1

Networking Basics

Some computers are independent units, with little need to exchange information with other computers near or far. At most, these computers may use local interfaces such as USB or RS-232 to communicate with printers or other devices close at hand.

But with a network connection, a computer can reach beyond its local interfaces to send and receive information of any kind, over distances large and small, via wires or through the air. Computers of different types can communicate using network protocols supported by all. In a network of embedded systems, each system can communicate with the other systems in the network, sharing information and sending and responding to requests as needed. Desktop computers in the network can monitor and control the operation of the embedded systems.

Many local networks follow the networking standard popularly known as Ethernet. Ethernet networks are capable and flexible. Many products designed for use in networks have support for Ethernet built in. A router, or

gateway, enables an Ethernet network to communicate with computers in other networks, including computers on the Internet.

Two or more computers that share a network connection form a local area network, or LAN. The smallest network links just two computers. For example, a data logger might connect to a remote computer that receives and displays the logger's data. Or a personal computer (PC) may use a network connection to monitor and control a piece of equipment. At the other extreme, the Internet is the largest network. With an Internet connection, the computers in a local network can access resources on the Internet and make local resources available to any computer on the Internet.

To design and program embedded systems for networking, you need to understand the elements that make up a network, so this chapter begins with the basics of how networks are structured. Following this is an introduction to Ethernet, including its capabilities and how Ethernet networks manage network traffic.

Quick Start: The Elements of a Network

All computer networks have some things in common. Every network must have the physical components that enable the computers in the network to exchange data. And in every network, the computers must agree about how to share the data path that connects the computers, to help ensure that transmitted data gets to its destination.

Components

All networks include the following physical components:

- Two or more computers that need to communicate with each other. In the networks described in this book, at least one of the computers is an embedded system, which is a device that contains a computer dedicated to a specific task or a series of related tasks.

- A defined physical interface, to ensure that the output of a transmitting computer is compatible with the inputs of the receiving computers. For Ethernet networks, the Ethernet standard specifies this interface.
- Cables or wireless transceivers to connect the computers. Ethernet networks have several options for cables. An Ethernet interface may also connect to a device called a wireless access point, which enables the embedded system to access a wireless network.

The computers in the network must also agree on the following aspects of sharing the network:

- Rules for deciding when a computer may transmit on the network. When multiple computers share a data path, whether in a cable or wireless medium, the computers need to know when the path is available for transmitting. The Ethernet standard contains rules that specify when a computer may transmit.
- A way of identifying a transmission's intended destination. In Ethernet networks, multiple computers may receive a message intended for one computer in the network. When a message arrives at a computer's network interface, the computer needs to know whether the message is intended for itself or another computer. Every communication in an Ethernet network includes a hardware address that identifies the Ethernet interface of the intended receiver. Some communications also use Internet protocols that contain additional addressing information, such as an addresses that identify the sending and receiving computers on the Internet and a port, or process, that receives the communication at the destination computer.
- A defined format for the information sent on the network, so a computer can understand and use the information it receives from the network. In Ethernet networks, all data travels in structures called frames. Each frame includes fields for data, addressing, and other information that helps the data reach its destination without errors. The information in a frame's data field may also use protocols that help the receiver of the frame decide what to do with the received data.

Modular Design

To make designing and maintaining a network as easy as possible, most networked computers use modules, or components, that work together to handle the job of network communications. Each module is responsible for a single task or a small set of related tasks. Each module knows how to exchange information with one or more of the other modules, but the modules don't need to know details about how the other modules accomplish their tasks.

The modular approach has a couple of benefits. If each module is as independent as possible, it's easier to make changes when needed, possibly even swapping in a different module entirely, without requiring changes in the other modules. And isolating problems is easier when a single module contains all of the code to perform a function.

A module may consist of hardware, software, or a combination. A software module may be as small as a procedure or subroutine within a larger application or unit of code. Or a module may be a library of routines or a class or package of classes in a separate file.

In an embedded system, the program code may be referred to as firmware, which typically means that the code is stored in Flash memory or another nonvolatile memory chip, rather than on a disk drive. In general, with software stored on a drive, users can install, run, and uninstall applications as needed. In contrast, firmware tends to be an integral, seldom-changing part of the device. Users may have the ability to load new firmware into a device, but the new firmware is typically an update or upgrade to existing code, rather than an entirely different type of application.

