TABLE 2.4 Telnet commands.

Name	Code	Meaning
EOF	236	End of file.
SUSP	237	Suspend cursor process.
ABORT	238	Abort process.
EOR	239	End of record.
SE	240	End of subnegotiation parameters.
NOP	241	No operation.
Data mark	242	The data stream portion of a synch signal. This code should always be accompanied by a TCP urgent notification.
Break	243	NVT character BRK.
Interrupt process	244	The function IP.
Abort output	245	The function AO.
Are you there	246	The function AYT.
Erase character	247	The function EC.
Erase line	248	The function EL.
Go ahead	249	The GA signal.
SB	250	Indicates that what follows is subnegotiation of the indicated option.
WILL (option code)	251	Option negotiation.
WONT (option code)	252	Option negotiation.
DO (option code)	253	Option negotiation.
DONT (option code)	254	Option negotiation.
IAC	255	Data byte 255.

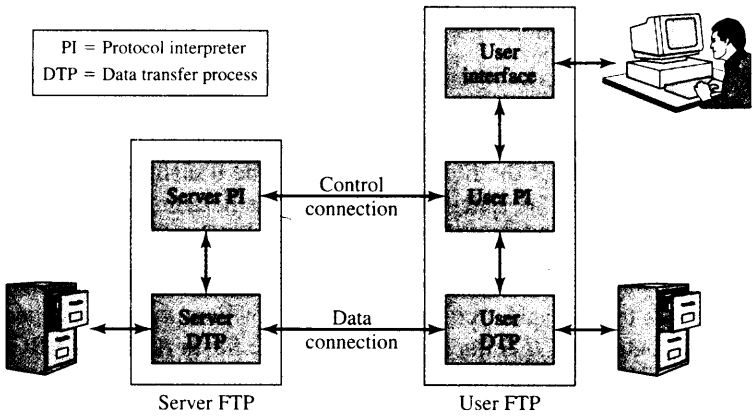


FIGURE 2.23 Transferring files using FTP.

FTP requires two TCP connections to transfer a file. One is the *control connection* that is established on port 21 at the server. The second TCP connection is a *data connection* used to perform a file transfer. A data connection must be established for each file transferred. Data connections are used for transferring a file in either direction or for obtaining lists of files or directories from the server to the client. Figure 2.23 shows the role of the two connections in FTP.

A control connection is established following the Telnet protocol from the user to the server port. FTP commands and replies are exchanged via the control connection. The user protocol interpreter (PI) is responsible for sending FTP commands and interpreting the replies. The server PI is responsible for interpreting commands, sending replies, and directing the server data transfer process (DTP) to establish a data connection and transfer. The commands are used to specify information about the data connection and about the particular file system operation being requested.

A data connection is established usually upon request from the user for some sort of file operation. The user PI usually chooses an ephemeral port number for its end of the operation and then issues a passive open from this port. The port number is then sent to the server PI using a PORT command. Upon receipt of the port number via the control connection, the server issues an active open to that same port. The server always uses port 20 for its end of the data connection. The user DTP then waits for the server to initiate and perform the file operation.

Note that the data connection may be used to send and receive simultaneously. Note also that the user may initiate a file transfer between two nonlocal machines, for example, between two servers. In this case there would be a control connection between the user and both servers but only one data connection, namely, the one between the two servers.

The user is responsible for requesting a close of the control connection, although the server performs the action. If the control connection is closed while the data connection is still open, then the server may terminate the data transfer. The data connection is usually closed by the server. The main exception is when the user DTP closes the data connection to indicate an end of file for a stream transmission. Note that FTP is not designed to detect lost or scrambled bits; the responsibility for error detection is left to TCP.

The Telnet protocol works across different systems because it specifies a common starting point for terminal emulation. FTP works across different systems because it can accommodate several different file types and structures. FTP commands are used to specify information about the file and how it will be transmitted. In general, three types of information must be specified. Note that the default specifications must be supported by every FTP implementation.

1. *File type.* FTP supports ASCII, EBCDIC, image (binary), or local. Local specifies that the data is to be transferred in logical bytes, where the size is specified in a separate parameter. *ASCII is the default type.* If the file is ASCII or EBCDIC, then a vertical format control may also be specified.
2. *Data structure.* FTP supports file structure (a continuous stream of bytes with no internal structure), record structure (used with text files), and page structure (file consists of independent indexed pages). *File structure is the default specification.*
3. *Transmission mode.* FTP supports stream, block, or compressed mode. When transmission is in stream mode, the user DTP closes the connection to indicate the end of file for data with file structure. If the data has block structure, then a special two-byte sequence indicates end of record and end of file. *The default is stream mode.*

An *FTP command* consists of three or four bytes of uppercase ASCII characters followed by a space if parameters follow, or by a Telnet end of option list (EOL) otherwise.

FTP commands fall into one of the following categories: access control identification, data transfer parameters, and FTP service requests. Table 2.5 lists some of the common FTP commands encountered.

Every command must produce at least one *FTP reply.* The replies are used to synchronize requests and actions and to keep the client informed of the state of the server. A reply consists of a three-digit number (in alphanumeric representation) followed by some text. The numeric code is intended for the user PI; the text, if processed, is intended for the user. For example, the reply issued following a successful connection termination request is "`221 Goodbye.`" The first digit indicates whether and to what extent the specified request has been completed. The second digit indicates the category of the reply, and the third digit provides additional information about the particular category. Table 2.6 lists the possible values of the first two digits and their meanings.

In this case of the goodbye message, the first 2 indicates a successful completion. The second digit is also 2 to indicate that the reply pertains to a connection request.

TABLE 2.5 Some common FTP commands.

Command	Meaning
`ABOR`	Abort the previous FTP command and any data transfer.
`LIST`	List files or directories.
`QUIT`	Log off from server.
`RETR filename`	Retrieve the specified file.
`STOR filename`	Store the specified file.

TABLE 2.6 FTP replies—the first and second digits.

Reply	Meaning
`1yz`	Positive preliminary reply (action has begun, but wait for another reply before sending a new command).
`2yz`	Positive completion reply (action completed successfully; new command may be sent).
`3yz`	Positive intermediary reply (command accepted, but action cannot be performed without additional information; user should send a command with the necessary information).
`4yz`	Transient negative completion reply (action currently cannot be performed; resend command later).
`5yz`	Permanent negative completion reply (action cannot be performed; do not resend it).
`x0z`	Syntax errors.
`x1z`	Information (replies to requests for status or help).
`x2z`	Connections (replies referring to the control and data connections).
`x3z`	Authentication and accounting (replies for the login process and accounting procedures).
`x4z`	Unspecified.
`x5z`	File system status.

2.5.3 Hypertext Transfer Protocol and the World Wide Web

The World Wide Web (WWW) provides a framework for accessing documents and resources that are located in computers connected to the Internet. These documents consist of text, graphics and other media and are interconnected by *hyperlinks* or *links* that appear within the documents. The **Hypertext Markup Language (HTML)** is used to prepare these documents. The WWW is accessed through a browser program that interprets HTML, displays the documents, and allows the user to access other documents by clicking on these links.

Each link provides the browser with a uniform resource locator (URL) that specifies the name of the machine where the document is located as well as the name of the file that contains the requested document. For example, the sequence of packet exchanges captured in Figure 2.18 and Figure 2.19 result after clicking on the URL http://www.nytimes.com/. The first term 'http' specifies the retrieval mechanism to be used, in this case, the HTTP protocol. The next term specifies the name of the host machine, namely, www.nytimes.com. The remaining term gives the path component, that is, it identifies the file on that server containing the desired article. In this example, the final slash (/) refers to the server root. By clicking a highlighted item in a browser page the user begins an interaction to obtain the desired file from the server where it is stored. The **Hypertext Transfer Protocol (HTTP)** is the application layer protocol that defines the interaction between the web client and the web server.

HTTP

HTTP is a client/server application defined in RFC 1945 and RFC 2616 to support communications between web browsers and web servers. HTTP defines how the client makes the request for an object (typically a document) and how the server replies with a response message. The typical interaction is shown in Figure 2.18. After the user clicks on a link, the browser program must first resolve the URL to an IP address by invoking the DNS protocol (frame 1 in the figure). Once the IP address is returned (frame 2), the HTTP client must set up a TCP connection to the desired server over well-known port 80 (frames 2, 3, and 4). The HTTP client and server are then ready to exchange messages: the client sends a GET message requesting the document, and the server replies with a response followed by the desired document.

HTTP is a *stateless* protocol in that it does not maintain any information ("state") about its clients. In other words, the HTTP server handles each request independently of all other requests. Thus if a client sends a request multiple times, the server handles each request in the same manner. HTTP was designed to be stateless in order to keep it simple. This design allows requests to be handled quickly and enables servers to handle large volumes of requests per second.

The initial design of HTTP/1.0 (and earlier versions) uses *nonpersistent connections*. The TCP connection is closed after each request-response interaction. Each subsequent request from the same client to the same server involves the setting up and tearing down of an additional TCP connection. From the example in Figure 2.18, we can see that each TCP connection setup involves the exchange of three segments between the client and server machines and hence the sending of the request is delayed

by multiple round-trip times.[11] Another disadvantage of nonpersistent connections is that TCP processing and memory resources are wasted in the server and the client. HTTP/1.1 made *persistent connections* the default mode. The server now keeps the TCP connection open for a certain period of time after sending a response. This enables the client to make multiple requests over the same TCP connection and hence avoid the inefficiency and delay of the nonpersistent mode.

MESSAGE FORMATS

Like Telnet and FTP, HTTP messages are written in ASCII text and can be read and interpreted readily. Figure 2.24 (middle pane) shows a typical HTTP request message. The first line of a request message is the *request line*, and subsequent lines are called *header lines*. Each line is written in ASCII text and terminated by a carriage return followed by a line feed character. The last header line is followed by an extra carriage return and line feed. Some request messages include an *entity body* that follows the header section and provides additional information to the server.

The request line has the following form: *Method URL HTTP-Version* \r\n. The *Method* field specifies the action method or action that is applied to the object. The second field identifies the object, and the remaining field is the HTTP version. In the first line of the HTTP section in the middle pane of Figure 2.24, the method is

[11] A round-trip time (RTT) is the time that elapses from when a message is sent from a transmitter to when a response is received back from the receiver.

