaf domain may contain a single host, or it may represent a company and contain thousands of hosts.

7.2 Electronic Mail

Electronic mail, or e-mail, as it is known to its many fans, has been around for over two decades. Before 1990, it was mostly used in academia. During the 1990s, it became known to the public at large and grew exponentially to the point where the number of e-mails sent per day now is vastly more than the number of snail mail (i.e., paper) letters.

E-mail, like most other forms of communication, has its own conventions and styles. In particular, it is very informal and has a low threshold of use. People who would never dream of calling up or even writing a letter to a Very Important Person do not hesitate for a second to send a sloppily-written e-mail.

E-mail is full of jargon such as BTW (By The Way), ROTFL (Rolling On The Floor Laughing), and IMHO (In My Humble Opinion). Many people also use little ASCII symbols called smileys or emoticons in their e-mail. A few of the more interesting ones are reproduced in Fig. 7-6. For most, rotating the book 90 degrees clockwise will make them clearer. For a minibook giving over 650 smileys, see (Sanderson and Dougherty, 1993).

Figure 7-6. Some smileys. They will not be on the final exam :-)

Smiley	Meaning	Smiley	Meaning	Smiley	Meaning
:-)	I'm happy	=|:-)	Abe Lincoln	:+)	Big nose
:-(	I'm sad/angry	=):-)	Uncle Sam	:-))	Double chin
:-|	I'm apathetic	*<:-)	Santa Claus	:-{)	Mustache
;-)	I'm winking	<:-(	Dunce	#:-)	Matted hair
:-(O)	I'm yelling	(-:	Australian	8-)	Wears glasses
:-(*)	I'm vomiting	:-)X	Man with bowtie	C:-)	Large brain

The first e-mail systems simply consisted of file transfer protocols, with the convention that the first line of each message (i.e., file) contained the recipient's address. As time went on, the limitations of this approach became more obvious.

Some of the complaints were as follows:

1.

1. Sending a message to a group of people was inconvenient. Managers often need this facility to send memos to all their subordinates.
2.

2. Messages had no internal structure, making computer processing difficult. For example, if a forwarded message was included in the body of another message, extracting the forwarded part from the received message was difficult.
3.

3. The originator (sender) never knew if a message arrived or not.
4.

4. If someone was planning to be away on business for several weeks and wanted all incoming e-mail to be handled by his secretary, this was not easy to arrange.

7.3 The World Wide Web

The World Wide Web is an architectural framework for accessing linked documents spread out over millions of machines all over the Internet. In 10 years, it went from being a way to distribute high-energy physics data to the application that millions of people think of as being "The Internet." Its enormous popularity stems from the fact that it has a colorful graphical interface that is easy for beginners to use, and it provides an enormous wealth of information on almost every conceivable subject, from aardvarks to Zulus.

The Web (also known as WWW) began in 1989 at CERN, the European center for nuclear research. CERN has several accelerators at which large teams of scientists from the participating European countries carry out research in particle physics. These teams often have members from half a dozen or more countries. Most experiments are highly complex and require years of advance planning and equipment construction. The Web grew out of the need to have these large teams of internationally dispersed researchers collaborate using a constantly changing collection of reports, blueprints, drawings, photos, and other documents.

The initial proposal for a web of linked documents came from CERN physicist Tim Berners-Lee in March 1989. The first (text-based) prototype was operational 18 months later. In December 1991, a public demonstration was given at the Hypertext '91 conference in San Antonio, Texas.

This demonstration and its attendant publicity caught the attention of other researchers, which led Marc Andreessen at the University of Illinois to start developing the first graphical browser, Mosaic. It was released in February 1993. Mosaic was so popular that a year later, Andreessen left to form a company, Netscape Communications Corp., whose goal was to develop clients, servers, and other Web software. When Netscape went public in 1995, investors, apparently thinking this was the next Microsoft, paid $1.5 billion for the stock. This record was all the more surprising because the company had only one product, was operating deeply in the red, and had announced in its prospectus that it did not expect to make a profit for the foreseeable future. For the next three years, Netscape Navigator and Microsoft's Internet Explorer engaged in a "browser war," each one trying frantically to add more features (and thus more bugs) than the other one. In 1998, America Online bought Netscape Communications Corp. for $4.2 billion, thus ending Netscape's brief life as an independent company.

In 1994, CERN and M.I.T. signed an agreement setting up the World Wide Web Consortium (sometimes abbreviated as W3C), an organization devoted to further developing the Web, standardizing protocols, and encouraging interoperability between sites. Berners-Lee became the director. Since then, several hundred universities and companies have joined the consortium. Although there are now more books about the Web than you can shake a stick at, the best place to get up-to-date information about the Web is (naturally) on the Web itself. The consortium's home page is at www.w3.org. Interested readers are referred there for links to pages covering all of the consortium's numerous documents and activities.

7.3.1 Architectural Overview

From the users' point of view, the Web consists of a vast, worldwide collection of documents or Web pages, often just called pages for short. Each page may contain links to other pages anywhere in the world. Users can follow a link by clicking on it, which then takes them to the page pointed to. This process can be repeated indefinitely. The idea of having one page point to another, now called hypertext, was invented by a visionary M.I.T. professor of electrical engineering, Vannevar Bush, in 1945, long before the Internet was invented.

7.4 Multimedia

The wireless Web is an exciting new development, but it is not the only one. For many people, multimedia is the holy grail of networking. When the word is mentioned, both the propeller heads and the suits begin salivating as if on cue. The former see immense technical challenges in providing (interactive) video on demand to every home. The latter see equally immense profits in it. Since multimedia requires high bandwidth, getting it to work over fixed connections is hard enough. Even VHS-quality video over wireless is a few years away, so our treatment will focus on wired systems.

Literally, multimedia is just two or more media. If the publisher of this book wanted to join the current hype about multimedia, it could advertise the book as using multimedia technology. After all, it contains two media: text and graphics (the figures). Nevertheless, when most people refer to multimedia, they generally mean the combination of two or more continuous media, that is, media that have to be played during some well-defined time interval, usually with some user interaction. In practice, the two media are normally audio and video, that is, sound plus moving pictures.

However, many people often refer to pure audio, such as Internet telephony or Internet radio as multimedia as well, which it is clearly not. Actually, a better term is streaming media, but we will follow the herd and consider real-time audio to be multimedia as well. In the following sections we will examine how computers process audio and video, how they are compressed, and some network applications of these technologies. For a comprehensive (three volume) treatment on networked multimedia, see (Steinmetz and Nahrstedt, 2002; Steinmetz and Nahrstedt, 2003a; and Steinmetz and Nahrstedt, 2003b).

7.4.1 Introduction to Digital Audio

An audio (sound) wave is a one-dimensional acoustic (pressure) wave. When an acoustic wave enters the ear, the eardrum vibrates, causing the tiny bones of the inner ear to vibrate along with it, sending nerve pulses to the brain. These pulses are perceived as sound by the listener. In a similar way, when an acoustic wave strikes a microphone, the microphone generates an electrical signal, representing the sound amplitude as a function of time. The representation, processing, storage, and transmission of such audio signals are a major part of the study of multimedia systems.

The frequency range of the human ear runs from 20 Hz to 20,000 Hz. Some animals, notably dogs, can hear higher frequencies. The ear hears logarithmically, so the ratio of two sounds with power A and B is conventionally expressed in dB (decibels) according to the formula

$$\text{dB} = 10 \log_{10}(A/B)$$

If we define the lower limit of audibility (a pressure of about 0.0003 dyne/cm2) for a 1-kHz sine wave as 0 dB, an ordinary conversation is about 50 dB and the pain threshold is about 120 dB, a dynamic range of a factor of 1 million.

The ear is surprisingly sensitive to sound variations lasting only a few milliseconds. The eye, in contrast, does not notice changes in light level that last only a few milliseconds. The result of this observation is that jitter of only a few milliseconds during a multimedia transmission affects the perceived sound quality more than it affects the perceived image quality.

7.5 Summary

Naming in the Internet uses a hierarchical scheme called the domain name system (DNS). At the top level are the well-known generic domains, including com and edu as well as about 200 country domains. DNS is implemented as a distributed database system with servers all over the world. DNS holds records with IP addresses, mail exchanges, and other information. By querying a DNS server, a process can map an Internet domain name onto the IP address used to communicate with that domain.

E-mail is one of the two killer apps for the Internet. Everyone from small children to grandparents now use it. Most e-mail systems in the world use the mail system now defined in RFCs 2821 and 2822. Messages sent in this system use system ASCII headers to define message properties. Many kinds of content can be sent using MIME. Messages are sent using SMTP, which works by making a TCP connection from the source host to the destination host and directly delivering the e-mail over the TCP connection.

The other killer app for the Internet is the World Wide Web. The Web is a system for linking hypertext documents. Originally, each document was a page written in HTML with hyperlinks to other documents. Nowadays, XML is gradually starting to take over from HTML. Also, a large amount of content is dynamically generated, using server-side scripts (PHP, JSP, and ASP), as well as clientside scripts (notably JavaScript). A browser can display a document by establishing a TCP connection to its server, asking for the document, and then closing the connection. These request messages contain a variety of headers for providing additional information. Caching, replication, and content delivery networks are widely used to enhance Web performance.

The wireless Web is just getting started. The first systems are WAP and i-mode, each with small screens and limited bandwidth, but the next generation will be more powerful.

Multimedia is also a rising star in the networking firmament. It allows audio and video to be digitized and transported electronically for display. Audio requires less bandwidth, so it is further along. Streaming audio, Internet radio, and voice over IP are a reality now, with new applications coming along all the time. Video on demand is an up-and-coming area in which there is great interest. Finally, the MBone is an experimental, worldwide digital live television service sent over the Internet.