The Network Protocol Stack

You can think of the modules used in networking as being stacked in layers, one above another. A computer's network protocol stack consists of the modules involved with networking. Figure 1-1 shows an example of a net-

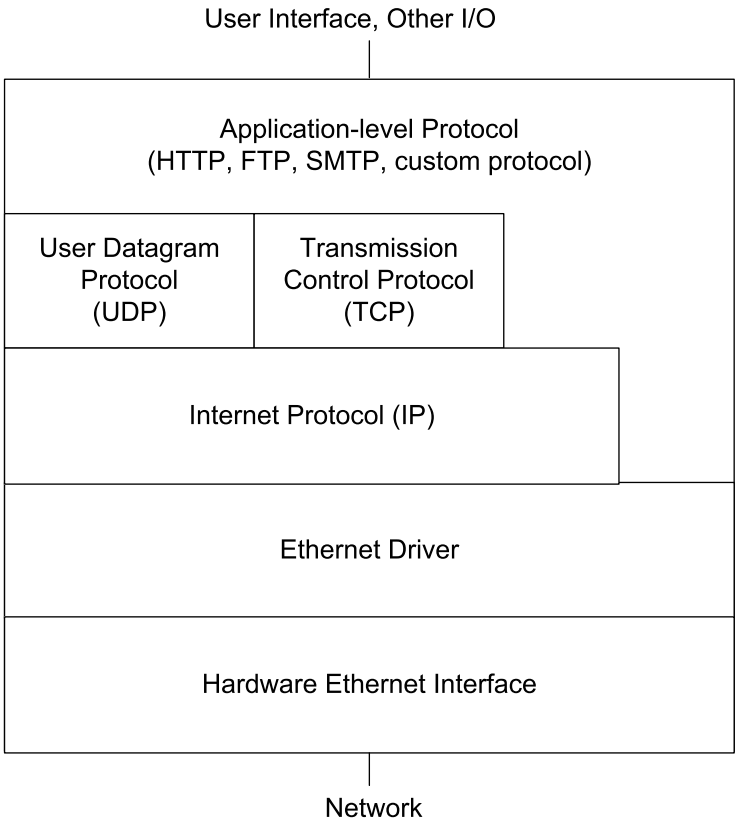

Figure 1-1: The networking support in most computers uses a layered model, where a variety of components each manage a portion of the job of network communications.

work protocol stack for a computer that connects to an Ethernet network and supports common Internet protocols.

(To prevent confusion, I should point out that this use of the term stack has nothing to do with the internal stack of a computer's central processing unit (CPU). A CPU's stack is a special area of memory for temporary storage. This type of stack has no direct relation to networking.)

At the bottom of the stack is the hardware interface to the network cable. At the top of the stack is a module or modules that provide data to send on the network and use the data received from the network. In the middle there

may be one or more modules involved with addressing, error-checking, and providing and using status and control information.

In transmitting, a message travels down the stack from the application layer that initiates the message to the network interface that places the message on the network. In receiving, the message travels up the stack from the network interface to the application layer that uses the data in the received message.

The number of layers a message passes through can vary. For some messages that travel only within a local network, the application layer can communicate directly with the Ethernet driver. Messages that travel on the Internet must use the Internet Protocol. Messages that use the Internet Protocol can also use the User Datagram Protocol or the Transmission Control Protocol to add error checking or flow-control capabilities.

The Application: Providing and Using Network Data

The application provides data to send on the network and uses data received from the network. An application often has a user interface that enables users to request data from a computer on the network or provide data to send on the network. In an embedded system, the user interface may just enable basic configuring and monitoring functions, while the system performs its network communications without user intervention.

The data that the application sends and receives may be anything: a single byte; a line of text; a request for a Web page; the contents of a Web page; a file containing text, an image, binary data, or program code; or anything that a computer wants to send to another computer in the network.

The data sent by an application follows a protocol, or set of rules, that enables the application at the receiving computer to understand what to do with the received data. An application may use a standard protocol such as the hypertext transfer protocol (HTTP) for requesting and sending Web pages, the file transfer protocol (FTP) for transferring files, or the simple mail transfer protocol (SMTP) or Post Office Protocol (POP3) for e-mail messages. Applications may also send and receive data using application-specific protocols.