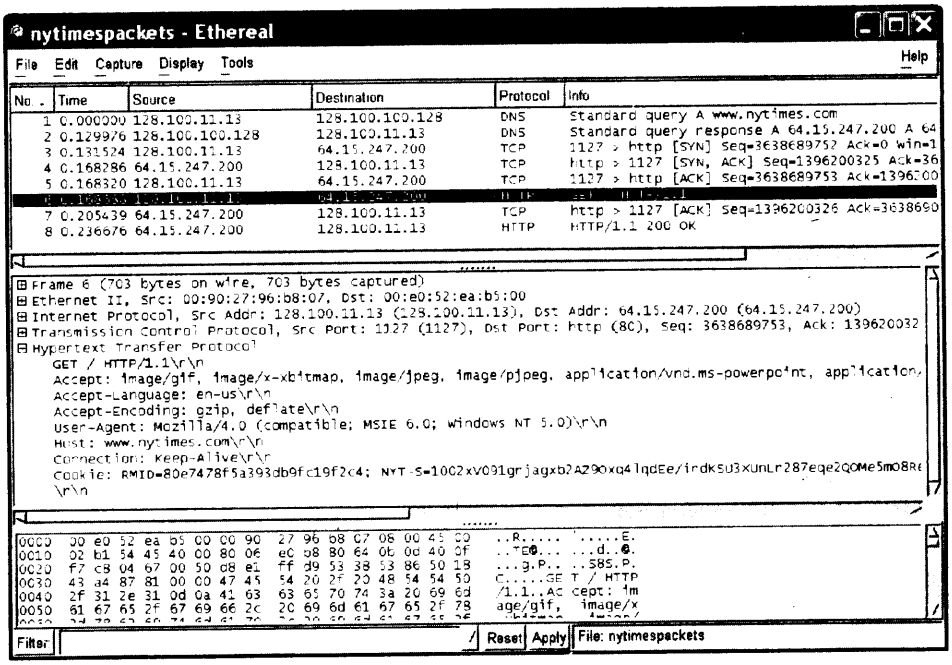


FIGURE 2.24 Ethereal capture of an HTTP GET message.

TABLE 2.7 HTTP request methods.

Request method	Meaning
GET	Retrieve information (object) identified by the URL.
HEAD	Retrieve meta-information about the object, but do not transfer the object; Can be used to find out if a document has changed.
POST	Send information to a URL (using the entity body) and retrieve result; used when a user fills out a form in a browser.
PUT	Store information in location named by URL.
DELETE	Remove object identified by URL.
TRACE	Trace HTTP forwarding through proxies, tunnels, etc.
OPTIONS	Used to determine the capabilities of the server, or characteristics of a named resource.

GET and the *absolute* URL of the file requested is http://www.nytimes.com/. However HTTP/1.1 uses the *relative* URL which consists of the path only, in this case /. The HTTP version is 1.1.

The HTTP headers consist of a sequence of zero or more lines each consisting of an attribute name, followed by a colon, ":", and an attribute value. The client uses header lines to inform the server about: the type of the client, the kind of content it can accept, and the identity of the requester. In the example in Figure 2.24, the Accept header indicates a list of document format types that the client can accept. The Accept-language header indicates that U.S. English is accepted. The User-agent header indicates that the browser is Mozilla/4.0.

The current request methods available in HTTP/1.1 are given in Table 2.7. HTTP/1.0 only provides the methods GET, POST, and HEAD, and so these are the three methods that are supported widely.

The HTTP response message begins with an ASCII status line, followed by a headers section, and then by content, which is usually either an image or an HTML document. Figure 2.25 shows an example of a captured HTTP response message.

The response status line has the form: *HTTP-Version Status-Code Message* \r \n. The status code is a 3-digit number that indicates the result of the request to the client. The message indicates the result of the request in text that can be interpreted by humans. Examples of commonly encountered status lines are:

- HTTP/1.0 200 OK
- HTTP/1.0 301 Moved Permanently
- HTTP/1.0 400 Bad Request
- HTTP/1.0 500 Internal Server Error

The response headers provide information about the object that is being transferred to the client. Header lines are used to indicate: the type of server; the date and time the HTTP response was prepared and sent; and the time and date when the object was created or last modified. A Content-length header line indicates the length in bytes

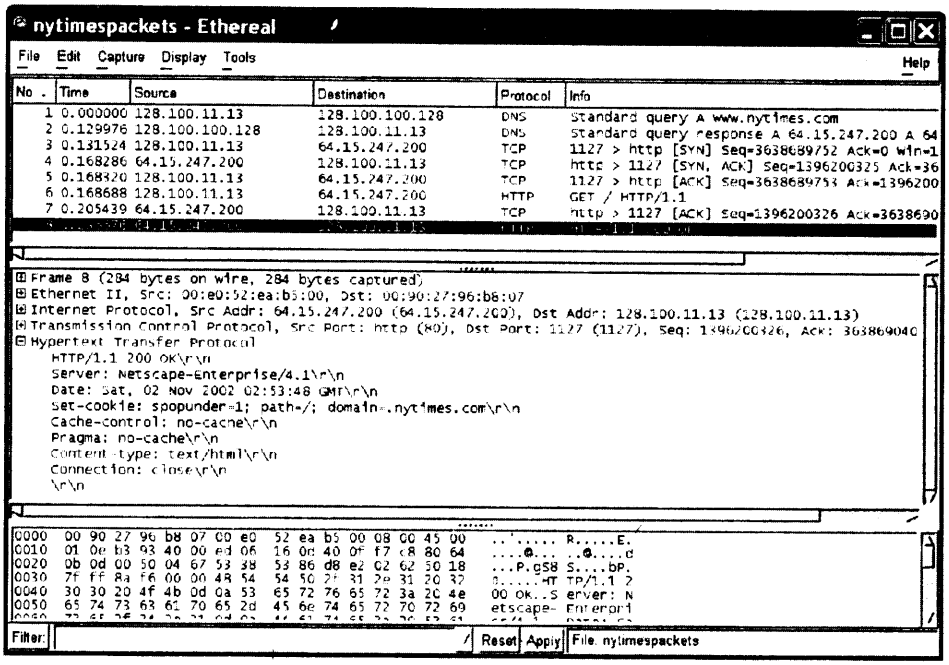


FIGURE 2.25 Ethereal capture showing HTTP response message.

of the object being transferred, and a Content-type header line indicates the type of the object (document) and how it is encoded. All these header lines are evident in the example in Figure 2.25. The response headers section ends with a blank line and may be followed by an entity body that carries the content.

HTTP PROXY SERVER AND CACHING

The simple-to-use graphical interface of web browsers made the web accessible to ordinary computer users and led to an explosion in the volume of traffic handled by the Internet. Web traffic is by far the largest component of all the traffic carried in the Internet. When the volume of requests for information from popular websites becomes sufficiently large, it makes sense to cache web information in servers closer to the user. By intercepting and responding to the HTTP request closer to the user, the volume of traffic that has to traverse the backbone of the Internet is reduced.

A *web proxy server* can be deployed to provide caching of web information. Typically proxy servers are deployed by Internet Service Providers to reduce the delay of web responses and to control the volume of web traffic. The user's browser must be configured to first access the proxy server when making a web request. If the proxy server has the desired object, it replies with an HTTP response. If it does not have the object, it sets up a TCP connection to the target URL and retrieves the desired object. It then replies to the client with the appropriate response, and caches the object for future requests.

COOKIES AND WEB SESSIONS

It was indicated that the HTTP protocol is stateless and does not maintain information about prior requests from a given client. The use of cookies makes it possible to have web sessions where a user interacts with a web site in a manner that takes into account the user's preferences. **Cookies** are data that are exchanged and stored by clients and servers and transferred as header lines in HTTP messages. These header lines provide context for each HTTP interaction.

When a client first accesses a web server that uses cookies, the server replies with Response message that includes a *Set-cookie* header line. This header line includes a unique ID number for the given client. If the client software accepts cookies, the cookie is added to the browser's cookie file. Each time the client makes a request to the given site, it includes a *Cookie* header line with the unique ID number in its requests. The server site maintains a separate cookie database where it can store which pages were accessed at what date and time by each client. In this manner, the server can prepare responses to HTTP requests that take into account the history of the given user. Cookies enable a website to keep track of a user's shopping cart during a session, as well as other longer term information such as address and credit card information. The example in Figure 2.24 can be seen to include a Cookie header line with an ID number that consists of 24 hexadecimal numerals.

Cookies are required to include an expiration date. Cookies that do not include an expiration date are deleted by the browser at the end of an interaction. Cookies are an indirect means for a user to identify itself to a server. If security and privacy are required, protocols such as SSL and TLS need to be used. These protocols are discussed in Chapter 12.

2.5.4 IP Utilities

A number of utilities are available to help in finding out about IP hosts and domains and to measure Internet performance. In this section we discuss PING, which can be used to determine whether a host is reachable; traceroute, a utility to determine the route that a packet will take to another host; netstat, which provides information about the network status of a local host; and tcpdump, which captures and observes packet exchanges in a link. We also discuss the use of Telnet with standard TCP/IP services as a troubleshooting and monitoring tool.

PING

PING is a fairly simple application used to determine whether a host is online and available. The name is said to derive from its analogous use in sonar operations to detect underwater objects.[12] PING makes use of Internet Control Message Protocol (ICMP) messages. The purpose of ICMP is to inform sending hosts about errors encountered in IP datagram processing or other control information by destination hosts or by routers. ICMP is discussed in Chapter 8. PING sends one or more ICMP Echo messages to a specified host requesting a reply. PING is often used to measure the round-trip delay

[12]PING is also reported to represent the acronym Packet Internet Groper [Murhammer 1998].

```
Microsoft(R) Windows DOS
(c)Copyright Microsoft Corp 1990-2001.

C:\DOCUME~1\1>ping nal.toronto.edu

Pinging nal.toronto.edu [128.100.244.3] with 32 bytes of data:

Reply from 128.100.244.3: bytes=32 time=84ms TTL=240
Reply from 128.100.244.3: bytes=32 time=110ms TTL=240
Reply from 128.100.244.3: bytes=32 time=81ms TTL=240
Reply from 128.100.244.3: bytes=32 time=79ms TTL=240

Ping statistics for 128.100.244.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 79ms, Maximum = 110ms, Average = 88ms

C:\DOCUME~1\1>_
```

FIGURE 2.26 Using PING to determine host accessibility.

between two hosts. The sender sends a datagram with a type 8 Echo message and a sequence number to detect a lost, reordered, or duplicated message. The receiver changes the type to Echo Reply (type 0) and returns the datagram. Because the TCP/IP suite incorporates ICMP, any machine with TCP/IP installed can reply to PING. However, because of the increased presence of security measures such as firewalls, the tool is not always successful. Nonetheless, it is still the first test used to determine accessibility of a host.

In Figure 2.26 PING is used to determine whether the NAL machine is available. In this example, the utility was run in an MS-DOS session under Windows XP. The command in its simplest form is `ping <hostname>`. The round-trip delay is indicated, as well as the time-to-live (TTL) value. The TTL is the maximum number of hops an IP packet is allowed to remain in the network. Each time an IP packet passes through a router, the TTL is decreased by 1. When the TTL reaches 0, the packet is discarded. See Chapter 8 for a PING and ICMP packet capture.