Problems

1.

1. Many business computers have three distinct and worldwide unique identifiers. What are they?
2.

2. According to the information given in Fig. 7-3, is little-sister.cs.vu.nl on a class A, B, or C network?
3.

3. In Fig. 7-3, there is no period after rowboat? Why not?
4.

4. Make a guess about what the smiley :-X (sometimes written as :-#) might mean.
5.

Chapter 8. Network Security

For the first few decades of their existence, computer networks were primarily used by university researchers for sending e-mail and by corporate employees for sharing printers. Under these conditions, security did not get a lot of attention. But now, as millions of ordinary citizens are using networks for banking, shopping, and filing their tax returns, network security is looming on the horizon as a potentially massive problem. In this chapter, we will study network security from several angles, point out numerous pitfalls, and discuss many algorithms and protocols for making networks more secure.

Security is a broad topic and covers a multitude of sins. In its simplest form, it is concerned with making sure that nosy people cannot read, or worse yet, secretly modify messages intended for other recipients. It is concerned with people trying to access remote services that they are not authorized to use. It also deals with ways to tell whether that message purportedly from the IRS saying: Pay by Friday or else is really from the IRS and not from the Mafia. Security also deals with the problems of legitimate messages being captured and replayed, and with people trying to deny that they sent certain messages.

Most security problems are intentionally caused by malicious people trying to gain some benefit, get attention, or to harm someone. A few of the most common perpetrators are listed in Fig. 8-1. It should be clear from this list that making a network secure involves a lot more than just keeping it free of programming errors. It involves outsmarting often intelligent, dedicated, and sometimes well-funded adversaries. It should also be clear that measures that will thwart casual adversaries will have little impact on the serious ones. Police records show that most attacks are not perpetrated by outsiders tapping a phone line but by insiders with a grudge. Consequently, security systems should be designed with this fact in mind.

Figure 8-1. Some people who cause security problems and why.

Adversary	Goal
Student	To have fun snooping on people's e-mail
Cracker	To test out someone's security system; steal data
Sales rep	To claim to represent all of Europe, not just Andorra
Businessman	To discover a competitor's strategic marketing plan
Ex-employee	To get revenge for being fired
Accountant	To embezzle money from a company
Stockbroker	To deny a promise made to a customer by e-mail
Con man	To steal credit card numbers for sale
Spy	To learn an enemy's military or industrial secrets
Terrorist	To steal germ warfare secrets

Network security problems can be divided roughly into four closely intertwined areas: secrecy, authentication, nonrepudiation, and integrity control. Secrecy, also called confidentiality, has to do with keeping information out of the hands of unauthorized users. This is what usually comes to mind when people think about network security. Authentication deals with determining whom you are talking to before revealing sensitive information or entering into a business deal. Nonrepudiation deals with signatures: How do you prove that your customer really placed an electronic order for ten million left-handed doohickeys at 89 cents each when he later claims the price was 69 cents? Or maybe he claims he never placed any order. Finally, how can you be sure that a message you received was really the one sent and not something that a malicious adversary modified in transit or concocted?

8.1 Cryptography

Cryptography comes from the Greek words for "secret writing." It has a long and colorful history going back thousands of years. In this section we will just sketch some of the highlights, as background information for what follows. For a complete history of cryptography, Kahn's (1995) book is recommended reading. For a comprehensive treatment of the current state-of-the-art in security and cryptographic algorithms, protocols, and applications, see (Kaufman et al., 2002). For a more mathematical approach, see (Stinson, 2002). For a less mathematical approach, see (Burnett and Paine, 2001).

Professionals make a distinction between ciphers and codes. A cipher is a character-for-character or bit-for-bit transformation, without regard to the linguistic structure of the message. In contrast, a code replaces one word with another word or symbol. Codes are not used any more, although they have a glorious history. The most successful code ever devised was used by the U.S. armed forces during World War II in the Pacific. They simply had Navajo Indians talking to each other using specific Navajo words for military terms, for example chay-dagahi-nail-tsaidi (literally: tortoise killer) for antitank weapon. The Navajo language is highly tonal, exceedingly complex, and has no written form. And not a single person in Japan knew anything about it.

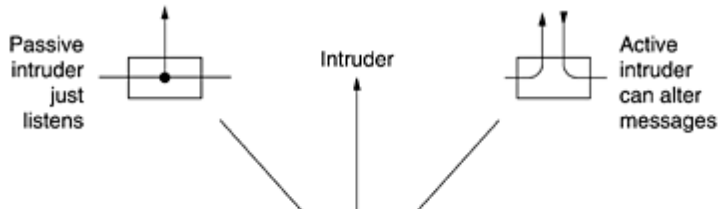
In September 1945, the San Diego Union described the code by saying "For three years, wherever the Marines landed, the Japanese got an earful of strange gurgling noises interspersed with other sounds resembling the call of a Tibetan monk and the sound of a hot water bottle being emptied." The Japanese never broke the code and many Navajo code talkers were awarded high military honors for extraordinary service and bravery. The fact that the U.S. broke the Japanese code but the Japanese never broke the Navajo code played a crucial role in the American victories in the Pacific.

8.1.1 Introduction to Cryptography

Historically, four groups of people have used and contributed to the art of cryptography: the military, the diplomatic corps, diarists, and lovers. Of these, the military has had the most important role and has shaped the field over the centuries. Within military organizations, the messages to be encrypted have traditionally been given to poorly-paid, low-level code clerks for encryption and transmission. The sheer volume of messages prevented this work from being done by a few elite specialists.

Until the advent of computers, one of the main constraints on cryptography had been the ability of the code clerk to perform the necessary transformations, often on a battlefield with little equipment. An additional constraint has been the difficulty in switching over quickly from one cryptographic method to another one, since this entails retraining a large number of people. However, the danger of a code clerk being captured by the enemy has made it essential to be able to change the cryptographic method instantly if need be. These conflicting requirements have given rise to the model of Fig. 8-2.

Figure 8-2. The encryption model (for a symmetric-key cipher).



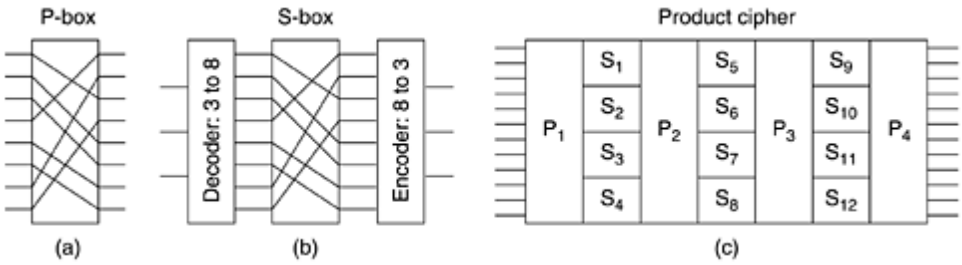
8.2 Symmetric-Key Algorithms

Modern cryptography uses the same basic ideas as traditional cryptography (transposition and substitution) but its emphasis is different. Traditionally, cryptographers have used simple algorithms. Nowadays the reverse is true: the object is to make the encryption algorithm so complex and involuted that even if the cryptanalyst acquires vast mounds of enciphered text of his own choosing, he will not be able to make any sense of it at all without the key.

The first class of encryption algorithms we will study in this chapter are called symmetric-key algorithms because they used the same key for encryption and decryption. Fig. 8-2 illustrates the use of a symmetric-key algorithm. In particular, we will focus on block ciphers, which take an n-bit block of plaintext as input and transform it using the key into n-bit block of ciphertext.

Cryptographic algorithms can be implemented in either hardware (for speed) or in software (for flexibility). Although most of our treatment concerns the algorithms and protocols, which are independent of the actual implementation, a few words about building cryptographic hardware may be of interest. Transpositions and substitutions can be implemented with simple electrical circuits. Figure 8-6(a) shows a device, known as a P-box (P stands for permutation), used to effect a transposition on an 8-bit input. If the 8 bits are designated from top to bottom as 01234567, the output of this particular P-box is 36071245. By appropriate internal wiring, a P-box can be made to perform any transposition and do it at practically the speed of light since no computation is involved, just signal propagation. This design follows Kerckhoff's principle: the attacker knows that the general method is permuting the bits. What he does not know is which bit goes where, which is the key.

Figure 8-6. Basic elements of product ciphers. (a) P-box. (b) S-box. (c) Product.



Substitutions are performed by S-boxes, as shown in Fig. 8-6(b). In this example a 3-bit plaintext is entered and a 3-bit ciphertext is output. The 3-bit input selects one of the eight lines exiting from the first stage and sets it to 1; all the other lines are 0. The second stage is a P-box. The third stage encodes the selected input line in binary again. With the wiring shown, if the eight octal numbers 01234567 were input one after another, the output sequence would be 24506713. In other words, 0 has been replaced by 2, 1 has been replaced by 4, etc. Again, by appropriate wiring of the P-box inside the S-box, any substitution can be accomplished. Furthermore, such a device can be built in hardware and can achieve great speed since encoders and decoders have only one or two (subnanosecond) gate delays and the propagation time across the P-box may well be less than 1 picosecond.

The real power of these basic elements only becomes apparent when we cascade a whole series of boxes to form a product cipher, as shown in Fig. 8-6(c). In this example, 12 input lines are transposed (i.e., permuted) by the first stage (P1). Theoretically, it would be possible to have the second stage be an S-box that mapped a 12-bit number onto another 12-bit number. However, such a device would need 212 = 4096 crossed wires in its middle stage. Instead, the input is broken up into four groups of 3 bits, each of which is substituted independently of the others. Although this method is less general, it is still powerful. By inclusion of a sufficiently large number of stages in the product cipher, the output can be made to be an exceedingly complicated function of the input.