In an embedded system, the application might be a module that periodically reads and stores sensor readings or the states of other external signals, or an application might use received data to control motors, relays, or other circuits. An embedded system can function as a Web server that receives and responds to requests for Web pages, which may enable users to provide input or view real-time data. Embedded systems can send and receive information via e-mail and in files via FTP.

An application layer may support multiple processes, or tasks. For example, a single system might host a Web page and also provide an FTP server that makes files available for downloading. Port numbers can identify specific processes at the destination computer.

TCP and UDP: Error Checking, Flow Control, and Ports

A network communication often includes additional information to help data get to its destination efficiently and without errors. A module that supports the Transmission Control Protocol (TCP) can add information for use in error checking, flow control, and identifying an application-level process at the source and destination computers.

Error-checking values help the receiver detect when received data doesn't match what was sent. Flow-control information helps the sender determine when the receiver is ready for more data. And a value that identifies an application-level port, or process, can help in routing received data to the correct process in the application layer.

TCP performs all of these functions. Many Internet and local-network communications such as requests for Web pages and sending and receiving e-mail use TCP. Windows and other operating systems have support for TCP built in. Development kits for network-capable embedded systems often include libraries or packages with TCP support.

In sending data using TCP, the application layer passes the data to send and values that identify the data's source and destination to a TCP layer. The TCP layer creates a TCP segment that consists of a header followed by the application data (Figure 1-2). The header is a defined structure with fields

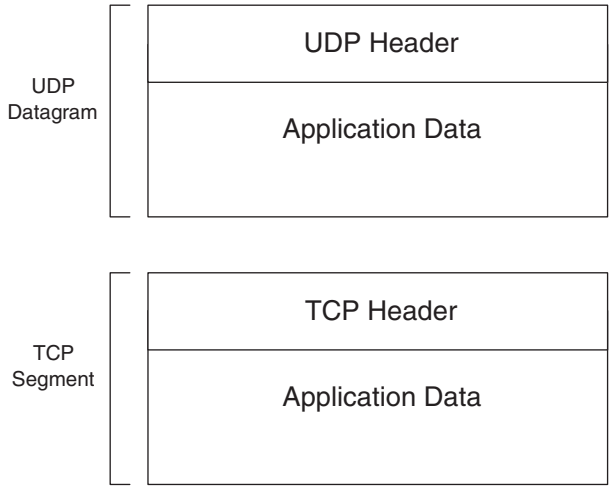

Figure 1-2: The UDP and TCP layers add a header to the data payload before passing the data down the stack. In the opposite direction, the UDP and TCP layers strip the headers before passing the data up the stack.

containing information used in error checking, flow control, and routing the message to the correct port at the destination. The TCP layer doesn't change the message to be sent. It just places the message in the data portion of the TCP segment. The TCP segment encapsulates, or provides a container for, the data received from the application layer. The TCP layer then passes the segment to the IP layer for transmitting on the network.

In the other direction, the TCP layer receives a segment from the IP layer, strips the TCP header, and passes the segment to the port specified in the TCP header.

A simpler alternative to TCP is the User Datagram Protocol (UDP). Like a TCP segment, a UDP datagram has a header, followed by a data portion that contains the application data. UDP includes fields for specifying ports and optional error-checking, but no support for flow control. Windows and many development kits for embedded systems include support for UDP.

Chapter 5 has more about TCP and UDP.

In some networks, communications may skip the TCP/UDP layer entirely. For example, a local network of embedded systems may have no need for

flow control or additional error-checking beyond what the Ethernet frame provides. In these cases, an application may communicate directly with a lower layer in the network protocol stack, such as the IP layer or Ethernet driver.

IP: Internet Addressing and Routing

The Internet Protocol (IP) layer can help data get to its destination even if the source and destination computers are on different local networks. As the name suggests, the Internet Protocol enables computers on the Internet to communicate with each other. Because IP is closely tied to TCP and UDP, local networks that use TCP and UDP also use IP.

The term *TCP/IP* refers to communications that use TCP and IP. The term can also refer more broadly to the suite of protocols that includes TCP, IP, and related protocols such as UDP.

In Ethernet networks, a unique hardware address identifies each interface on the network. IP addresses are more flexible because they aren't specific to a network type. A message that uses IP can travel through different types of networks, including Ethernet, token-ring, and wireless networks, as long as all of the networks support IP.