TELNET AND STANDARD SERVICES

Because ICMP operates at the IP level, PING tests the reachability of the IP layer only in the destination machine. PING does not test the layers above IP. A number of standard TCP/IP application layer services can be used to test the layers above IP. Telnet can be used to access these services for testing purposes. Examples of these services include Echo (port number 7), which echoes a character back to the sender, and Daytime (port number 13), which returns the time and date. A variety of utilities are becoming available for testing reachability and performance of HTTP and Web servers. The student is referred to the Cooperative Association for Internet Data Analysis (CAIDA) website, currently www.caida.org.

TRACEROUTE

A second TCP/IP utility that is commonly used is traceroute. This tool allows users to determine the route that a packet takes from the local host to a remote host, as well as latency and reachability from the source to each hop. Traceroute is generally used as a debugging tool by network managers.

```
Tracing route to www.comm.utoronto.ca [128.100.11.60]
over a maximum of 30 hops:

  1    1 ms  <10 ms  <10 ms  192.168.2.1
  2    3 ms    3 ms    3 ms  10.202.128.1
  3    4 ms    3 ms    3 ms  gw04.ym.phub.net.cable.rogers.com [66.185.83.142]
  4    *       *       *     Request timed out.
  5   47 ms   59 ms   66 ms  gw01.bloor.phub.net.cable.rogers.com [66.185.80.230]
  6    3 ms    3 ms   38 ms  gw02.bloor.phub.net.cable.rogers.com [66.185.80.242]
  7    8 ms    3 ms    5 ms  gw01.wlfdle.phub.net.cable.rogers.com [66.185.80.2]
  8    8 ms    7 ms    7 ms  gw02.wlfdle.phub.net.cable.rogers.com [66.185.80.142]
  9    4 ms   10 ms    4 ms  gw01.front.phub.net.cable.rogers.com [66.185.81.18]
 10    6 ms    4 ms    5 ms  ralsh-ge3-4.mt.bigpipeinc.com [66.244.223.237]
 11   16 ms   17 ms   13 ms  rx0sh-hydro-one-telecom.mt.bigpipeinc.com [66.244.223.246]
 12    7 ms   14 ms    8 ms  142.46.4.2
 13   10 ms    7 ms    6 ms  utorgw.onet.on.ca [206.248.221.6]
 14    7 ms    6 ms   11 ms  mcl-gateway.gw.utoronto.ca [128.100.96.101]
 15    7 ms    5 ms    8 ms  sf-gpb.gw.utoronto.ca [128.100.96.17]
 16    7 ms    7 ms   10 ms  bi15000.ece.utoronto.ca [128.100.96.236]
 17    7 ms    9 ms    9 ms  www.comm.utoronto.ca [128.100.11.60]

Trace complete.
```

FIGURE 2.27 Output from traceroute (running from a home PC to a host at the University of Toronto).

Traceroute makes use of both ICMP and UDP. The sender first sends a UDP datagram with TTL = 1 as well as an invalid port number to the specified destination host. The first router to see the datagram sets the TTL field to zero, discards the datagram, and sends an ICMP Time Exceeded message to the sender. This information allows the sender to identify the first machine in the route. Traceroute continues to identify the remaining machines between the source and destination machines by sending datagrams with successively larger TTL fields. When the datagram finally reaches its destination, that host machine returns an ICMP Port Unreachable message to the sender because of the invalid port number deliberately set in the datagram.

Figure 2.27 shows the result from running traceroute from a home PC to a host at the University of Toronto. The first line corresponds to the first hop in a home router. The next eight lines correspond to hops within the Internet Service Provider's network. The University of Toronto router gateway is reached in hop 13, and then various routers inside the university are traversed before arriving at the desired host.

IPCONFIG

The ipconfig utility, available on Microsoft® Windows operating systems, can be used to display the TCP/IP information about a host. In its simplest form the command returns the IP address, subnet mask (discussed in Chapter 8) and default gateway for the host. The utility can also be used to obtain information for each IP network interface for the host, for example, DNS hostname, IP addresses of DNS servers, physical address of the network card, IP address for the network interface, and whether DHCP is enabled for automatic configuration of the card's IP address. The ipconfig/renew command is used to renew an IP address with a DHCP server.

NETSTAT

The netstat queries a host about its TCP/IP network status. For example, netstat can be used to find the status of the network drivers and their interface cards, such as the

```
IPv4 Statistics

  Packets Received                    = 71271
  Received Header Errors              = 0
  Received Address Errors             = 9
  Datagrams Forwarded                 = 0
  Unknown Protocols Received          = 0
  Received Packets Discarded          = 0
  Received Packets Delivered          = 71271
  Output Requests                     = 70138
  Routing Discards                    = 0
  Discarded Output Packets            = 0
  Output Packet No Route              = 0
  Reassembly Required                 = 0
  Reassembly Successful               = 0
  Reassembly Failures                 = 0
  Datagrams Successfully Fragmented   = 0
  Datagrams Failing Fragmentation     = 0
  Fragments Created                   = 0

UDP Statistics for IPv4

  Datagrams Received    = 6810
  No Ports              = 15
  Receive Errors        = 0
  Datagrams Sent        = 6309

ICMPv4 Statistics

                            Received    Sent
  Messages                  10          6
  Errors                    0           0
  Destination Unreachable   8           1
  Time Exceeded             0           0
  Parameter Problems        0           0
  Source Quenches           0           0
  Redirects                 0           0
  Echos                     0           2
  Echo Replies              2           0
  Timestamps                0           0
  Timestamp Replies         0           0
  Address Masks             0           0
  Address Mask Replies      0           0

TCP Statistics for IPv4

  Active Opens                            = 798
  Passive Opens                           = 17
  Failed Connection Attempts              = 13
  Reset Connections                       = 467
  Current Connections                     = 0
  Segments Received                       = 64443
  Segments Sent                           = 63724
  Segments Retransmitted                  = 80
```

FIGURE 2.28 Sample protocol statistics output from netstat.

number of in packets, out packets, errored packets, and so on. It can also find out the state of the routing table in a host, which TCP/IP server processes are active in the host, as well as which TCP connections are active. Figure 2.28 shows the result from running netstat with the protocol statistics option. Various counts for IP, ICMP, TCP, and UDP are displayed.

2.5.5 Tcpdump and Network Protocol Analyzers

The tcpdump program can capture and observe IP packet exchanges on a network interface. The program usually involves setting an Ethernet network interface card into a "promiscuous" mode so that the card listens and captures every frame that traverses the Ethernet broadcast network. A packet filter is used to select the IP packets that are of interest in a given situation. These IP packets and their higher-layer contents can then be observed and analyzed. Because of security concern, normal users typically cannot run the tcpdump program.

The tcpdump utility can be viewed as an early form of a protocol analyzer. A *network protocol analyzer* is a tool for capturing, displaying, and analyzing the PDUs that are exchanged in a network. Current analyzers cover a very broad range of protocols and are constantly being updated. Protocol analyzers are indispensable in troubleshooting network problems and in designing new network systems. Protocol analyzers are also extremely useful in teaching the operation of protocols by providing a means of examining traffic from a live network.

The first component for a protocol analyzer is hardware to capture the digital information from the physical medium. The most cost-effective means for capturing

information is to use a LAN network interface card. Most LANs support operation in promiscuous mode where all frames on the LAN are captured for examination. Note that in most LAN protocols the frames can be seen by all devices attached to the medium, even if the frame is not intended for them. Since most computers are connected to Ethernet LANs, packet capture can be done readily by installing device driver software to control the network interface card.

Given the increasingly high speeds of LAN operation, the volume of information that can be captured can quickly become huge. The second component of a protocol analyzer is filtering software to select the frames that contain the desired information. Filtering can be done by frame address, by IP address, by protocol, and by many other combinations. The final component of a protocol analyzer consists of the utilities for the display and analysis of protocol exchanges. A number of commercial and open source network protocol analyzers packages are available. In this book we will use the Ethereal open source package. Several hundred developers have contributed to the development of Ethereal leading to a tool that supports an extensive set of protocols. The Ethereal package can be downloaded from www.ethereal.com. Their website also contains instructions and example screen captures.

Network protocol analyzers give the ability to capture all packets in a LAN and in doing so provide an opportunity to gain unauthorized access to network information. These tools should *always* be used in a responsible and ethical manner.

SUMMARY

This chapter describes how network architectures are based on the notion of layering. Layering involves combining network functions into groups that can be implemented together. Each layer provides a set of services to the layer above it; each layer builds its services using the services of the layer below. Thus applications are developed using application layer protocols, and application layer protocols are built on top of the communication services provided by TCP and UDP. These transport protocols in turn build on the datagram service provided by IP, which is designed to operate over various network technologies. IP allows the applications above it to be developed independently of specific underlying network technologies. The network technologies below IP range from full-fledged packet-switching networks, such as ATM, to LANs, and individual point-to-point links.

The Berkeley socket API allows the programmer to develop applications using the services provided by TCP and UDP. Examples of applications that run over TCP are HTTP, FTP, SMTP and Telnet. DNS and RTP are examples that run over UDP. The power of the TCP/IP architecture is that any new application that runs over TCP or UDP will run over the entire global Internet. Consequently, new services and applications can be deployed globally very quickly, a capability that no other network architecture can provide. We also introduced various TCP/IP utilities and tools that allow the programmer to determine the state and configuration of a TCP/IP network. Students can use these tools to get some hands on experience with the operation of TCP/IP.

CHECKLIST OF IMPORTANT TERMS

application layer
blocking/unblocking
client/server
confirmed/unconfirmed service
connectionless service
connection-oriented service
cookie
daemon
data link layer
datagram
Domain Name System (DNS)
encapsulation
ephemeral port number
frame
globally unique IP address
header
HyperText Markup Language (HTML)
Hypertext Transfer Protocol (HTTP)
internet layer
internetworking
layer
layer n entity
layer n protocol
multiplexing/demultiplexing
network architecture
network interface layer
network layer
OSI reference model
packet
peer process
physical address
physical layer
Point-to-Point Protocol (PPP)
port
Post Office Protocol version 3 (POP3)
presentation layer
protocol
protocol data unit (PDU)
segment
segmentation and reassembly
service access point (SAP)
service data unit (SDU)
session layer
Simple Mail Transfer Protocol (SMTP)
socket
socket address
splitting/recombining
TCP/IP network architecture
Transmission Control Protocol (TCP)
transport layer
User Datagram Protocol (UDP)
well-known port number

FURTHER READING

Comer, D. E. and D. L. Stevens, *Internetworking with TCP/IP, Vol. III: Client-Server Programming and Applications,* Prentice Hall, Englewood Cliffs, New Jersey, 1993.