Actually, NSA stands for National Security Agency.

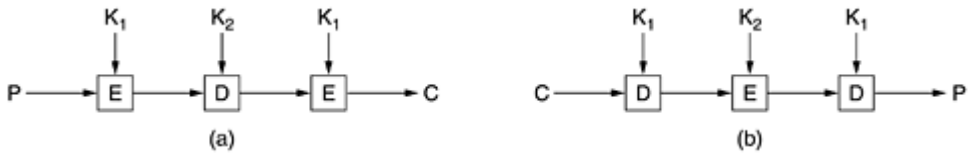
After these discussions took place, IBM reduced the key from 128 bits to 56 bits and decided to keep secret the process by which DES was designed. Many people suspected that the key length was reduced to make sure that NSA could just break DES, but no organization with a smaller budget could. The point of the secret design was supposedly to hide a back door that could make it even easier for NSA to break DES. When an NSA employee discreetly told IEEE to cancel a planned conference on cryptography, that did not make people any more comfortable. NSA denied everything.

In 1977, two Stanford cryptography researchers, Diffie and Hellman (1977), designed a machine to break DES and estimated that it could be built for 20 million dollars. Given a small piece of plaintext and matched ciphertext, this machine could find the key by exhaustive search of the 256-entry key space in under 1 day. Nowadays, such a machine would cost well under 1 million dollars.

Triple DES

As early as 1979, IBM realized that the DES key length was too short and devised a way to effectively increase it, using triple encryption (Tuchman, 1979). The method chosen, which has since been incorporated in International Standard 8732, is illustrated in Fig. 8-8. Here two keys and three stages are used. In the first stage, the plaintext is encrypted using DES in the usual way with K1. In the second stage, DES is run in decryption mode, using K2 as the key. Finally, another DES encryption is done with K1.

Figure 8-8. (a) Triple encryption using DES. (b) Decryption.



This design immediately gives rise to two questions. First, why are only two keys used, instead of three? Second, why is EDE (Encrypt Decrypt Encrypt) used, instead of EEE (Encrypt Encrypt Encrypt)? The reason that two keys are used is that even the most paranoid cryptographers believe that 112 bits is adequate for routine commercial applications for the time being. (And among cryptographers, paranoia is considered a feature, not a bug.) Going to 168 bits would just add the unnecessary overhead of managing and transporting another key for little real gain.

The reason for encrypting, decrypting, and then encrypting again is backward compatibility with existing single-key DES systems. Both the encryption and decryption functions are mappings between sets of 64-bit numbers. From a cryptographic point of view, the two mappings are equally strong. By using EDE, however, instead of EEE, a computer using triple encryption can speak to one using single encryption by just setting K1 = K2. This property allows triple encryption to be phased in gradually, something of no concern to academic cryptographers, but of considerable importance to IBM and its customers.

8.2.2 AES—The Advanced Encryption Standard

As DES began approaching the end of its useful life, even with triple DES, NIST (National Institute of Standards and

Technology), the agency of the U.S. Dept. of Commerce charged with approving standards for the U.S. Federal Government, decided that the government needed a new cryptographic standard for unclassified use. NIST was keenly aware of all the controversy surrounding DES and well knew that if it just announced a new standard, everyone knowing anything about cryptography would automatically assume that NSA had built a back door into it so NSA could read everything encrypted with it. Under these conditions, probably no one would use the standard and it would most likely die a quiet death.

So NIST took a surprisingly different approach for a government bureaucracy: it sponsored a cryptographic bake-off (contest). In January 1997, researchers from all over the world were invited to submit proposals for a new standard, to be called AES (Advanced Encryption Standard). The bake-off rules were:

1.

1. The algorithm must be a symmetric block cipher.
2.

2. The full design must be public.
3.

3. Key lengths of 128, 192, and 256 bits must be supported.
4.

4. Both software and hardware implementations must be possible.
5.

5. The algorithm must be public or licensed on nondiscriminatory terms.

Fifteen serious proposals were made, and public conferences were organized in which they were presented and attendees were actively encouraged to find flaws in all of them. In August 1998, NIST selected five finalists primarily on the basis of their security, efficiency, simplicity, flexibility, and memory requirements (important for embedded systems). More conferences were held and more pot-shots taken. A nonbinding vote was taken at the last conference. The finalists and their scores were as follows:

1.

1. Rijndael (from Joan Daemen and Vincent Rijmen, 86 votes).
2.

2. Serpent (from Ross Anderson, Eli Biham, and Lars Knudsen, 59 votes).
3.

3. Twofish (from a team headed by Bruce Schneier, 31 votes).
4.

4. RC6 (from RSA Laboratories, 23 votes).
5.

5. MARS (from IBM, 13 votes).

In October 2000, NIST announced that it, too, voted for Rijndael, and in November 2001 Rijndael became a U.S. Government standard published as Federal Information Processing Standard FIPS 197. Due to the extraordinary openness of the competition, the technical properties of Rijndael, and the fact that the winning team consisted of two young Belgian cryptographers (who are unlikely to have built in a back door just to please NSA), it is expected that Rijndael will become the world's dominant cryptographic standard for at least a decade. The name Rijndael, pronounced Rhine-doll (more or less), is derived from the last names of the authors: Rijmen + Daemen.

Rijndael supports key lengths and block sizes from 128 bits to 256 bits in steps of 32 bits. The key length and block length may be chosen independently. However, AES specifies that the block size must be 128 bits and the key length must be 128, 192, or 256 bits. It is doubtful that anyone will ever use 192-bit keys, so de facto, AES has two variants: a 128-bit block with 128-bit key and a 128-bit block with a 256-bit key.

In our treatment of the algorithm below, we will examine only the 128/128 case because this is likely to become the commercial norm. A 128-bit key gives a key space of 2128 ≈3 x 1038 keys. Even if NSA manages to build a machine with 1 billion parallel processors, each being able to evaluate one key per picosecond, it would take such a machine about 1010 years to search the key space. By then the sun will have burned out, so the folks then present will have to read the results by candlelight.

Rijndael

From a mathematical perspective, Rijndael is based on Galois field theory, which gives it some provable security properties. However, it can also be viewed as C code, without getting into the mathematics.

Like DES, Rijndael uses substitution and permutations, and it also uses multiple rounds. The number of rounds depends on the key size and block size, being 10 for 128-bit keys with 128-bit blocks and moving up to 14 for the largest key or the largest block. However, unlike DES, all operations involve entire bytes, to allow for efficient implementations in both hardware and software. An outline of the code is given in Fig. 8-9.

Figure 8-9. An outline of Rijndael.

```
#define LENGTH 16                       /* # bytes in data block or key */
#define NROWS 4                         /* number of rows in state */
#define NCOLS 4                         /* number of columns in state */
#define ROUNDS 10                       /* number of iterations */
typedef unsigned char byte;             /* unsigned 8-bit integer */

rijndael(byte plaintext[LENGTH], byte ciphertext[LENGTH], byte key[LENGTH])
{
  int r;                                /* loop index */
  byte state[NROWS][NCOLS];             /* current state */
  struct {byte k[NROWS][NCOLS];} rk[ROUNDS + 1];     /* round keys */

  expand_key(key, rk);                  /* construct the round keys */
  copy_plaintext_to_state(state, plaintext);    /* init current state */
  xor_roundkey_into_state(state, rk[0]);        /* XOR key into state */

  for (r = 1; r <= ROUNDS; r++) {
      substitute(state);                /* apply S-box to each byte */
      rotate_rows(state);               /* rotate row i by i bytes */
      if (r < ROUNDS) mix_columns(state);       /* mix function */
      xor_roundkey_into_state(state, rk[r]);    /* XOR key into state */
  }
  copy_state_to_ciphertext(ciphertext, state);  /* return result */
}
```

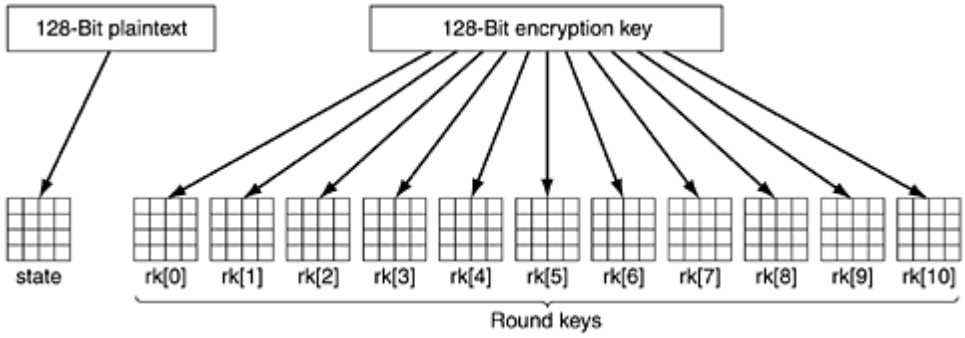
The function rijndael has three parameters. They are: plaintext, an array of 16 bytes containing the input data, ciphertext, an array of 16 bytes where the enciphered output will be returned, and key, the 16-byte key. During the calculation, the current state of the data is maintained in a byte array, state, whose size is NROWS x NCOLS. For 128-bit blocks, this array is 4 x 4 bytes. With 16 bytes, the full 128-bit data block can be stored.

The state array is initialized to the plaintext and modified by every step in the computation. In some steps, byte-for-byte substitution is performed. In others, the bytes are permuted within the array. Other transformations are also used. At the end, the contents of the state are returned as the ciphertext.