In sending a message, the TCP layer passes the TCP segment and the source and destination addresses to the IP layer. The IP layer encapsulates the TCP segment in an IP datagram, which consists of a header followed by a data portion that may contain a UDP datagram or a TCP segment (Figure 1-3). The header has fields for the source and destination IP addresses, error checking of the header, routing, and a value that identifies the protocol, such as TCP or UDP, used by the data portion.

In a similar way, a UDP layer may pass a UDP datagram to the IP layer.

In receiving a message, the IP layer receives an IP datagram from a lower level in the network stack. The IP layer performs error-checking and uses the protocol value to determine where to pass the contents of the data portion.

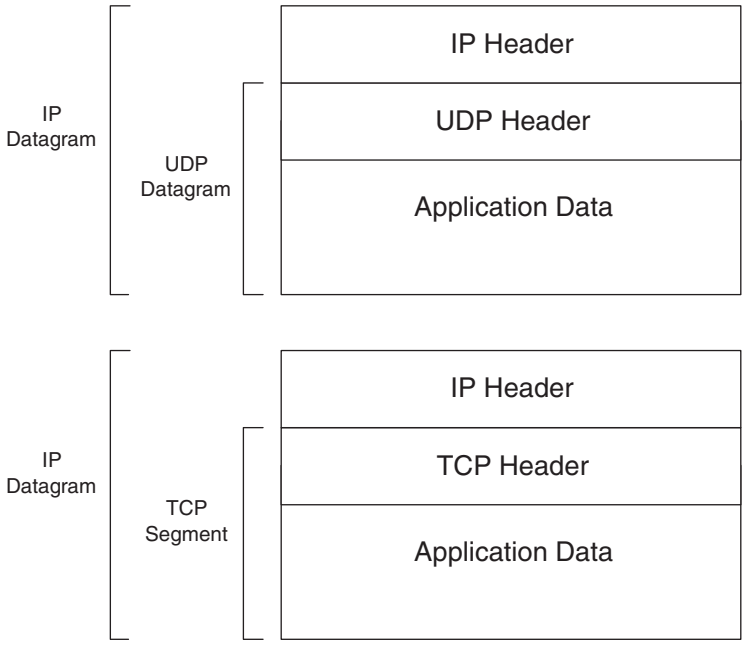

Figure 1-3: The IP layer adds a header to a UDP datagram or TCP segment before passing the data down the stack. In the opposite direction, the IP layer strips the IP header before passing a UDP datagram or TCP segment up the stack.

In the IP header, the source and destination IP addresses identify the sending and receiving computers. Each computer in a network that uses IP addresses must have an address that is unique within the network or networks that the sending computer can communicate with. Local networks can use addresses in three blocks reserved for local networks. A computer that communicates over the Internet must have an address that is different from the address of every other computer on the Internet. The Internet Corporation for Assigned Names and Numbers (ICANN) assigns blocks of addresses to Internet Service Providers and others who may in turn assign portions of their addresses to other users.

Three protocols often used along with IP for assigning and learning IP addresses are the dynamic host configuration protocol (DHCP), the domain name system (DNS) protocol, and the Address Resolution Protocol (ARP).

A computer functioning as a DHCP server can use DHCP to assign IP addresses to the computers in a local network. A computer that wants to learn the IP address of a domain such as *Lvr.com* can use the DNS protocol to request the information from a computer functioning as a DNS server. And a computer that wants to learn the Ethernet hardware address that corresponds to an IP address in a local network can broadcast an ARP request for this information.

Chapter 4 has more details about IP and related protocols.

A communication in a local network that doesn't use TCP or UDP may not require IP. Instead, the application layer may communicate directly with a lower layer such as the Ethernet driver.

The Ethernet Driver and Controller: The Hardware Interface

In an Ethernet network, the interface to the network is an Ethernet controller chip and its driver. The Ethernet driver contains program code that manages communications between the controller chip and a higher level in the network protocol stack. To send an IP datagram over an Ethernet network, the IP layer passes the datagram to the Ethernet controller's driver. The driver instructs the Ethernet controller to transmit an Ethernet frame containing the datagram, preceded by a header that contains addressing and error-checking information (Figure 1-4).

In receiving an IP datagram from the network, the Ethernet controller checks to see if the destination address matches the interface's hardware address or a multicast or broadcast address that the controller is configured to accept. If there is a match, the controller checks for errors and the driver passes the datagram or an error indication to the IP layer.

Chapter 3 has more about Ethernet controllers.