Murhammer, M. W., O. Atakan, S. Bretz, L. R. Pugh, K. Suzuki, and D. H. Wood, *TCP/IP Tutorial and Technical Overview,* Prentice Hall PTR, Upper Saddle River, New Jersey, 1998.

Perlman, R., *Interconnections: Bridges, Routers Switches and Internet Protocols,* Addison-Wesley, Reading, Massachusetts, 2000.

Piscitello, D. M. and A. L. Chapin, *Open Systems Networking: TCP/IP and OSI,* Addison-Wesley, Reading, Massachusetts, 1993.

Sechrest, S., "An Introductory 4.4 BSD Interprocess Communication Tutorial," *Computer Science Network Group,* UC Berkeley.

Stevens, W. R., *TCP/IP Illustrated, Volume 1: The Protocols,* Addison-Wesley, Reading, Massachusetts, 1994.

Stevens, W. R., *UNIX Network Programming,* Volume 1, 2nd edition, Prentice Hall, Englewood Cliffs, New Jersey, 1998. An excellent treatment of socket programming.

Yeager, N. J. and R. E. McGrath, *Web Server Technology: The Advanced Guide for World Wide Web Information Providers,* Morgan Kaufmann, San Francisco, 1996.
RFC 821, J. Postel, "Simple Mail Transfer Protocol," August 1982.
RFC 854, J. Postel and J. Reynolds, "Telnet Protocol Specification," May 1983.
RFC 959, J. Postel and J. Reynolds, "File Transfer Protocol," October 1985.
RFC 1034, Mockapetris, "Domain Names—Concepts and Facilities," November 1987.
RFC 1035, Mockapetris, "Domain Names—Implementation and Specification," November 1987.
RFC 1945, T. Berners-Lee, R. Fielding, and H. Frystik, "Hypertext Transfer Protocol/1.0," May 1996.
RFC 2068, R. Fielding, J. Geetys, J. Mogul, H. Frystyk, T. Berners-Lee, "Hypertext Transfer Protocol," January 1997.
RFC 2151, G. Kessler and S. Shepard, "Internet & TCP/IP Tools and Utilities," June 1997.
RFC 2616, R. Fielding, J. Geetys, J. Mogul, H. Frystyk, L. Masinder, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol/1.1," June 1999.
RFC 2821, J. Klensin, ed., "Simple Mail Transfer Protocol," April 2001.
RFC 3000, J. Reynolds, R. Braden, S. Ginoza, and L. Shiota, eds., "Internet Official Protocol Standards," November 2001.

See our website for additional references available through the Internet.

PROBLEMS

2.1. Explain how the notion of layering and internetworking make the rapid growth of applications such as the World Wide Web possible.

2.2. (a) What universal set of communication services is provided by TCP/IP?
(b) How is independence from underlying network technologies achieved?
(c) What economies of scale result from (a) and (b)?

2.3. What difference does it make to the network layer if the underlying data link layer provides a connection-oriented service versus a connectionless service?

2.4. Suppose transmission channels become virtually error free. Is the data link layer still needed?

2.5. Why is the transport layer not present inside the network?

2.6. Which OSI layer is responsible for the following?
(a) Determining the best path to route packets.
(b) Providing end-to-end communications with reliable service.
(c) Providing node-to-node communications with reliable service.

2.7. Should connection establishment be a confirmed service or an unconfirmed service? What about data transfer in a connection-oriented service? Connection release?

2.8. Does it make sense for a network to provide a confirmed, connectionless packet transfer service?

2.9. Explain how the notion of multiplexing can be applied at the data link, network, and transport layers. Draw a figure that shows the flow of PDUs in each multiplexing scheme.

2.10. Give two features that the data link layer and transport layer have in common. Give two features in which they differ. Hint: Compare what can go wrong to the PDUs that are handled by these layers.

2.11. (a) Can a connection-oriented, reliable message transfer service be provided across a connectionless packet network? Explain.
(b) Can a connectionless datagram transfer service be provided across a connection-oriented network?

2.12. An internet path between two hosts involves a hop across network A, a packet-switching network, to a router and then another hop across packet-switching network B. Suppose that packet-switching network A carries the packet between the first host and the router over a two-hop path involving one intermediate packet switch. Suppose also that the second network is an Ethernet LAN. Sketch the sequence of IP and non-IP packets and frames that are generated as an IP packet goes from host 1 to host 2.

2.13. Does Ethernet provide connection-oriented or connectionless service?

2.14. Ethernet is a LAN so it is placed in the data link layer of the OSI reference model.
(a) How is the transfer of frames in Ethernet similar to the transfer of frames across a wire? How is it different?
(b) How is the transfer of frames in Ethernet similar to the transfer of frames in a packet-switching network? How is it different?

2.15. Suppose that a group of workstations is connected to an Ethernet LAN. If the workstations communicate only with each other, does it make sense to use IP in the workstations? Should the workstations run TCP directly over Ethernet? How is addressing handled?

2.16. Suppose two Ethernet LANs are interconnected by a box that operates as follows. The box has a table that tells it the physical addresses of the machines in each LAN. The box listens to frame transmissions on each LAN. If a frame is destined to a station at the other LAN, the box retransmits the frame onto the other LAN; otherwise, the box does nothing.
(a) Is the resulting network still a LAN? Does it belong in the data link layer or the network layer?
(b) Can the approach be extended to connect more than two LANs? If so, what problems arise as the number of LANs becomes large?

2.17. Suppose all laptops in a large city are to communicate using radio transmissions from a high antenna tower. Is the data link layer or network layer more appropriate for this situation? Now suppose the city is covered by a large number of small antennas covering smaller areas. Which layer is more appropriate?

2.18. Suppose that a host is connected to a connection-oriented packet-switching network and that it transmits a packet to a server along a path that traverses two packet switches. Suppose that each hop in the path involves a point-to-point link, that is, a wire. Show the sequence of network layer and data link layer PDUs that is generated as the packet travels from the host to the server.

2.19. Suppose an application layer entity wants to send an L-byte message to its peer process, using an existing TCP connection. The TCP segment consists of the message plus 20 bytes of header. The segment is encapsulated into an IP packet that has an additional 20 bytes

of header. The IP packet in turn goes inside an Ethernet frame that has 18 bytes of header and trailer. What percentage of the transmitted bits in the physical layer corresponds to message information if $L = 100$ bytes? 500 bytes? 1000 bytes?

2.20. Suppose that the TCP entity receives a 1.5-megabyte file from the application layer and that the IP layer is willing to carry blocks of maximum size 1500 bytes. Calculate the amount of overhead incurred from segmenting the file into packet-sized units.

2.21. Suppose a TCP entity receives a digital voice stream from the application layer. The voice stream arrives at a rate of 8000 bytes/second. Suppose that TCP arranges bytes into block sizes that result in a total TCP and IP header overhead of 50 percent. How much delay is incurred by the first byte in each block?

2.22. How does the network layer in a connection-oriented packet-switching network differ from the network layer in a connectionless packet-switching network?

2.23. Identify session layer and presentation layer functions in the HTTP protocol.

2.24. Suppose we need a communication service to transmit real-time voice over the Internet. What features of TCP and what features of UDP are appropriate?

2.25. Consider the end-to-end IP packet transfer examples in Figure 2.15. Sketch the sequences of IP packets and Ethernet and PPP frames that are generated by the three examples of packet transfers: from the workstation to the server, from the server to the PC, and from the PC to the server. Include all relevant header information in the sketch.

2.26. Suppose a user has two browser applications active at the same time and suppose that the two applications are accessing the same server to retrieve HTTP documents at the same time. How does the server tell the difference between the two applications?

2.27. Consider the operation of nonpersistent HTTP and persistent HTTP.

(a) In nonpersistent HTTP (version 1.0): Each client/server interaction involves setting up a TCP connection, carrying out the HTTP exchange, and closing the TCP connection. Let T be the time that elapses from when a packet is sent from client to server to when the response is received. Find the rate at which HTTP exchanges can be made using nonpersistent HTTP.

(b) In persistent HTTP (version 1.1) the TCP connection is kept alive. Find the rate at which HTTP exchanges can be made if the client cannot send an additional request until it receives a response for each request.

(c) Repeat part (b) if the client is allowed to pipeline requests, that is, it does not have to wait for a response before sending a new request.

2.28. What is the difference between a physical address, a network address, and a domain name?

2.29. Explain how a DNS query proceeds if the local name server does not have the IP address for a given host when the following approaches are used. Assume an example where four machines are involved in ultimately resolving a given query.

(a) When a machine B cannot resolve an address in response to a query from A, machine B sends the query to another machine in the chain. When B receives the response, it forwards the result to B.

(b) When a machine B cannot resolve an address in response to a query from A, machine B sends a DNS reply to A with the IP address of the next machine in the chain, and machine A contacts that machine.

2.30. Suppose that the DNS system used a single centralized database handle all queries. Compare this centralized approach to the distributed approach in terms of reliability, throughput (volume of queries/second that can be processed), query response delay, and maintainability.

2.31. What is wrong with the following methods of assigning host id addresses?
(a) Copy the address from the machine in the next office.
(b) Modify the address from the machine in the next office.
(c) Use an example from the vendor's brochure.

2.32. Suppose a machine is attached to several physical networks. Why does it need a different IP address for each attachment?

2.33. Suppose a computer is moved from one department to another. Does the physical address need to change? Does the IP address need to change? Does it make a difference if the computer is a laptop?

2.34. Suppose the population of the world is 6 billion people and that there is an average of 1000 communicating devices per person. How many bits are required to assign a unique host address to each communicating device? Suppose that each device attaches to a single network and that each network on average has 10,000 devices. How many bits are required to provide unique network ids to each network?

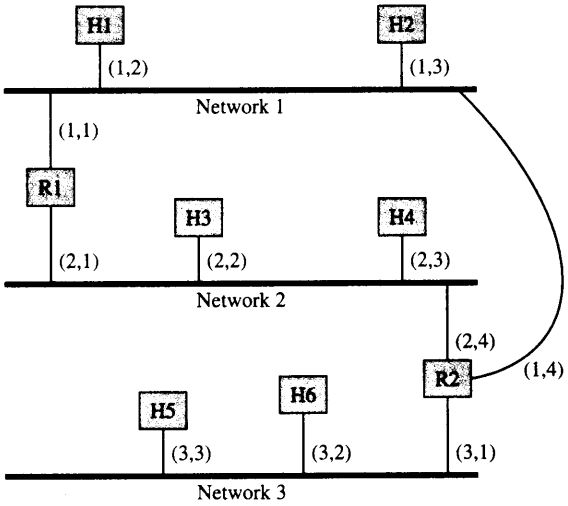
2.35. Can the Internet protocol be used to run a homogeneous packet-switching network, that is, a network with identical packet switches interconnected with point-to-point links?