The code starts out by expanding the key into 11 arrays of the same size as the state. They are stored in rk, which is an array of structs, each containing a state array. One of these will be used at the start of the calculation and the other 10 will be used during the 10 rounds, one per round. The calculation of the round keys from the encryption key is too complicated for us to get into here. Suffice it to say that the round keys are produced by repeated rotation and XORing of various groups of key bits. For all the details, see (Daemen and Rijmen, 2002).

The next step is to copy the plaintext into the state array so it can be processed during the rounds. It is copied in column order, with the first four bytes going into column 0, the next four bytes going into column 1, and so on. Both the columns and the rows are numbered starting at 0, although the rounds are numbered starting at 1. This initial setup of the 12 byte arrays of size 4 x 4 is illustrated in Fig. 8-10.

Figure 8-10. Creating of the state and rk arrays.



There is one more step before the main computation begins: rk[0] is XORed into state byte for byte. In other words each of the 16 bytes in state is replaced by the XOR of itself and the corresponding byte in rk[0].

Now it is time for the main attraction. The loop executes 10 iterations, one per round, transforming state on each iteration. The contents of each round consist of four steps. Step 1 does a byte-for-byte substitution on state. Each byte in turn is used as an index into an S-box to replace its value by the contents of that S-box entry. This step is a straight monoalphabetic substitution cipher. Unlike DES, which has multiple S-boxes, Rijndael has only one S-box.

Step 2 rotates each of the four rows to the left. Row 0 is rotated 0 bytes (i.e., not changed), row 1 is rotated 1 byte, row 2 is rotated 2 bytes, and row 3 is rotated 3 bytes. This step diffuses the contents of the current data around the block, analogous to the permutations of Fig. 8-6.

Step 3 mixes up each column independently of the other ones. The mixing is done using matrix multiplication in which the new column is the product of the old column and a constant matrix, with the multiplication done using the finite Galois field, GF(28). Although this may sound complicated, an algorithm exists that allows each element of the new column to be computed using two table lookups and three XORs (Daemen and Rijmen, 2002, Appendix E).

Finally, step 4 XORs the key for this round into the state array.

Since every step is reversible, decryption can be done just by running the algorithm backward. However, there is also a trick available in which decryption can be done by running the encryption algorithm, using different tables.

The algorithm has been designed not only for great security, but also for great speed. A good software implementation on a 2-GHz machine should be able to achieve an encryption rate of 700 Mbps, which is fast enough to encrypt over 100 MPEG-2 videos in real time. Hardware implementations are faster still.

8.2.3 Cipher Modes

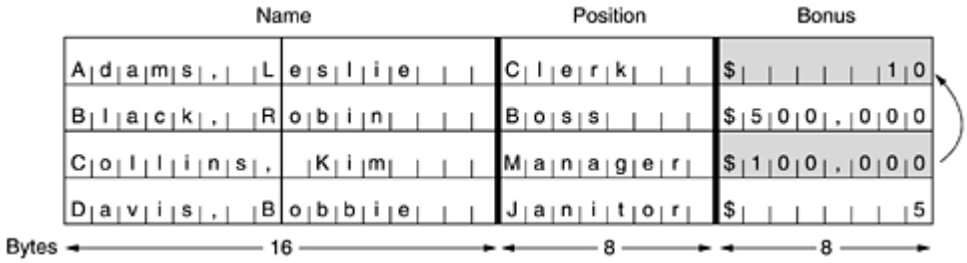
Despite all this complexity, AES (or DES or any block cipher for that matter) is basically a monoalphabetic substitution cipher using big characters (128-bit characters for AES and 64-bit characters for DES). Whenever the same plaintext block goes in the front end, the same ciphertext block comes out the back end. If you encrypt the plaintext abcdefgh 100 times with the same DES key, you get the same ciphertext 100 times. An intruder can exploit this property to help subvert the cipher.

Electronic Code Book Mode

To see how this monoalphabetic substitution cipher property can be used to partially defeat the cipher, we will use (triple) DES because it is easier to depict 64-bit blocks than 128-bit blocks, but AES has exactly the same problem. The straightforward way to use DES to encrypt a long piece of plaintext is to break it up into consecutive 8-byte (64-bit) blocks and encrypt them one after another with the same key. The last piece of plaintext is padded out to 64 bits, if need be. This technique is known as ECB mode (Electronic Code Book mode) in analogy with old-fashioned code books where each plaintext word was listed, followed by its ciphertext (usually a five-digit decimal number).

In Fig. 8-11 we have the start of a computer file listing the annual bonuses a company has decided to award to its employees. This file consists of consecutive 32-byte records, one per employee, in the format shown: 16 bytes for the name, 8 bytes for the position, and 8 bytes for the bonus. Each of the sixteen 8-byte blocks (numbered from 0 to 15) is encrypted by (triple) DES.

Figure 8-11. The plaintext of a file encrypted as 16 DES blocks.



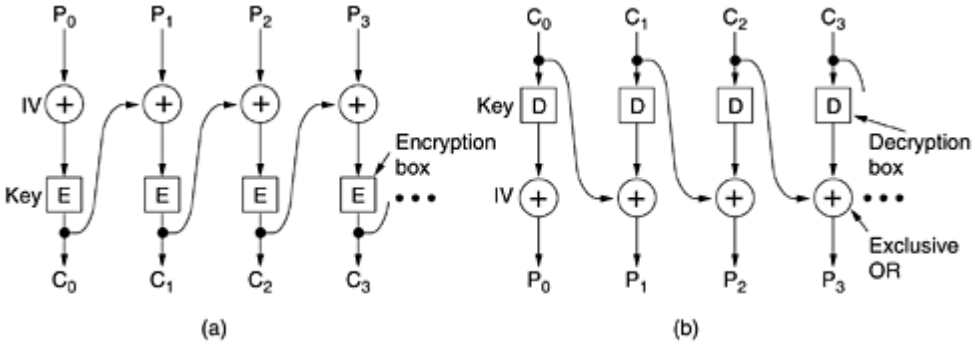
Leslie just had a fight with the boss and is not expecting much of a bonus. Kim, in contrast, is the boss' favorite, and everyone knows this. Leslie can get access to the file after it is encrypted but before it is sent to the bank. Can Leslie rectify this unfair situation, given only the encrypted file?

No problem at all. All Leslie has to do is make a copy of the 12th ciphertext block (which contains Kim's bonus) and use it to replace the 4th ciphertext block (which contains Leslie's bonus). Even without knowing what the 12th block says, Leslie can expect to have a much merrier Christmas this year. (Copying the 8th ciphertext block is also a possibility, but is more likely to be detected; besides, Leslie is not a greedy person.)

Cipher Block Chaining Mode

To thwart this type of attack, all block ciphers can be chained in various ways so that replacing a block the way Leslie did will cause the plaintext decrypted starting at the replaced block to be garbage. One way of chaining is cipher block chaining. In this method, shown in Fig. 8-12, each plaintext block is XORed with the previous ciphertext block before being encrypted. Consequently, the same plaintext block no longer maps onto the same ciphertext block, and the encryption is no longer a big monoalphabetic substitution cipher. The first block is XORed with a randomly chosen IV (Initialization Vector), which is transmitted (in plaintext) along with the ciphertext.

Figure 8-12. Cipher block chaining. (a) Encryption. (b) Decryption.



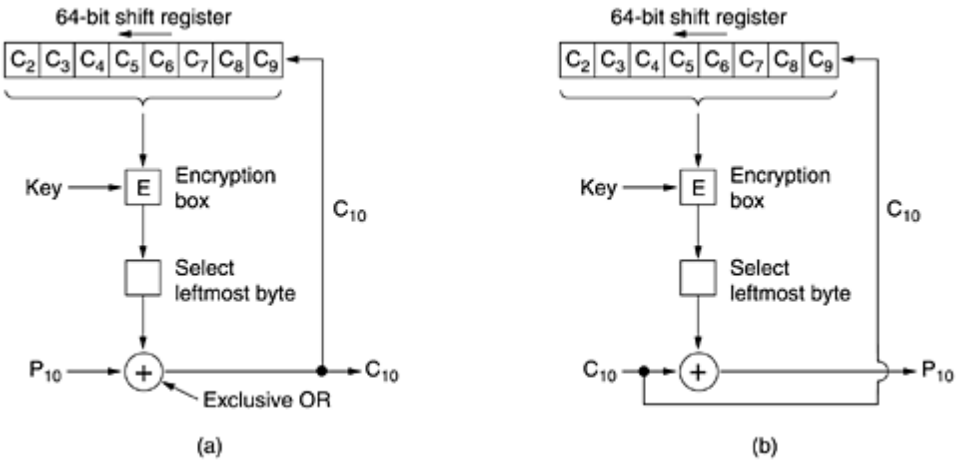
We can see how cipher block chaining mode works by examining the example of Fig. 8-12. We start out by computing C0 = E(P0 XOR IV). Then we compute C1 = E(P1 XOR C0), and so on. Decryption also uses XOR to reverse the process, with P0 = IV XOR D(C0), and so on. Note that the encryption of block i is a function of all the plaintext in blocks 0 through i - 1, so the same plaintext generates different ciphertext depending on where it occurs. A transformation of the type Leslie made will result in nonsense for two blocks starting at Leslie's bonus field. To an astute security officer, this peculiarity might suggest where to start the ensuing investigation.

Cipher block chaining also has the advantage that the same plaintext block will not result in the same ciphertext block, making cryptanalysis more difficult. In fact, this is the main reason it is used.