Clients and Servers

In some networks, the computers may send messages to other computers in the network at any time. For example, a computer that performs monitoring functions might send an alarm notification to a master computer as soon as

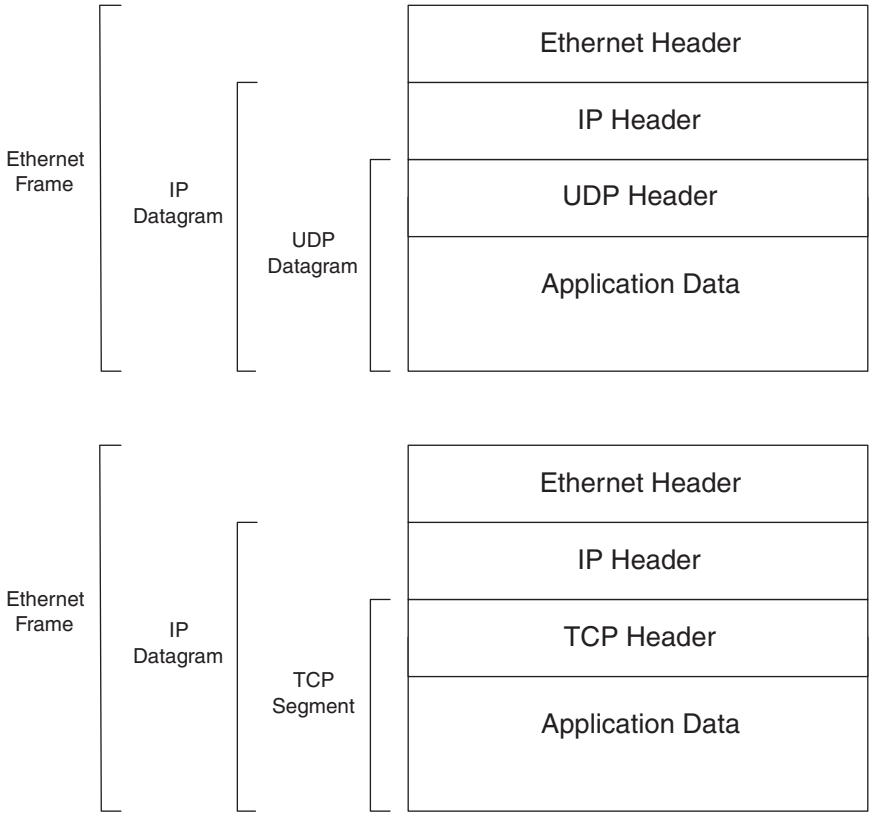

Figure 1-4: The Ethernet controller adds a header to an IP datagram before sending the datagram on the network. In the opposite direction, the Ethernet controller strips the Ethernet header before passing a datagram to the IP layer.

the condition causing the alarm occurs. The computer doesn't have to wait for the master computer to request the information.

In contrast, many other network communications are between a client computer, which requests resources, and a server computer, which provides the resources on request. A resource may be a Web page, a file, or any other information the server makes available. By running multiple processes, a single computer can function as both a client and a server.

The client and server must agree on a protocol for requesting and sending resources. For common tasks, there are standard protocols such as HTTP,

FTP, and SMTP, and POP3. The computers typically send the requests and responses in the data portions of TCP segments.

When you use a browser to view Web pages, the browser is functioning as a client, requesting pages from server computers that store Web pages. To function as a server that provides Web pages on request, a computer must be running server software. Server software for PCs includes Apache Software Foundation's Apache HTTP Server and Microsoft's Personal Web Server. Embedded systems can function as servers with the addition of program code that decodes and responds to received requests.

Another use for the terms client and server is to refer to computers that have established a TCP connection, which enables the computers to exchange messages using TCP. The client is the computer that initiated the connection, while the server is the computer that accepted the request to connect. Once the connection is established, either computer can send messages to the other, though a higher-level protocol may limit what each computer can do.

Requirements for Internet Communications

To communicate on the Internet, a computer in an Ethernet network has additional requirements. Internet communications of course require a physical connection to the Internet. And messages sent on the Internet must use the Internet Protocol.

Large businesses and schools are likely to have Internet access available. For others, obtaining an Internet connection involves contracting with an Internet Service Provider (ISP). The ISP provides an IP address that identifies the computer on the Internet.