2.36. Is it possible to build a homogeneous packet-switching network with Ethernet LANs interconnecting the packet switches? If so, can connection-oriented service be provided over such a network?

2.37. In telephone networks one basic network is used to provide worldwide communications. In the Internet a multiplicity of networks are interconnected to provide global connectivity. Compare these two approaches, namely, a single network versus an internetwork, in terms of the range of services that can be provided and the cost of establishing a worldwide network.

2.38. Consider an internetwork architecture that is defined using gateways/routers to communicate across networks but that uses a connection-oriented approach to packet switching? What functionality is required in the routers? Are any additional constraints imposed on the underlying networks?

2.39. The internet below consists of three LANs interconnected by two routers. Assume that the hosts and routers have the IP addresses as shown.
(a) Suppose that all traffic from network 3 that is destined to H1 is to be routed directly through router R2 and that all other traffic from network 3 is to go to network 2. What routing table entries should be present in the network 3 hosts and in R2?



(b) Suppose that all traffic from network 1 to network 3 is to be routed directly through R2. What routing table entries should be present in the network 1 hosts and in R2?

2.40. Explain why it is useful for application layer programs to have a "well-known" TCP port number?

2.41. Use a Web browser to connect to cnn.com. Explain what layers in the protocol stack are involved in the delivery of the video newscast.

2.42. Use a Web browser to connect to an audio program, say, www.rollingstone.com/radio/ (Rolling Stone Radio) or www.cbc.ca (CBC Radio). Explain what layers in the protocol stack are involved here. How does this situation differ from the delivery of video in problem 2.41?

2.43. Which of the TCP/IP transport protocol (UDP or TCP) would you select for the following applications: packet voice, file transfers, remote login, multicast communication (i.e., multiple destinations).

2.44. (a) Use the Telnet program to send an e-mail by directly interacting with your local mail server. The SMTP server has port 25. You can find the list of commands for the SMTP protocol in RFC 2821, which can be downloaded from www.ietf.org.

(b) Use Ethereal to capture and analyze the sequence of messages exchanged. Identify the various types of addresses for Ethernet, IP, and TCP PDUs. Examine the data in the Telnet messages to determine whether the login name and password are encrypted.

2.45. (a) Use the Telnet program to retrieve e-mail from your local mail server. The POP3 server has port 110. You can find the list of commands for the POP3 protocol in RFC 1939, which can be downloaded from www.ietf.org.

(b) Repeat Problem 2.44(b).

2.46. The `nslookup` program can be used to query the Internet domain name servers. Use this program to look up the IP address of www.utoronto.ca.

2.47. (a) Use PING to find the round-trip time to the home page of your university and to the home page of your department.
(b) Use Ethereal to capture the ICMP packets exchanged. Correlate the information in the packet capture with the information displayed by the PING result.

2.48. (a) Use netstat to find out the routing table for a host in your network.
(b) Use netstat to find the IP statistics for your host.

2.49. Suppose regularly spaced PING packets are sent to a remote host. What can you conclude from the following results?
(a) No replies arrive back.
(b) Some replies are lost.
(c) All replies arrive but with variable delays.
(d) What kind of statistics would be useful to calculate for the round-trip delays?

2.50. Suppose you want to test the response time of a specific web server. What attributes would such a measurement tool have? How would such a tool be designed?

2.51. A denial-of-service attack involves loading a network resource to the point where it becomes nonfunctional.
(a) Explain how PING can be used to carry out a denial-of-service attack.
(b) On October 21, 2002, the 13 DNS root servers were subject to a distributed denial-of-service attack. Explain the impact of the attack on the operation of the Internet if some of the servers are brought down; if all of the servers are brought down.

2.52. (a) Use a web browser to retrieve a file from a local web server.
(b) HTTP relies on ASCII characters. To verify the sequence of messages shown in Table 2.1, use the Telnet program to retrieve the same file from the local website.

2.53. Use Ethereal to capture the sequence of PDUs exchanged in problem 2.52 parts (a) and (b).
(a) Identify the Ethernet, IP, and TCP addresses of the machines involved in the exchange.
(b) Are there any DNS queries?
(c) Identify the TCP connection setup.
(d) Examine the contents of the HTTP GET and response messages.
(e) Examine how the TCP sequence numbers evolve over time.

2.54. Discuss the similarities and differences between the control connection in FTP and the remote control used to control a television. Can the FTP approach be used to provide VCR-type functionality to control the video from a video-on-demand service?

2.55. Use a Web browser to access the CAIDA Web page (http://www.caida.org/tools/taxonomy/) to retrieve the CAIDA measurement tool taxonomy document. You will find links there to many free Internet measurement tools and utilities.

2.56. Use traceroute to determine the path from your home PC to your university's main web page, while capturing the packets using Ethereal.
(a) Using the output from traceroute, try to identify how many different networks and service providers are traversed.
(b) Verify the operation of traceroute by examining the contents of the packets.

2.57. Run the UDP client and server programs from the Berkeley API section on different machines, record the round-trip delays with respect to the size of the data, and plot the results.

2.58. In the TCP example from the Berkeley API section, the message size communicated is fixed regardless of how many characters of actual information a user types. Even if the user wants to send only one character, the programs still sends 256 bytes of messages—clearly an inefficient method. One possible way to allow variable-length messages to be communicated is to indicate the end of a message by a unique character, called the sentinel. The receiver calls `read` for every character (or byte), compares each character with the sentinel value, and terminates after this special value is encountered. Modify the TCP client and server programs to handle variable-length messages using a sentinel value.

2.59. Another possible way to allow variable-length messages to be communicated is to precede the data to be transmitted by a header indicating the length of the data. After the header is decoded, the receiver knows how many more bytes it should read. Assuming the length of the header is two bytes, modify the TCP client and server programs to handle variable-length messages.

2.60. The UDP client program in the example from the Berkeley API section may wait forever if the datagram from the server never arrives. Modify the client program so that if the response from the server does not arrive after a certain timeout (say, 5 seconds), the `read` call is interrupted. The client then retransmits a datagram to the server and waits for a new response. If the client does not receive a response after a fixed number of trials (say, 10 trials), the client should print an error message and abandon the program. Hint: Use the `sigaction` and `alarm` functions.

2.61. Modify the UDP client to access a date-and-time server in a host local to your network. A date-and-time server provides client programs with the current day and time on demand. The system internal clock keeps the current day and time as a 32-bit integer. The time is incremented by the system (every second). When an application program (the server in this case) asks for the date or time, the system consults the internal clock and formats the date and time of day in human-readable format. Sending any datagram to a date-and-time server is equivalent to making a request for the current date and time; the server responds by returning a UDP message containing the current date and time. The date-and-time server can be accessed in UDP port 13.

2.62. Write a file transfer application that runs over UDP. Assume the transfer occurs over a local area network so reliable transfer is not a concern. Assume also that UDP will accept at most 500 bytes/datagram. Implement a server that opens a socket and listens for incoming data at a particular port number. Implement a client that reads a file from the file system and transfers the file to the server. When the server receives the client's data, it writes this data to a file. Include a means for the client to indicate the end of transmission.

CHAPTER 3

Digital Transmission Fundamentals

Modern communication networks based on digital transmission systems have the potential to carry all types of information and hence to support many types of applications. We saw in Chapter 1 that the design of the early network architectures was tailored to very specific applications that led to the development of corresponding transmission systems. Telegraphy was developed specifically for the transfer of text messages. Morse and Baudot pioneered the use of binary representations for the transfer of text and developed digital transmission systems for the transfer of the resulting binary information. Later telephony was developed for the transfer of voice information. Initially the voice information was transmitted using analog transmission systems. The invention of pulse code modulation (PCM) enabled voice to be transmitted over digital transmission networks. In the same way that the Morse and Baudot codes standardized the transfer of text, PCM standardized the transfer of voice in terms of 0s and 1s. We are currently undergoing another major transition from analog to digital transmission technology, namely, the transition from analog television systems to entirely digital television systems. When this transition is complete, all major forms of information will be represented in digital form. This change will open the way for the deployment of digital transmission networks that can transfer the information for all the major types of information services.

In this chapter we present the fundamental concepts concerning digital transmission. These concepts form the basis for the design of the digital transmission systems that constitute the physical layer of modern network architectures. The chapter is organized into the following sections:

1. *Digital representation of information.* We consider different types of information and their representation in digital form. Text, image, voice, audio, and video are used as examples.
2. *Why digital transmission?* We explain the advantages of digital transmission over analog transmission. We present the key parameters that determine the transmission

capacity of a physical medium. We also indicate where various media are used in digital transmission systems. This section is a summary of the following three sections.[1]

3. *Digital representation of analog signals.* We explain how PCM is used to convert an analog signal into a binary information stream. Voice and audio signals are used as examples.
4. *Characterization of communication channels.* We discuss communication channels in terms of their ability to transmit pulse and sinusoidal signals. We introduce the concept of bandwidth as a measure of a channel's ability to transmit pulse information.
5. *Fundamental limits of digital transmission.* We discuss binary and multilevel digital transmission systems, and we develop fundamental limits on the bit rate that can be obtained over a channel.
6. *Line coding.* We introduce various signal formats for transmitting binary information and discuss the criteria for selecting an appropriate line code.
7. *Modems and digital modulation.* We discuss digital transmission systems that use sinusoidal signals, and we explain existing telephone modem standards.
8. *Properties of transmission media.* We discuss copper wire, radio, and optical fiber systems and their role in access and backbone digital networks. Examples from various physical layer standards are provided.
9. *Error detection and correction.* We present coding techniques that can be used to detect and correct errors that may occur during digital transmission. These coding techniques form the basis for protocols that provide reliable transfer of information. Protocols that use error detection and correction are found in the physical, data link, network, and transport layers.

3.1 DIGITAL REPRESENTATION OF INFORMATION

Applications that run over networks involve the transfer of information of various types. Some applications involve the transfer of blocks of text characters, e-mail, for example. Other applications involve the transfer of a stream of information, such as telephony. In the case of text, the information is already in digital form. In the case of voice, the information is analog in nature and must be converted into digital form. This section focuses on the number of bits required to represent various types of information, for example, text, speech, audio, data, images, and video. In the case of block-oriented information, we are interested in the number of bits required to represent a block. In the case of stream-oriented information, we are interested in the *bit rate* (number of bits/second) required to represent the information.