Cipher Feedback Mode

However, cipher block chaining has the disadvantage of requiring an entire 64-bit block to arrive before decryption can begin. For use with interactive terminals, where people can type lines shorter than eight characters and then stop, waiting for a response, this mode is unsuitable. For byte-by-byte encryption, cipher feedback mode, using (triple) DES is used, as shown in Fig. 8-13. For AES the idea is exactly the same, only a 128-bit shift register is used. In this figure, the state of the encryption machine is shown after bytes 0 through 9 have been encrypted and sent. When plaintext byte 10 arrives, as illustrated in Fig. 8-13(a), the DES algorithm operates on the 64-bit shift register to generate a 64-bit ciphertext. The leftmost byte of that ciphertext is extracted and XORed with P10. That byte is transmitted on the transmission line. In addition, the shift register is shifted left 8 bits, causing C2 to fall off the left end, and C10 is inserted in the position just vacated at the right end by C9. Note that the contents of the shift register depend on the entire previous history of the plaintext, so a pattern that repeats multiple times in the plaintext will be encrypted differently each time in the ciphertext. As with cipher block chaining, an initialization vector is needed to start the ball rolling.

Figure 8-13. Cipher feedback mode. (a) Encryption. (b) Decryption.



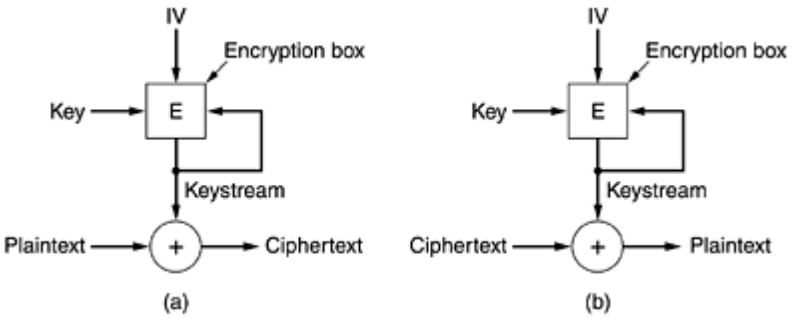
Decryption with cipher feedback mode just does the same thing as encryption. In particular, the content of the shift register is encrypted, not decrypted, so the selected byte that is XORed with C10 to get P10 is the same one that was XORed with P10 to generate C10 in the first place. As long as the two shift registers remain identical, decryption works correctly. It is illustrated in Fig. 8-13(b).

A problem with cipher feedback mode is that if one bit of the ciphertext is accidentally inverted during transmission, the 8 bytes that are decrypted while the bad byte is in the shift register will be corrupted. Once the bad byte is pushed out of the shift register, correct plaintext will once again be generated. Thus, the effects of a single inverted bit are relatively localized and do not ruin the rest of the message, but they do ruin as many bits as the shift register is wide.

Stream Cipher Mode

Nevertheless, applications exist in which having a 1-bit transmission error mess up 64 bits of plaintext is too large an effect. For these applications, a fourth option, stream cipher mode, exists. It works by encrypting an initialization vector, using a key to get an output block. The output block is then encrypted, using the key to get a second output block. This block is then encrypted to get a third block, and so on. The (arbitrarily large) sequence of output blocks, called the keystream, is treated like a one-time pad and XORed with the plaintext to get the ciphertext, as shown in Fig. 8-14(a). Note that the IV is used only on the first step. After that, the output is encrypted. Also note that the keystream is independent of the data, so it can be computed in advance, if need be, and is completely insensitive to transmission errors. Decryption is shown in Fig. 8-14(b).

Figure 8-14. A stream cipher. (a) Encryption. (b) Decryption.



Decryption occurs by generating the same keystream at the receiving side. Since the keystream depends only on the IV and the key, it is not affected by transmission errors in the ciphertext. Thus, a 1-bit error in the transmitted ciphertext generates only a 1-bit error in the decrypted plaintext.

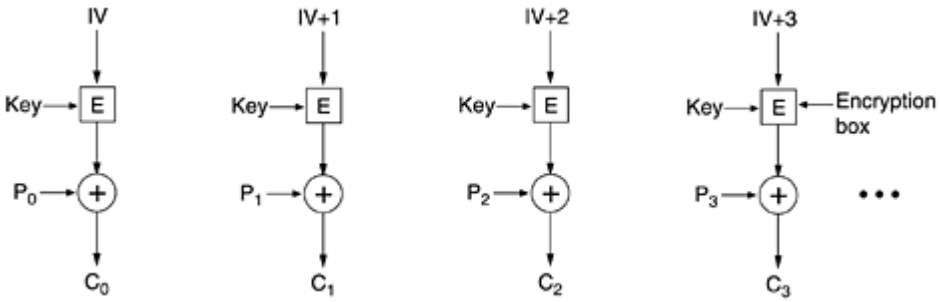
It is essential never to use the same (key, IV) pair twice with a stream cipher because doing so will generate the same keystream each time. Using the same keystream twice exposes the ciphertext to a keystream reuse attack. Imagine that the plaintext block, P0, is encrypted with the keystream to get P0 XOR K0. Later, a second plaintext block, Q0, is encrypted with the same keystream to get Q0 XOR K0. An intruder who captures both of these ciphertext blocks can simply XOR them together to get P0 XOR Q0, which eliminates the key. The intruder now has the XOR of the two plaintext blocks. If one of them is known or can be guessed, the other can also be found. In any event, the XOR of two plaintext streams can be attacked by using statistical properties of the message. For example, for English text, the most common character in the stream will probably be the XOR of two spaces, followed by the XOR of space and the letter "e", etc. In short, equipped with the XOR of two plaintexts, the cryptanalyst has an excellent chance of deducing both of them.

Counter Mode

One problem that all the modes except electronic code book mode have is that random access to encrypted data is impossible. For example, suppose a file is transmitted over a network and then stored on disk in encrypted form. This might be a reasonable way to operate if the receiving computer is a notebook computer that might be stolen. Storing all critical files in encrypted form greatly reduces the damage due to secret information leaking out in the event that the computer falls into the wrong hands.

However, disk files are often accessed in nonsequential order, especially files in databases. With a file encrypted using cipher block chaining, accessing a random block requires first decrypting all the blocks ahead of it, an expensive proposition. For this reason, yet another mode has been invented, counter mode,as illustrated in Fig. 8-15. Here the plaintext is not encrypted directly. Instead, the initialization vector plus a constant is encrypted, and the resulting ciphertext XORed with the plaintext. By stepping the initialization vector by 1 for each new block, it is easy to decrypt a block anywhere in the file without first having to decrypt all of its predecessors.

Figure 8-15. Encryption using counter mode.



Although counter mode is useful, it has a weakness that is worth pointing out. Suppose that the same key, K, is used again in the future (with a different plaintext but the same IV) and an attacker acquires all the ciphertext from both runs. The keystreams are the same in both cases, exposing the cipher to a keystream reuse attack of the same kind we saw with stream ciphers. All the cryptanalyst has to do is to XOR the two ciphertexts together to eliminate all the cryptographic protection and just get the XOR of the plaintexts. This weakness does not mean counter mode is a bad idea. It just means that both keys and initialization vectors should be chosen independently and at random. Even if the same key is accidentally used twice, if the IV is different each time, the plaintext is safe.

8.2.4 Other Ciphers

DES and Rijndael are the best-known symmetric-key, cryptographic algorithms. However, it is worth mentioning that numerous other symmetric-key ciphers have been devised. Some of these are embedded inside various products. A few of the more common ones are listed in Fig. 8-16.

Figure 8-16. Some common symmetric-key cryptographic algorithms.

Cipher	Author	Key length	Comments
Blowfish	Bruce Schneier	1–448 bits	Old and slow
DES	IBM	56 bits	Too weak to use now
IDEA	Massey and Xuejia	128 bits	Good, but patented
RC4	Ronald Rivest	1–2048 bits	Caution: some keys are weak
RC5	Ronald Rivest	128–256 bits	Good, but patented
Rijndael	Daemen and Rijmen	128–256 bits	Best choice
Serpent	Anderson, Biham, Knudsen	128–256 bits	Very strong
Triple DES	IBM	168 bits	Second best choice
Twofish	Bruce Schneier	128–256 bits	Very strong; widely used

8.2.5 Cryptanalysis

Before leaving the subject of symmetric-key cryptography, it is worth at least mentioning four developments in cryptanalysis. The first development is differential cryptanalysis (Biham and Shamir, 1993). This technique can be used to attack any block cipher. It works by beginning with a pair of plaintext blocks that differ in only a small number of bits and watching carefully what happens on each internal iteration as the encryption proceeds. In many cases, some bit patterns are much more common than other patterns, and this observation leads to a probabilistic attack.

The second development worth noting is linear cryptanalysis (Matsui, 1994). It can break DES with only 243 known plaintexts. It works by XORing certain bits in the plaintext and ciphertext together and examining the result for patterns. When this is done repeatedly, half the bits should be 0s and half should be 1s. Often, however, ciphers introduce a bias in one direction or the other, and this bias, however small, can be exploited to reduce the work factor. For the details, see Matsui's paper.

The third development is using analysis of the electrical power consumption to find secret keys. Computers typically use 3 volts to represent a 1 bit and 0 volts to represent a 0 bit. Thus, processing a 1 takes more electrical energy than processing a 0. If a cryptographic algorithm consists of a loop in which the key bits are processed in order, an attacker who replaces the main n-GHz clock with a slow (e.g., 100-Hz) clock and puts alligator clips on the CPU's power and ground pins, can precisely monitor the power consumed by each machine instruction. From this data, deducing the key is surprisingly easy. This kind of cryptanalysis can be defeated only by carefully coding the algorithm in assembly language to make sure power consumption is independent of the key and also independent of all the individual round keys.

The fourth development is timing analysis. Cryptographic algorithms are full of if statements that test bits in the round keys. If the then and else parts take different amounts of time, by slowing down the clock and seeing how long various steps take, it may also be possible to deduce the round keys. Once all the round keys are known, the original key can usually be computed. Power and timing analysis can also be employed simultaneously to make the job easier. While power and timing analysis may seem exotic, in reality they are powerful techniques that can break any cipher not specifically designed to resist them.

8.3 Public-Key Algorithms

Historically, distributing the keys has always been the weakest link in most cryptosystems. No matter how strong a cryptosystem was, if an intruder could steal the key, the system was worthless. Cryptologists always took for granted that the encryption key and decryption key were the same (or easily derived from one another). But the key had to be distributed to all users of the system. Thus, it seemed as if there was an inherent built-in problem. Keys had to be protected from theft, but they also had to be distributed, so they could not just be locked up in a bank vault.

In 1976, two researchers at Stanford University, Diffie and Hellman (1976), proposed a radically new kind of cryptosystem, one in which the encryption and decryption keys were different, and the decryption key could not feasibly be derived from the encryption key. In their proposal, the (keyed) encryption algorithm, E, and the (keyed) decryption algorithm, D, had to meet three requirements. These requirements can be stated simply as follows:

1.

1. D(E(P)) = P.
2.

2. It is exceedingly difficult to deduce D from E.
3.

3. E cannot be broken by a chosen plaintext attack.

The first requirement says that if we apply D to an encrypted message, E(P), we get the original plaintext message, P, back. Without this property, the legitimate receiver could not decrypt the ciphertext. The second requirement speaks for itself. The third requirement is needed because, as we shall see in a moment, intruders may experiment with the algorithm to their hearts' content. Under these conditions, there is no reason that the encryption key cannot be made public.

The method works like this. A person, say, Alice, wanting to receive secret messages, first devises two algorithms meeting the above requirements. The encryption algorithm and Alice's key are then made public, hence the name public-key cryptography. Alice might put her public key on her home page on the Web, for example. We will use the notation EA to mean the encryption algorithm parameterized by Alice's public key. Similarly, the (secret) decryption algorithm parameterized by Alice's private key is DA. Bob does the same thing, publicizing EB but keeping DB secret.

Now let us see if we can solve the problem of establishing a secure channel between Alice and Bob, who have never had any previous contact. Both Alice's encryption key, EA, and Bob's encryption key, EB, are assumed to be in publicly readable files. Now Alice takes her first message, P, computes EB(P), and sends it to Bob. Bob then decrypts it by applying his secret key DB [i.e., he computes DB(EB(P)) = P]. No one else can read the encrypted message, EB(P), because the encryption system is assumed strong and because it is too difficult to derive DB from the publicly known EB. To send a reply, R, Bob transmits EA(R). Alice and Bob can now communicate securely.

A note on terminology is perhaps useful here. Public-key cryptography requires each user to have two keys: a public key, used by the entire world for encrypting messages to be sent to that user, and a private key, which the user needs for decrypting messages. We will consistently refer to these keys as the public and private keys, respectively, and distinguish them from the secret keys used for conventional symmetric-key cryptography.

8.4 Digital Signatures

The authenticity of many legal, financial, and other documents is determined by the presence or absence of an authorized handwritten signature. And photocopies do not count. For computerized message systems to replace the physical transport of paper and ink documents, a method must be found to allow documents to be signed in an unforgeable way.

The problem of devising a replacement for handwritten signatures is a difficult one. Basically, what is needed is a system by which one party can send a signed message to another party in such a way that the following conditions hold:

1.

1. The receiver can verify the claimed identity of the sender.
2.

2. The sender cannot later repudiate the contents of the message.
3.

3. The receiver cannot possibly have concocted the message himself.

The first requirement is needed, for example, in financial systems. When a customer's computer orders a bank's computer to buy a ton of gold, the bank's computer needs to be able to make sure that the computer giving the order really belongs to the company whose account is to be debited. In other words, the bank has to authenticate the customer (and the customer has to authenticate the bank).

The second requirement is needed to protect the bank against fraud. Suppose that the bank buys the ton of gold, and immediately thereafter the price of gold drops sharply. A dishonest customer might sue the bank, claiming that he never issued any order to buy gold. When the bank produces the message in court, the customer denies having sent it. The property that no party to a contract can later deny having signed it is called nonrepudiation. The digital signature schemes that we will now study help provide it.

The third requirement is needed to protect the customer in the event that the price of gold shoots up and the bank tries to construct a signed message in which the customer asked for one bar of gold instead of one ton. In this fraud scenario, the bank just keeps the rest of the gold for itself.

8.4.1 Symmetric-Key Signatures

One approach to digital signatures is to have a central authority that knows everything and whom everyone trusts, say Big Brother (BB). Each user then chooses a secret key and carries it by hand to BB's office. Thus, only Alice and BB know Alice's secret key, KA, and so on.

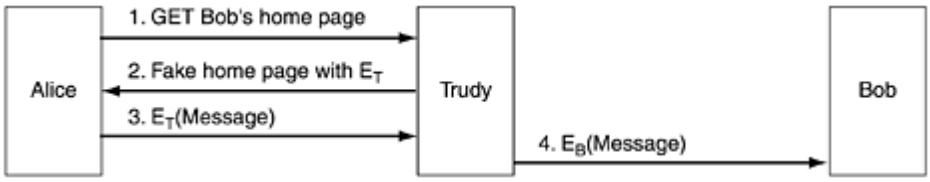
When Alice wants to send a signed plaintext message, P, to her banker, Bob, she generates KA(B, RA, t, P), where B is Bob's identity, RA is a random number chosen by Alice, t is a timestamp to ensure freshness, and KA(B, RA, t, P) is the message encrypted with her key, KA. Then she sends it as depicted in Fig. 8-18. BB sees that the message

8.5 Management of Public Keys

Public-key cryptography makes it possible for people who do not share a common key to communicate securely. It also makes signing messages possible without the presence of a trusted third party. Finally, signed message digests make it possible to verify the integrity of received messages easily.

However, there is one problem that we have glossed over a bit too quickly: if Alice and Bob do not know each other, how do they get each other's public keys to start the communication process? The obvious solution—put your public key on your Web site—does not work for the following reason. Suppose that Alice wants to look up Bob's public key on his Web site. How does she do it? She starts by typing in Bob's URL. Her browser then looks up the DNS address of Bob's home page and sends it a GET request, as shown in Fig. 8-23. Unfortunately, Trudy intercepts the request and replies with a fake home page, probably a copy of Bob's home page except for the replacement of Bob's public key with Trudy's public key. When Alice now encrypts her first message with ET, Trudy decrypts it, reads it, reencrypts it with Bob's public key, and sends it to Bob, who is none the wiser that Trudy is reading his incoming messages. Worse yet, Trudy could modify the messages before reencrypting them for Bob. Clearly, some mechanism is needed to make sure that public keys can be exchanged securely.

Figure 8-23. A way for Trudy to subvert public-key encryption.



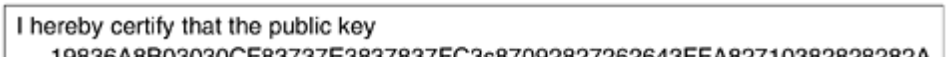
8.5.1 Certificates

As a first attempt at distributing public keys securely, we could imagine a key distribution center available on-line 24 hours a day to provide public keys on demand. One of the many problems with this solution is that it is not scalable, and the key distribution center would rapidly become a bottleneck. Also, if it ever went down, Internet security would suddenly grind to a halt.

For these reasons, people have developed a different solution, one that does not require the key distribution center to be on-line all the time. In fact, it does not have to be on-line at all. Instead, what it does is certify the public keys belonging to people, companies, and other organizations. An organization that certifies public keys is now called a CA (Certification Authority).

As an example, suppose that Bob wants to allow Alice and other people to communicate with him securely. He can go to the CA with his public key along with his passport or driver's license and ask to be certified. The CA then issues a certificate similar to the one in Fig. 8-24 and signs its SHA-1 hash with the CA's private key. Bob then pays the CA's fee and gets a floppy disk containing the certificate and its signed hash.

Figure 8-24. A possible certificate and its signed hash.



8.6 Communication Security

We have now finished our study of the tools of the trade. Most of the important techniques and protocols have been covered. The rest of the chapter is about how these techniques are applied in practice to provide network security, plus some thoughts about the social aspects of security at the end of the chapter.

In the following four sections, we will look at communication security, that is, how to get the bits secretly and without modification from source to destination and how to keep unwanted bits outside the door. These are by no means the only security issues in networking, but they are certainly among the most important ones, making this a good place to start.

8.6.1 IPsec

IETF has known for years that security was lacking in the Internet. Adding it was not easy because a war broke out about where to put it. Most security experts believe that to be really secure, encryption and integrity checks have to be end to end (i.e., in the application layer). That is, the source process encrypts and/or integrity protects the data and sends that to the destination process where it is decrypted and/or verified. Any tampering done in between these two processes, including within either operating system, can then be detected. The trouble with this approach is that it requires changing all the applications to make them security aware. In this view, the next best approach is putting encryption in the transport layer or in a new layer between the application layer and the transport layer, making it still end to end but not requiring applications to be changed.

The opposite view is that users do not understand security and will not be capable of using it correctly and nobody wants to modify existing programs in any way, so the network layer should authenticate and/or encrypt packets without the users being involved. After years of pitched battles, this view won enough support that a network layer security standard was defined. In part the argument was that having network layer encryption does not prevent security-aware users from doing it right and it does help security-unaware users to some extent.

The result of this war was a design called IPsec (IP security), which is described in RFCs 2401, 2402, and 2406, among others. Not all users want encryption (because it is computationally expensive). Rather than make it optional, it was decided to require encryption all the time but permit the use of a null algorithm. The null algorithm is described and praised for its simplicity, ease of implementation, and great speed in RFC 2410.