It is useful to identify which layers in the OSI reference model we are dealing with in this section. In general, the information associated with an application is generated above the application layer. The blocks or streams of information generated by the application must be handled by all the lower layers in the protocol stack. Ultimately,

[1]This arrangement allows Sections 3.3 to 3.5 to be skipped in courses that are under tight time constraints.

the physical layer must carry out the transfer of all the bits generated by the application and the layers below. In a sense, the application generates flows of information that need to be carried across the network; the digital transmission systems at the physical layer provide the pipes that actually carry the information flows across the network. The purpose of this section, then, is to introduce the important examples of information types, such as text, voice, audio, and video, so that we can relate their requirements to the bit rates provided by transmission systems.

3.1.1 Block-Oriented Information

Information can be grouped into two broad categories: information that occurs naturally in the form of a single *block* and *stream information* that is produced continuously and that needs to be transmitted as it is produced. Table 3.1 gives examples of *block-oriented information*, which include data files, black-and-white documents, and pictures.

The most common examples of block information are files that contain text, numerical, or graphical information. We routinely deal with these types of information when we send e-mail and when we retrieve documents. These blocks of information can range from a few bytes to several hundred kilobytes and occasionally several megabytes. The normal form in which these files occur can contain a fair amount of statistical redundancy. For example, in English text certain characters and patterns such as *e* and *the*, occur very frequently. *Data compression* utilities such as compress, zip, and other variations exploit these redundancies to encode the original information into files that require fewer bits to transfer and less disk storage space.[2] Some modem standards also apply these data compression schemes to the information prior to transmission. The *compression ratio* is defined as the ratio of the number of bits in the original file to the number of bits in the compressed file. Typically the compression ratio for these types of information is two or more, thus providing an apparent doubling or more of the transmission speed or storage capacity.

[2]The details of the data compression techniques discussed in this section are found in Chapter 12.

TABLE 3.1 Block-oriented information.

Information type	Data compression technique	Format	Uncompressed	Compressed (compression ratio)	Applications
Text files	Compress, zip, and variations	ASCII	kbytes to Mbytes	(2–6)	Disk storage, file transfer
Scanned black-and-white documents	CCITT Group 3 facsimile standard	A4 page at 200 × 100 pixels/inch and options	256 kbytes	15–54 kbytes (1-D) 5–35 kbytes (2-D) (5–50)	Facsimile transmission, document storage
Color images	JPEG	8 × 10 inch photo scanned at 400 pixels/inch	38.4 Mbytes	1.2–8 Mbytes (5–30)	Image storage or transmission

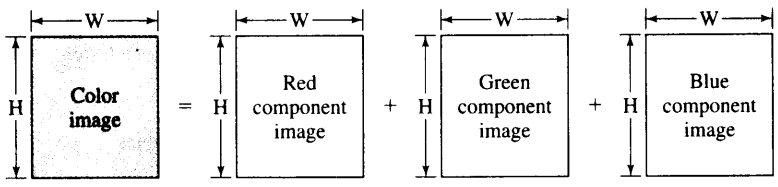
Certain applications require that a block of information be delivered within a certain maximum **delay**. The time to deliver a block of L bits of information over a transmission system of R bits/second consists of the propagation delay and the block transmission time: delay $= t_{prop} + L/R$. The propagation delay $t_{prop} = d/v$ where d is the distance that the information has to travel and v is the speed of light in the transmission medium. Clearly, the designer cannot change the speed of light, but the distance that the information has to travel can be controlled through the placement of the file servers. The time to transmit the file can be reduced by increasing the transmission bit rate R. In the following discussion we consider several examples where delay and bit rate are traded off.

A facsimile document system scans a black-and-white document into an array of dots that are either white or black. A *pixel* is defined as a single dot in a digitized image. The CCITT Group 3 facsimile standards provide for resolutions of 200, 300, or 400 dots per horizontal inch and 100, 200, or 400 vertical dots per inch. For example, a standard A4 page at 200 × 100 pixels/inch (slightly bigger than 8.5 × 11 inches) produces 256 kilobytes prior to compression. At a speed of 28.8 kbps, such an uncompressed page would require more than 1 minute to transmit. Existing fax compression algorithms can reduce this transmission time typically by a factor of 8 to 16.

An individual color image produces a huge number of bits. For example, an 8 × 10-inch picture scanned at a resolution of 400 × 400 pixels per square inch yields 400 × 400 × 8 × 10 = 12.8 million pixels; see Table 3.1. A color image is decomposed into red, green, and blue subimages as shown in Figure 3.1. Normally eight bits are used to represent each of the red, green, and blue color components, resulting in a total of 12.8 megapixels × 3 bytes/pixel = 38.4 megabytes. At a speed of 28.8 kbps, this image would require about 3 hours to transmit! Clearly, data compression methods are required to reduce these transmission times.

The *Graphics Interchange Format (GIF)* takes image data, in binary form, and applies lossless data compression. *Lossless data compression* schemes produce a compressed file from which the original data can be recovered *exactly*. (Facsimile and file compression utilities also use lossless data compression.) However, lossless data compression schemes are limited in the compression rates they can achieve. For this reason, GIF is used mainly for simple images such as line drawings and images containing simple geometrical shapes.

Lossy data compression schemes produce a compressed file from which only an *approximation* to the original information can be recovered. Much higher compression ratios are possible. In the case of images, lossy compression is acceptable as long as there is little or no visible degradation in image quality. The *Joint Photographic Experts*



Total bits before compression = 3 × H × W pixels × B bits/pixel = 3 HWB

FIGURE 3.1 The three components of a color image.

TABLE 3.2 Properties of audio and video stream information.

Information type	Compression technique	Format	Uncompressed	Compressed	Applications
Voice	PCM	4 kHz voice	64 kbps	n/a	Digital telephony
Voice	ADPCM (+ silence detection)	4 kHz voice	64 kbps	16–32 kbps	Digital telephony, voice mail
Voice	Residual-excited linear prediction	4 kHz voice	64 kbps	8–16 kbps	Digital cellular telephony
Audio	MPEG audio MP3 compression	16–24 kHz audio	512–748 kbps	32–384 kbps	MPEG audio
Video	H.261 coding	176 × 144 or 352 × 288 frames at 10–30 frames/second	2–36.5 Mbps	64 kbps–1.544 Mbps	Video conferencing
Video	MPEG-2	720 × 480 frames at 30 frames/second	249 Mbps	2–6 Mbps	Full-motion broadcast video, DVD
Video	MPEG-2	1920 × 1080 frames at 30 frames/second	1.6 Gbps	19–38 Mbps	High-definition television

Group (JPEG) standard provides a lossy compression algorithm that can be adjusted to balance image quality versus file size.[3] The compression ratio that is achieved for a given image depends on the degree of detail and busyness of the content. Images that contain a few large, smooth objects are highly compressible, as, for example, in a chart containing a few large circles with uniform color in each picture. Images that contain large numbers of small objects, for example, a picture of the fans in the stands in a stadium, will be much less compressible. As an example, JPEG can typically produce a high-quality reproduction with a compression ratio of about 15. Combined with the fastest telephone modems, say 56 kbps, this compression ratio reduces the transmission time of the image in Table 3.1 to several minutes. Clearly in the case of images, we either make do with lower resolution and/or lower quality images or we procure higher speed communications.

3.1.2 Stream Information

Information such as voice, music, or video is produced in a steady stream. Table 3.2 lists the properties of this type of information. In the case of a voice or music signal, the sound, which consists of variations in air pressure, is converted into a voltage that

[3]JPEG is discussed in Chapter 12.

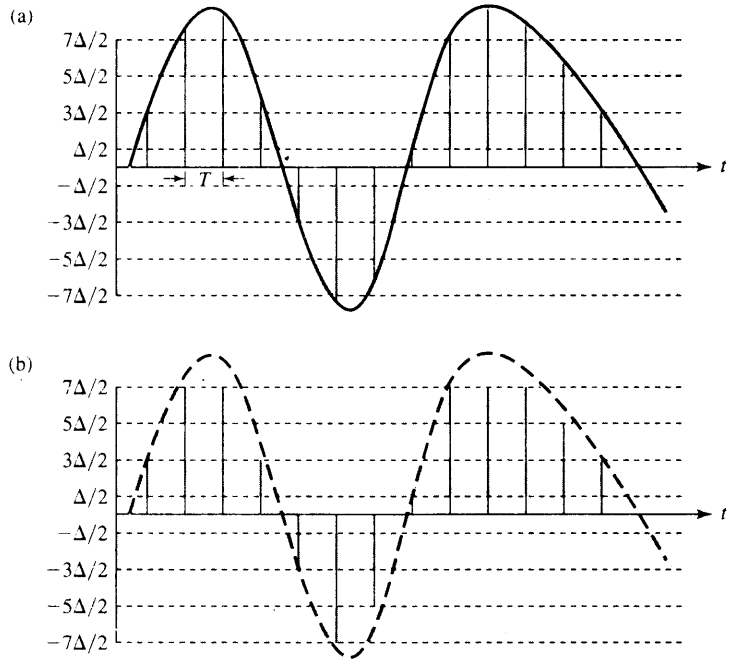


FIGURE 3.2 Sampling of a speech signal: (a) original waveform and the sample values; (b) original waveform and the quantized values.

varies continuously with time assuming values over a continuous range. We refer to these signals as *analog* signals.

The first step in digitizing an analog signal is to obtain sample values of the signal every T seconds as shown in Figure 3.2a. Clearly, the value of T between samples depends on how fast the signal varies with time. The **bandwidth of a signal** is a measure of how fast the signal varies. Bandwidth is measured in cycles/second or Hertz. A basic result from signal theory is that if a signal has bandwidth W then the minimum sampling rate is $2W$ samples/second. For example, for a **pulse code modulation (PCM)** telephone-quality voice, the signal has a bandwidth of 4 kHz and so the signal is sampled at a rate of 8000 samples/second, that is, $T = 1/8000 = 125$ microseconds as shown in Figure 3.2a.[4]

The second step in digitizing a signal involves *quantizing* each of the sample values. Figure 3.2b shows the operation of a quantizer: In this example, each of the signal samples is approximated by one of eight levels. Each level can then be represented by a three-bit number. Clearly, the accuracy of the reproduced signal increases as the number of bits used to represent each sample is increased. In the case of telephone systems, the PCM voice samples are represented by 8 bits in resolution, resulting in a bit rate for PCM of 8000 samples/second $\times$ 8 bits/sample = 64 kbps.

[4]PCM is discussed in Section 3.3.

Many applications involve information that is produced continuously and that needs to be transferred with small delay. For example, communications between people is real-time and requires a maximum delay of about 250 ms to ensure interactivity close to that of normal conversation. Suppose that an information source produces information at a rate of R_s bps. Suppose that the digital transmission system transfers information at a rate of R bps. To attain real-time communications it is then necessary that the transmission rate R be greater than or equal to the rate R_s at which the source produces information. If the source produces information faster than the transmission system can transfer it, then a backlog of information will build up at the input to the transmission system. For example, in the telephone network the digitized voice signal has a bit rate of $R_s = 64$ kbps and the network provides transmission channels of bit rate $R = 64$ kbps. If the real-time requirement for the transfer of information is removed, then the binary encoded stream produced by a signal such as audio or video can be stored and sent as a block. In this sense, the distinction between block and stream information depends on the requirements of the situation.

The high cost of transmission in certain situations, for example, cellular radio systems, has led to the development of more complex algorithms for reducing the bit rate while maintaining the quality of a voice signal that one encounters in conventional networks. Differential PCM (DPCM) encodes the difference between successive samples of the voice signal. Adaptive DPCM (ADPCM) adapts to variations in voice signal level, that is, the loudness of the signal. Linear predictive methods adapt to the type of sound, for example, *ee* versus *ss*. These systems can reduce the bit rate of telephone quality voice to the range 8 to 32 kbps. Despite the fact that they are "lossy," these schemes achieve compression and high quality due to the imperceptibility of the approximation errors.

Music signals vary much more rapidly, that is, have higher bandwidth than voice signals. Thus for example, audio compact disk (CD) systems assume a bandwidth of 22 kHz and sample the music signals at 44 kilosamples/second and at a resolution of 16 bits/sample. For a stereo music system, this sampling results in a bit rate of 44,000 samples/second $\times$ 16 bits/sample $\times$ 2 channels = 1.4 Mbps. One hour of music will then produce 317 Mbytes of information. More complex compression techniques can be used to reduce the bit rate of the digitized signal. For example, the subband coding technique used in the MPEG audio standard, for example, MP3, can reduce the bit rate by a factor of 14 to about 100 kbps.

Video signals ("moving pictures" or "flicks") can be viewed as a succession of pictures that is fast enough to give the human eye the appearance of continuous motion. If there is very little motion, such as a close-up view of a face in a videoconference, then the system needs to transmit only the differences between successive pictures. Typical videoconferencing systems operate with frames of 176×144 pixels at 10 to 30 frames/second as shown in Figure 3.3a. The color of each pixel is initially represented by 24 bits, that is, 8 bits per color component. When compressed, these videoconferencing signals produce bit rates in the range of several hundred kilobits/second as shown in Table 3.2.

Broadcast television requires greater resolution (720×480 pixels/frame) than videoconferencing requires, as shown in Figure 3.3b, and can contain a high degree of motion. The MPEG-2 coding system can achieve a reduction from the uncompressed

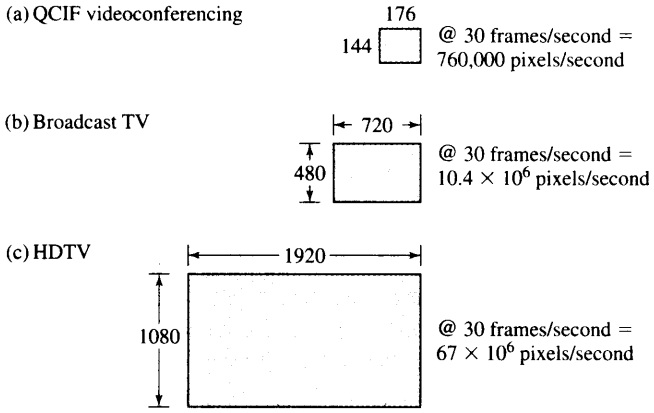


FIGURE 3.3 Video image pixel rates.

bit rate of 249 Mbps to the range of 2 to 6 Mbps. Current DVD movies are encoded in the range of 4 Mbps.[5] The Advanced Television Systems Committee (ATSC) U.S. standard for high-definition television applies the MPEG-2 coding system in a system that operates with more detailed 1920 × 1080 pixel frames at 30 frames/second as shown in Figure 3.3c. The 16:9 aspect ratio of the frame gives a more theaterlike experience; ordinary television has a 4:3 aspect ratio. The uncompressed bit rate is 1.6 gigabits/second. The MPEG-2 coding can reduce this to 19 to 38 Mbps, which can be supported by digital transmission over terrestrial broadcast and cable television systems.[6]

We have seen in this section that the information generated by various applications can span a broad range of bit rates. Even a specific information type, for example, voice or video, can be represented over a wide range of bit rates and qualities. There is a cost associated with the signal processing required to compress a signal, and in general, this cost increases as the compression ratio increases. On the other hand, there is also a cost associated with the bit rate of the transmission system. The choice of bit rate to represent an information signal and bit rate to transmit the signal depends on the relative value of these two costs. For example, in cellular telephony the cost of transmission is expensive and high-performance voice compression is used. In the case of file transfer in a high-bandwidth local area network, the cost of transmission is cheap, so compression is seldom used. However, file transfer using a long-distance telephone line is expensive, so compression is highly desirable for large files.

[5]Some DVD players can display the variation of the bit rate while playing a movie. Try the display menu in a DVD player and check for the variation in bit rate for different types of movies, for example, cartoons versus sport scenes.

[6]MPEG and MP3 are discussed in Chapter 12.

3.2 WHY DIGITAL COMMUNICATIONS?[7]

A transmission system makes use of a physical **transmission medium** or **channel** that allows the propagation of energy in the form of pulses or variations in voltage, current, or light intensity as shown in Figure 3.4. Copper wire pairs, coaxial cable, optical fiber, infrared, and radio are all examples of transmission media. In analog communications the objective is to transmit a waveform, which is a function that varies continuously with time, as shown in Figure 3.5a. For example, the electrical signal coming out of a microphone corresponds to the variation in air pressure corresponding to sound. This function of time must be reproduced exactly at the output of the analog communication system. In practice, communications channels cannot achieve perfect reproduction, so some degree of distortion is unavoidable.

In digital transmission the objective is to transmit a given symbol that is selected from some finite set of possibilities. For example, in binary digital transmission the objective is to transmit either a 0 or a 1. This can be done, for instance, by transmitting positive voltage for a certain period of time to convey a 1 or a negative voltage to convey a 0, as shown in Figure 3.5b. The task of the receiver is to determine the input symbol. The positive or negative pulses that were transmitted for the given symbols can undergo a great degree of distortion. Where signaling uses positive or negative voltages, the system will operate correctly as long as the receiver can determine whether the original voltage was positive or negative. For example, the waveform in Figure 3.5b corresponds to binary sequence 1, 0, 1. Despite significant distortion, the original binary sequence can still be discerned from the received signal shown in the figure.

3.2.1 Comparison of Analog and Digital Transmission

The cost advantages of digital transmission over analog transmission become apparent when transmitting over a long distance. Consider, for example, a system that involves transmission over a pair of copper wires. As the length of the pair of wires increases, the signal at the output is attenuated and the original shape of the signal is increasingly distorted. In addition, interference from extraneous sources, such as radiation from radio signals, car ignitions, and power lines, as well as noise inherent in electronic systems result in the addition of random noise to the transmitted signal. To transmit over long distances, it is necessary to introduce **repeaters** periodically to compensate for the attenuation and distortion of the signal, as shown in Figure 3.6. Such signal reconditioning is fundamentally different for analog and digital transmission.

[7]This section summarizes the main results of Sections 3.3, 3.4 and 3.5, allowing these three sections to be skipped if necessary.

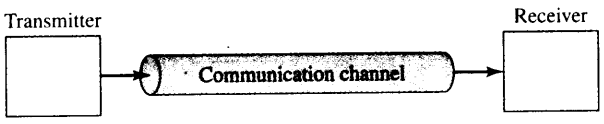


FIGURE 3.4 General transmission system.

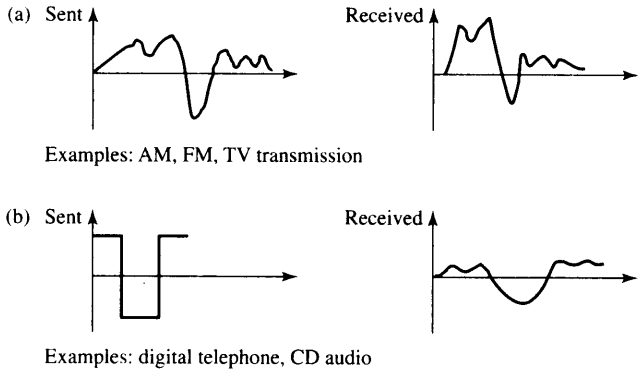


FIGURE 3.5 (a) Analog transmission requires an accurate replica of the original signal whereas (b) digital transmission reproduces discrete levels.

In an analog communication system, the task of the repeater is to regenerate a signal that resembles as closely as possible the signal at the input of the repeater segment. Figure 3.7 shows the basic functions carried out by the repeater. The input to the repeater is an attenuated and distorted version of the original transmitted signal plus the random noise added in the segment. At the transmitter the original signal is much higher in power than the ambient noise. If the signal is attenuated too much then the noise level can become comparable to the desired signal. The function of the repeater is to boost the signal power before this occurs. First the repeater deals with the attenuation by amplifying the received signal. To do so the repeater multiplies the signal by a factor that is the reciprocal of the attenuation a. The resulting signal is still distorted by the channel.

The repeater next uses a device called an **equalizer** in an attempt to eliminate the distortion. The source of the distortion in the signal shape has two primary causes. The first cause is that different frequency components of the signal are attenuated differently.[8] In general, high-frequency components are attenuated more than low-frequency components. The equalizer compensates for this situation by amplifying different frequency components by different amounts. The second cause is that different frequency components of a signal are delayed by different amounts as they propagate through the channel.

[8]Periodic signals can be represented as a sum of sinusoidal signals using Fourier series. Each sinusoidal signal has a distinct frequency. We refer to the sinusoidal signals as the "frequency components" of the original signal. (Fourier series are reviewed in Appendix 3B.)

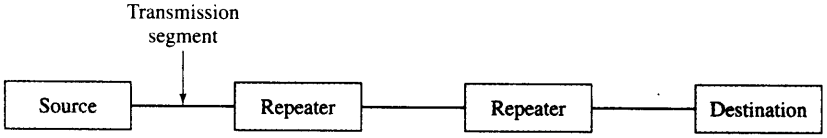


FIGURE 3.6 Typical long-distance link.