The complete IPsec design is a framework for multiple services, algorithms and granularities. The reason for multiple services is that not everyone wants to pay the price for having all the services all the time, so the services are available a la carte. The major services are secrecy, data integrity, and protection from replay attacks (intruder replays a conversation). All of these are based on symmetric-key cryptography because high performance is crucial.

The reason for having multiple algorithms is that an algorithm that is now thought to be secure may be broken in the future. By making IPsec algorithm-independent, the framework can survive even if some particular algorithm is later broken.

The reason for having multiple granularities is to make it possible to protect a single TCP connection, all traffic between a pair of hosts, or all traffic between a pair of secure routers, among other possibilities.

8.7 Authentication Protocols

Authentication is the technique by which a process verifies that its communication partner is who it is supposed to be and not an imposter. Verifying the identity of a remote process in the face of a malicious, active intruder is surprisingly difficult and requires complex protocols based on cryptography. In this section, we will study some of the many authentication protocols that are used on insecure computer networks.

As an aside, some people confuse authorization with authentication. Authentication deals with the question of whether you are actually communicating with a specific process. Authorization is concerned with what that process is permitted to do. For example, a client process contacts a file server and says: I am Scott's process and I want to delete the file cookbook.old. From the file server's point of view, two questions must be answered:

1.

1. Is this actually Scott's process (authentication)?
2.

2. Is Scott allowed to delete cookbook.old (authorization)?

Only after both of these questions have been unambiguously answered in the affirmative can the requested action take place. The former question is really the key one. Once the file server knows to whom it is talking, checking authorization is just a matter of looking up entries in local tables or databases. For this reason, we will concentrate on authentication in this section.

The general model that all authentication protocols use is this. Alice starts out by sending a message either to Bob or to a trusted KDC (Key Distribution Center), which is expected to be honest. Several other message exchanges follow in various directions. As these messages are being sent Trudy may intercept, modify, or replay them in order to trick Alice and Bob or just to gum up the works.

Nevertheless, when the protocol has been completed, Alice is sure she is talking to Bob and Bob is sure he is talking to Alice. Furthermore, in most of the protocols, the two of them will also have established a secret session key for use in the upcoming conversation. In practice, for performance reasons, all data traffic is encrypted using symmetric-key cryptography (typically AES or triple DES), although public-key cryptography is widely used for the authentication protocols themselves and for establishing the session key.

The point of using a new, randomly-chosen session key for each new connection is to minimize the amount of traffic that gets sent with the users' secret keys or public keys, to reduce the amount of ciphertext an intruder can obtain, and to minimize the damage done if a process crashes and its core dump falls into the wrong hands. Hopefully, the only key present then will be the session key. All the permanent keys should have been carefully zeroed out after the session was established.

8.7.1 Authentication Based on a Shared Secret Key

For our first authentication protocol, we will assume that Alice and Bob already share a secret key, KAB . This shared key might have been agreed upon on the telephone or in person, but, in any event, not on the (insecure)

8.8 E-Mail Security

When an e-mail message is sent between two distant sites, it will generally transit dozens of machines on the way. Any of these can read and record the message for future use. In practice, privacy is nonexistent, despite what many people think. Nevertheless, many people would like to be able to send e-mail that can be read by the intended recipient and no one else: not their boss and not even their government. This desire has stimulated several people and groups to apply the cryptographic principles we studied earlier to e-mail to produce secure e-mail. In the following sections we will study a widely-used secure e-mail system, PGP, and then briefly mention two others, PEM and S/MIME. For additional information about secure e-mail, see (Kaufman et al., 2002; and Schneier, 1995).

8.8.1 PGP—Pretty Good Privacy

Our first example, PGP (Pretty Good Privacy) is essentially the brainchild of one person, Phil Zimmermann (Zimmermann, 1995a, 1995b). Zimmermann is a privacy advocate whose motto is: If privacy is outlawed, only outlaws will have privacy. Released in 1991, PGP is a complete e-mail security package that provides privacy, authentication, digital signatures, and compression, all in an easy-to-use form. Furthermore, the complete package, including all the source code, is distributed free of charge via the Internet. Due to its quality, price (zero), and easy availability on UNIX, Linux, Windows, and Mac OS platforms, it is widely used today.

PGP encrypts data by using a block cipher called IDEA (International Data Encryption Algorithm), which uses 128-bit keys. It was devised in Switzerland at a time when DES was seen as tainted and AES had not yet been invented. Conceptually, IDEA is similar to DES and AES: it mixes up the bits in a series of rounds, but the details of the mixing functions are different from DES and AES. Key management uses RSA and data integrity uses MD5, topics that we have already discussed.

PGP has also been embroiled in controversy since day 1 (Levy, 1993). Because Zimmermann did nothing to stop other people from placing PGP on the Internet, where people all over the world could get it, the U.S. Government claimed that Zimmermann had violated U.S. laws prohibiting the export of munitions. The U.S. Government's investigation of Zimmermann went on for 5 years, but was eventually dropped, probably for two reasons. First, Zimmermann did not place PGP on the Internet himself, so his lawyer claimed that he never exported anything (and then there is the little matter of whether creating a Web site constitutes export at all). Second, the government eventually came to realize that winning a trial meant convincing a jury that a Web site containing a downloadable privacy program was covered by the arms-trafficking law prohibiting the export of war materiel such as tanks, submarines, military aircraft, and nuclear weapons. Years of negative publicity probably did not help much, either.

As an aside, the export rules are bizarre, to put it mildly. The government considered putting code on a Web site to be an illegal export and harassed Zimmermann for 5 years about it. On the other hand, when someone published the complete PGP source code, in C, as a book (in a large font with a checksum on each page to make scanning it in easy) and then exported the book, that was fine with the government because books are not classified as munitions. The sword is mightier than the pen, at least for Uncle Sam.

Another problem PGP ran into involved patent infringement. The company holding the RSA patent, RSA Security, Inc., alleged that PGP's use of the RSA algorithm infringed on its patent, but that problem was settled with releases starting at 2.6. Furthermore, PGP uses another patented encryption algorithm, IDEA, whose use caused some problems at first.

8.9 Web Security

We have just studied two important areas where security is needed: communications and e-mail. You can think of these as the soup and appetizer. Now it is time for the main course: Web security. The Web is where most of the Trudies hang out nowadays and do their dirty work. In the following sections we will look at some of the problems and issues relating to Web security.

Web security can be roughly divided into three parts. First, how are objects and resources named securely? Second, how can secure, authenticated connections be established? Third, what happens when a Web site sends a client a piece of executable code? After looking at some threats, we will examine all these issues.

8.9.1 Threats

One reads about Web site security problems in the newspaper almost weekly. The situation is really pretty grim. Let us look at a few examples of what has already happened. First, the home page of numerous organizations has been attacked and replaced by a new home page of the crackers' choosing. (The popular press calls people who break into computers "hackers," but many programmers reserve that term for great programmers. We prefer to call these people "crackers.") Sites that have been cracked include Yahoo, the U.S. Army, the CIA, NASA, and the New York Times. In most cases, the crackers just put up some funny text and the sites were repaired within a few hours.

Now let us look at some much more serious cases. Numerous sites have been brought down by denial-of-service attacks, in which the cracker floods the site with traffic, rendering it unable to respond to legitimate queries. Often the attack is mounted from a large number of machines that the cracker has already broken into (DDoS atacks). These attacks are so common that they do not even make the news any more, but they can cost the attacked site thousands of dollars in lost business.

In 1999, a Swedish cracker broke into Microsoft's Hotmail Web site and created a mirror site that allowed anyone to type in the name of a Hotmail user and then read all of the person's current and archived e-mail.

In another case, a 19-year-old Russian cracker named Maxim broke into an e-commerce Web site and stole 300,000 credit card numbers. Then he approached the site owners and told them that if they did not pay him $100,000, he would post all the credit card numbers to the Internet. They did not give in to his blackmail, and he indeed posted the credit card numbers, inflicting great damage to many innocent victims.

In a different vein, a 23-year-old California student e-mailed a press release to a news agency falsely stating that the Emulex Corporation was going to post a large quarterly loss and that the C.E.O. was resigning immediately. Within hours, the company's stock dropped by 60%, causing stockholders to lose over $2 billion. The perpetrator made a quarter of a million dollars by selling the stock short just before sending the announcement. While this event was not a Web site break-in, it is clear that putting such an announcement on the home page of any big corporation would have a similar effect.

We could (unfortunately) go on like this for many pages. But it is now time to examine some of the technical issues related to Web security. For more information about security problems of all kinds, see (Anderson, 2001; Garfinkel with Spafford, 2002; and Schneier, 2000). Searching the Internet will also turn up vast numbers of specific cases.

8.10 Social Issues

The Internet and its security technology is an area where social issues, public policy, and technology meet head on, often with huge consequences. Below we will just briefly examine three areas: privacy, freedom of speech, and copyright. Needless to say, we can only scratch the surface here. For additional reading, see (Anderson, 2001; Garfinkel with Spafford, 2002; and Schneier, 2000). The Internet is also full of material. Just type words such as "privacy," "censorship," and "copyright" into any search engine. Also, see this book's Web site for some links.

8.10.1 Privacy

Do people have a right to privacy? Good question. The Fourth Amendment to the U.S. Constitution prohibits the government from searching people's houses, papers, and effects without good reason, and goes on to restrict the circumstances under which search warrants shall be issued. Thus, privacy has been on the public agenda for over 200 years, at least in the U.S.