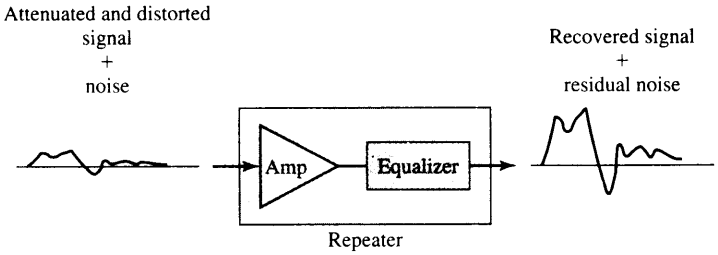


FIGURE 3.7 An analog repeater.

The equalizer attempts to provide differential delays to realign the frequency components. In practice it is very difficult to carry out the two functions of the equalizer. For the sake of argument, suppose that the equalization is perfect. The output of the repeater then consists of the original signal plus the noise.

In the case of analog signals, the repeater is limited in what it can do to deal with noise. If it is known that the original signal does not have components outside a certain frequency band, then the repeater can remove noise components that are outside the signal band. However, the noise within the signal band cannot be reduced and consequently the signal that is finally recovered by the repeater will contain some noise. The repeater then proceeds to send the recovered signal over the next transmission segment.

The effect on signal quality after multiple analog repeaters is similar to that in repeated recordings using analog audiocassette tapes or VCR tapes. The first time a signal is recorded, a certain amount of noise, which is audible as hiss, is introduced. Each additional recording adds more noise. After a large number of recordings, the signal quality degrades considerably.[9] A similar effect occurs in the transmission of analog signals over multiple repeater segments.

Next consider the same copper wire transmission system for digital communications. Suppose that a string of 0s and 1s is conveyed by a sequence of positive and negative voltages. As the length of the pair of wires increases, the pulses are increasingly distorted and more noise is added. A **digital regenerator** is required as shown in Figure 3.8. The sole objective of the regenerator is to restore with high probability the original binary stream. The regenerator also uses an equalizer to compensate for the

[9]The recent introduction of digital recording techniques in consumer products almost makes this example obsolete! Another example involves noting the degradation in image quality as a photocopy of a photocopy is made.

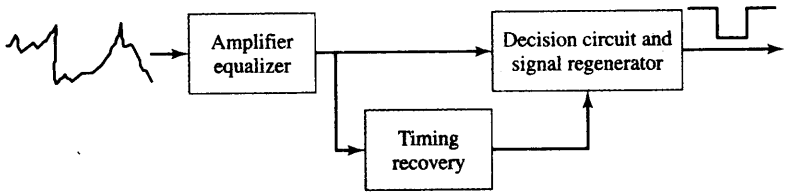


FIGURE 3.8 A digital regenerator.

distortion introduced by the channel. However, the regenerator does not need to completely recover the original shape of the transmitted signal. It only needs to determine whether the original pulse was positive or negative. To do so, a digital regenerator is organized in the manner shown in Figure 3.8.

A timing recovery circuit keeps track of the intervals that define each pulse by noting the transition instants between pulses. The decision circuit then samples the signal at the midpoint of each interval to determine the polarity of the pulse. In a properly designed system, in the absence of noise, the original symbol would be recovered every time, and consequently the binary stream would be regenerated exactly over any number of regenerators and hence over arbitrarily long distances. Unfortunately, noise is unavoidable in electronic systems, which implies that errors will occur from time to time. An error occurs when the noise signal is sufficiently large to change the polarity of the original signal at the sampling point. Digital transmission systems are designed for very low bit error rates, for example, 10^{-7}, and in optical transmission systems even 10^{-12}, which corresponds to one error in every trillion bits!

The impact on signal quality in multiple digital regenerators is similar to the digital recording of music where the signal is stored as a file of binary information. We can copy the file digitally any number of times with extremely small probabilities of errors being introduced in the process. In effect, the quality of the sound is unaffected by the number of times the file is copied.

The preceding discussion shows that digital transmission has superior performance over analog transmission. Digital regenerators eliminate the accumulation of noise that takes place in analog systems and provide for long-distance transmission that is nearly independent of distance. Digital transmission systems can operate with lower signal levels or with greater distances between regenerators than analog systems can. This factor translates into lower overall system cost and was the original motivation for the introduction of digital transmission.

Another advantage of digital transmission over analog transmission is in monitoring the quality of a transmission channel while the channel is in service. In digital transmission systems, certain predetermined patterns can be imposed on the transmitted information. By checking these patterns it is possible to determine the error rate in the overall channel. Nonintrusive monitoring is much more difficult in analog transmissions systems.

Over time, other benefits of digital transmission have become more prominent. Networks based on digital transmission can multiplex and switch *any* type of information that can be represented in digital form. Thus digital networks are suitable for handling many types of services. Digital transmission also allows networks to exploit the advances in digital computer technology to increase not only the volume of information that can be transmitted but also the types of processing that can be carried out within the network, that is, error correction, data encryption, and the various types of network protocol processing that are the subject of this book.

3.2.2 Basic Properties of Digital Transmission Systems

The purpose of a **digital transmission** system is to transfer a sequence of 0s and 1s from a transmitter (on the left end) to a receiver (on the right) as shown in Figure 3.9.

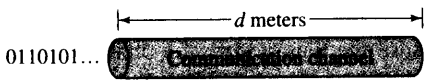


FIGURE 3.9 A digital transmission system.

We are particularly interested in the **bit rate** or transmission speed as measured in bits/second. The bit rate R can be viewed as the cross-section of the information pipe that connects the transmitter to the receiver. As the value of R increases, the volume of information that can flow across the pipe per second increases.

The transmission system uses pulses or sinusoids to transmit binary information over a physical transmission medium. A fundamental question in digital transmission is *how fast* can bits be transmitted *reliably* over a given medium. This capability is clearly affected by several factors including:

- The amount of *energy* put into transmitting each signal.
- The *distance* that the signal has to traverse (because the energy is dissipated and dispersed as it travels along the medium).
- The amount of *noise* that the receiver needs to contend with.
- The *bandwidth* of the transmission channel, which we explain below.

A transmission channel can be characterized by its effect on input sinusoidal signals (tones) of various frequencies. A sinusoid of a given frequency f Hertz is applied at the input, and the sinusoid at the output of the channel is measured. The ability of the channel to transfer a tone of the frequency f is given by the **amplitude-response function** $A(f)$, which is defined as the ratio of the amplitude of the output tone divided by the amplitude of the input tone. Figure 3.10 shows the typical amplitude-response functions of a low-pass channel and its idealized counterpart. As indicated by the figure, the low-pass channel passes sinusoidal signals up to some frequency w and blocks sinusoids of higher frequencies. The **bandwidth of a channel** is defined as the range of frequencies that is passed by a channel.

Consider next what happens when an arbitrary signal is applied to a channel. Recall that the bandwidth of a *signal* W_s is defined as the range of frequencies contained in the signal. On the other hand, the bandwidth of a *channel* W_c is the range of input frequencies *passed* by the channel. Clearly, if the bandwidth of the input signal is

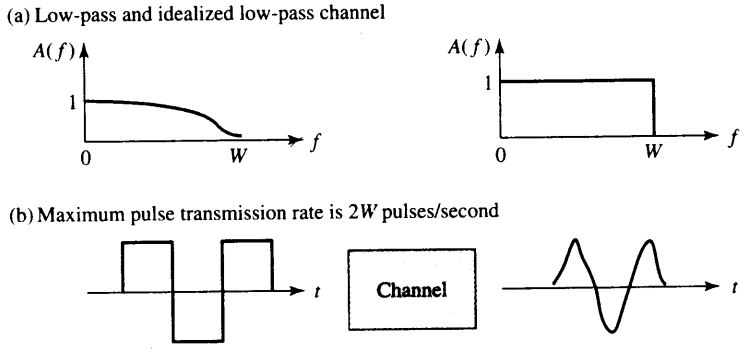


FIGURE 3.10 Typical amplitude-response functions.

larger than the bandwidth of the channel, then the output of the channel will not contain all of the frequencies of the input signal. Therefore the bandwidth of a channel limits the bandwidth of the signals that can pass through the channel. Now let's see what this implies for the transmission of digital signals through a transmission channel. Figure 3.10b shows a typical digital signal before it is input into a channel. The polarity of each pulse corresponds to one bit of information. As we increase the signaling speed, the pulses become narrower and so the signal varies more quickly. Thus higher signaling speed translates into higher signal bandwidth. However we just found that the bandwidth of a channel limits the bandwidth of the input signal that can be passed. Therefore we conclude that the bandwidth of a channel places a limit on the rate at which we can send pulses through the channel.

A major result for digital transmission pertains to the maximum rate at which pulses can be transmitted over a channel. If a channel has bandwidth W, then the narrowest pulse that can be transmitted over the channel has duration $\tau = 1/2W$ seconds. *Thus the maximum rate at which pulses can be transmitted through the channel is given by*: $r_{max} = 2W$ pulses/second.[10]

We can transmit binary information by sending a pulse with amplitude $+A$ to send a 1 bit and $-A$ to send a 0 bit. Each pulse transmits one bit of information, so this system then has a bit rate of $2W$ pulses/second $\times$ 1 bit/pulse $= 2W$ bps. We can increase the bit rate by sending pulses with more levels. For example, if pulses can take on amplitudes from the set $\{-A, -A/3, +A/3, +A\}$ to transmit the pairs of bits {00, 01, 10, 11}, then each pulse conveys two bits of information and the bit rate is $4W$ bps. Thus in general, *if we use* ***multilevel transmission*** *with* $M = 2^m$ *amplitude levels, we can transmit at a bit rate*

$$R = 2W \text{ pulses/second} \times m \text{ bits/pulse} = 2Wm \text{ bits/second} \qquad (3.1)$$

In the absence of noise, the bit rate can be increased without limit by increasing the number of signal levels M. However, noise is an impairment encountered in all communication channels. Noise consists of extraneous signals that are added to the desired signal at the input to the receiver. Figure 3.11 gives two examples where the desired signal is a square wave and where noise is added to the signal. In the first example the amplitude of the noise is less than that of the desired signal, and so the desired signal is discernable even after the noise has been added. In the second example the noise amplitude is greater than that of the desired signal, which is now more difficult to discern. The **signal-to-noise ratio (SNR)**, defined in Figure 3.11, measures the relative amplitudes of the desired signal and the noise. The SNR is usually stated in decibels (dB).

Returning to multilevel transmission, suppose we increase the number of levels while keeping the maximum signal levels $\pm A$ fixed. Each increase in the number of signal levels requires a reduction in the spacing between levels. At some point these reductions will imply significant increases in the probability of detection errors as the noise will be more likely to convert the transmitted signal level into other signal levels. *Thus the presence of noise limits the reliability with which the receiver can correctly determine the information that was transmitted.*

[10]The term *baud rate* is also used to denote the signaling rate in pulses/second.