What has changed in the past decade is both the ease with which governments can spy on their citizens and the ease with which the citizens can prevent such spying. In the 18th century, for the government to search a citizen's papers, it had to send out a policeman on a horse to go to the citizen's farm demanding to see certain documents. It was a cumbersome procedure. Nowadays, telephone companies and Internet providers readily provide wiretaps when presented with search warrants. It makes life much easier for the policeman and there is no danger of falling off the horse.

Cryptography changes all that. Anybody who goes to the trouble of downloading and installing PGP and who uses a well-guarded alien-strength key can be fairly sure that nobody in the known universe can read his e-mail, search warrant or no search warrant. Governments well understand this and do not like it. Real privacy means it is much harder for them to spy on criminals of all stripes, but it is also much harder to spy on journalists and political opponents. Consequently, some governments restrict or forbid the use or export of cryptography. In France, for example, prior to 1999, all cryptography was banned unless the government was given the keys.

France was not alone. In April 1993, the U.S. Government announced its intention to make a hardware cryptoprocessor, the clipper chip, the standard for all networked communication. In this way, it was said, citizens' privacy would be guaranteed. It also mentioned that the chip provided the government with the ability to decrypt all traffic via a scheme called key escrow, which allowed the government access to all the keys. However, it promised only to snoop when it had a valid search warrant. Needless to say, a huge furor ensued, with privacy advocates denouncing the whole plan and law enforcement officials praising it. Eventually, the government backed down and dropped the idea.

A large amount of information about electronic privacy is available at the Electronic Frontier Foundation's Web site, www.eff.org.

Anonymous Remailers

PGP, SSL, and other technologies make it possible for two parties to establish secure, authenticated communication, free from third-party surveillance and interference. However, sometimes privacy is best served by not having authentication, in fact by making communication anonymous. The anonymity may be desired for point-to-point

8.11 Summary

Cryptography is a tool that can be used to keep information confidential and to ensure its integrity and authenticity. All modern cryptographic systems are based on Kerckhoff's principle of having a publicly-known algorithm and a secret key. Many cryptographic algorithms use complex transformations involving substitutions and permutations to transform the plaintext into the ciphertext. However, if quantum cryptography can be made practical, the use of one-time pads may provide truly unbreakable cryptosystems.

Cryptographic algorithms can be divided into symmetric-key algorithms and public-key algorithms. Symmetric-key algorithms mangle the bits in a series of rounds parameterized by the key to turn the plaintext into the ciphertext. Triple DES and Rijndael (AES) are the most popular symmetric-key algorithms at present. These algorithms can be used in electronic code book mode, cipher block chaining mode, stream cipher mode, counter mode, and others.

Public-key algorithms have the property that different keys are used for encryption and decryption and that the decryption key cannot be derived from the encryption key. These properties make it possible to publish the public key. The main public-key algorithm is RSA, which derives its strength from the fact that it is very difficult to factor large numbers.

Legal, commercial, and other documents need to be signed. Accordingly, various schemes have been devised for digital signatures, using both symmetric-key and public-key algorithms. Commonly, messages to be signed are hashed using algorithms such as MD5 or SHA-1, and then the hashes are signed rather than the original messages.

Public-key management can be done using certificates, which are documents that bind a principal to a public key. Certificates are signed by a trusted authority or by someone (recursively) approved by a trusted authority. The root of the chain has to be obtained in advance, but browsers generally have many root certificates built into them.

These cryptographic tools can be used to secure network traffic. IPsec operates in the network layer, encrypting packet flows from host to host. Firewalls can screen traffic going into or out of an organization, often based on the protocol and port used. Virtual private networks can simulate an old leased-line network to provide certain desirable security properties. Finally, wireless networks need good security and 802.11's WEP does not provide it, although 802.11i should improve matters considerably.

When two parties establish a session, they have to authenticate each other and if need be, establish a shared session key. Various authentication protocols exist, including some that use a trusted third party, Diffie-Hellman, Kerberos, and public-key cryptography.

E-mail security can be achieved by a combination of the techniques we have studied in this chapter. PGP, for example, compresses messages, then encrypts them using IDEA. It sends the IDEA key encrypted with the receiver's public key. In addition, it also hashes the message and sends the signed hash to verify message integrity.

Web security is also an important topic, starting with secure naming. DNSsec provides a way to prevent DNS spoofing, as do self-certifying names. Most e-commerce Web sites use SSL to establish secure, authenticated sessions between the client and server. Various techniques are used to deal with mobile code, especially sandboxing and code signing.

Chapter 9. Reading List and Bibliography

We have now finished our study of computer networks, but this is only the beginning. Many interesting topics have not been treated in as much detail as they deserve, and others have been omitted altogether for lack of space. In this chapter we provide some suggestions for further reading and a bibliography, for the benefit of readers who wish to continue their study of computer networks.

9.1 Suggestions for Further Reading

There is an extensive literature on all aspects of computer networks. Three journals that frequently publish papers in this area are IEEE Transactions on Communications, IEEE Journal on Selected Areas in Communications, and Computer Communication Review. Many other journals also publish occasional papers on the subject.

IEEE also publishes three magazines—IEEE Internet Computing, IEEE Network Magazine, and IEEE Communications Magazine—that contain surveys, tutorials, and case studies on networking. The first two emphasize architecture, standards, and software, and the last tends toward communications technology (fiber optics, satellites, and so on).

In addition, there are a number of annual or biannual conferences that attract numerous papers on networks and distributed systems, in particular, SIGCOMM Annual Conference, The International Conference on Distributed Computer Systems. and The Symposium on Operating Systems Principles,

Below we list some suggestions for supplementary reading, keyed to the chapters of this book. Most of these are tutorials or surveys on the subject. A few are chapters from textbooks.

9.1.1 Introduction and General Works

Bi et al., "Wireless Mobile Communications at the Start of the 21st Century"

A new century, a new technology. Sounds good. After some history of wireless, the major topics are covered here, including standards, applications, Internet, and technologies.

Comer, The Internet Book

Anyone looking for an easy-going introduction to the Internet should look here. Comer describes the history, growth, technology, protocols, and services of the Internet in terms that novices can understand, but so much material is covered that the book is also of interest to more technical readers.

Garber, "Will 3G Really Be the Next Big Wireless Technology?"

Third-generation mobile phones are supposed to combine voice and data and provide data rates up to 2 Mbps. They have been slow to take off. The promises, pitfalls, technology, politics, and economics of using broadband wireless communication are all covered in this easy-to-read article.

IEEE Internet Computing, Jan.-Feb. 2000

The first issue of IEEE Internet Computing in the new millennium did exactly what you would expect: ask the people who helped create the Internet in the previous millennium to speculate on where it is going in the next one. The experts are Paul Baran, Lawrence Roberts, Leonard Kleinrock, Stephen Crocker, Danny Cohen, Bob Metcalfe, Bill Gates, Bill Joy, and others. For best results, wait 500 years, then read their predictions.

Kipnis, "Beating the System: Abuses of the Standards Adoption Process"

Standards committees try to be fair and vendor neutral in their work, but unfortunately there are companies that try to abuse the system. For example, it has happened repeatedly that a company helps develop a standard and after it is approved, announces that the standard is based on a patent it owns and which it will license to companies that it likes and not to companies that it does not like, and at prices that it alone determines. For a look at the dark side of standardization, this article is an excellent start.

Kyas and Crawford, ATM Networks

ATM was once touted as the networking protocol of the future, and is still important within the telephone system. This book is an up-to-date guide to ATM's current status, with detailed information on ATM protocols and how they can integrate with IP-based networks.

9.2 Alphabetical Bibliography

ABRAMSON, N.: "Internet Access Using VSATs," IEEE Commun. Magazine, vol. 38, pp. 60-68, July 2000.

ABRAMSON,N.: "Development of the ALOHANET," IEEE Trans. on Information Theory, vol. IT-31, pp. 119-123, March 1985.

ADAMS, M., and DULCHINOS, D.: "OpenCable," IEEE Commun. Magazine, vol. 39, pp. 98-105, June 2001.

ALKHATIB, H.S., BAILEY, C., GERLA, M., and MCRAE, J.: "Wireless Data Networks: Reaching the Extra Mile," Computer, vol. 30, pp. 59-62, Dec. 1997.

ANDERSON, R.J.: "Free Speech Online and Office," Computer, vol. 25, pp. 28-30, June 2002.

ANDERSON, R.J.: Security Engineering, New York: Wiley, 2001.

ANDERSON, R.J.: "The Eternity Service," Proc. First Int'l Conf. on Theory and Appl. of Cryptology, CTU Publishing House, 1996.

ANDERSON, R.J.: "Why Cryptosystems Fail," Commun. of the ACM, vol. 37, pp. 32-40, Nov. 1994.

ARTZ, D.: "Digital Steganography," IEEE Internet Computing, vol. 5, pp. 75-80, 2001.

AZZAM, A.A., and RANSOM, N.: Broadband Access Technologies, New York: McGraw-Hill, 1999.

BAKNE, A., and BADRINATH, B.R.: "I-TCP: Indirect TCP for Mobile Hosts," Proc. 15th Int'l Conf. on Distr. Computer Systems, IEEE, pp. 136-143, 1995.

BALAKRISHNAN, H., SESHAN, S., and KATZ, R.H.: "Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks," Proc. ACM Mobile Computing and Networking Conf., ACM, pp. 2-11, 1995.

BALLARDIE, T., FRANCIS, P., and CROWCROFT, J.: "Core Based Trees (CBT)," Proc. SIGCOMM '93 Conf., ACM, pp. 85-95, 1993.

BARAKAT, C., ALTMAN, E., and DABBOUS, W.: "On TCP Performance in a Heterogeneous Network: A Survey," IEEE Commun. Magazine, vol. 38, pp. 40-46, Jan. 2000.

BELLAMY, J.: Digital Telephony, 3rd ed., New York: Wiley, 2000.