Domain names such as *rabbitsemiconductor.com* and *dalsemi.com* provide a more user-friendly way to request a resource on the Internet, compared to using numeric IP addresses. Domain names are available from a variety of registrars for a yearly fee.

Local networks that connect to the Internet typically have a firewall, which is hardware, software, or a combination that protects the local network by

limiting the types of communications that local computers can send and receive. To make a server or other resource available on the Internet, you may need to configure your firewall to permit receiving requests or other communications from outside the local network.

The focus of this book is Ethernet networks, but a computer doesn't have to have an Ethernet connection to connect to the Internet. Another option is to use a modem and a dial-up connection to an ISP, using the Point-to-Point Protocol (PPP).

Chapter 4 has more about obtaining an Internet connection.

A Word about Web Servers

Many networked embedded systems function as Web servers, which respond to requests for Web pages from browsers in the network. The pages hosted by embedded systems are likely to display dynamic content that can change each time the page is requested. Examples of dynamic content include sensor readings, date and time information, or counts of Web-page visitors. Some pages may also enable users to provide data to the server, which can process the data and perhaps return a result on a Web page.

There are several ways to support dynamic content in Web pages. Applications programmed in C often use common gateway interface (CGI) and server side include (SSI) programming. Applications programmed in Java often use Java servlets. A few products, such as Netmedia's SitePlayer, support product-specific methods that may involve defining variables and inserting codes in the Web page to cause the values of the variables to display on the page. Chapter 6 and Chapter 7 have more about Web pages and dynamic content.

In Depth: Inside Ethernet

Ethernet isn't the only way to network embedded devices, but it's a popular choice. It's possible to put together and use an Ethernet network without knowing much about its inner workings. Hardware and software compo-

nents with built-in Ethernet support can shield you from the details. But a little knowledge about Ethernet can help in selecting network components, writing the software that exchanges data over the network, and troubleshooting any problems that come up.

This section discusses Ethernet's advantages and limits and how Ethernet uses frames and media-access control to help get data to its destination. Chapter 2 covers Ethernet's media systems (such as 10BASE-T), which allow a choice of cable type and network speed.

Advantages

There are many reasons why Ethernet is popular and useful for networks of embedded systems and other computers.

It's Versatile

Ethernet is versatile enough to suit many purposes. Probably the best known use for Ethernet is in linking desktop computers in offices, but that's not its only use. Ethernet can transfer any kind of data, from short messages to huge files. An Ethernet communication can take advantage of existing higher-level protocols such as TCP and IP, or it can use an application-specific protocol. Ethernet doesn't require a large or fast computer. With the addition of an Ethernet controller chip, even an 8-bit microcontroller can communicate in an Ethernet network.

It's Easy to Use

With Ethernet, much of the work has been done for you. All of the computers in the network follow standard Ethernet specifications for interconnecting, managing network traffic, and exchanging data. You don't have to design the hardware interface or invent the rules from scratch. Yet Ethernet is flexible enough to allow choices. For example, a network may use twisted pair, fiber-optic, or coaxial cable. The requirements for each cable type and speed are specified, so all you need to do is decide which cable type best fits your application and select cable of that type.

A Wide Selection of Products Is Available

Hardware, software, and debugging tools for Ethernet are readily available. Ethernet's popularity means that components and tools are easy to find and inexpensive. Many PCs and other desktop computers have Ethernet support built in. At most, a PC requires an expansion card or adapter to provide the hardware interface. Windows and other operating systems include software support for Ethernet networking.

Designers of embedded systems have a good selection of modules with Ethernet capability. Many modules include a CPU, while others contain just a controller chip and an interface that you connect to your own CPU. Or you can put together your own circuits by selecting a CPU, Ethernet controller, and related components. Code to support TCP/IP, and related protocols is available from a variety of sources. Many vendors of Ethernet-capable modules also provide support for TCP/IP. Debugging tools such as bus analyzers are readily available.

The Hardware Controls Network Access

With Ethernet, the hardware manages the network traffic, so there's no need for software to control network access. In a network that uses half-duplex interfaces, the computers share a transmission path, and all computers have equal access when the network is idle. When a computer has something to send, its Ethernet controller waits for the network to be idle and then attempts to transmit. If two or more interfaces try to transmit at the same time, the interfaces detect a collision and each delays a random amount of time, then tries again. In a network that uses full-duplex interfaces, each computer has its own transmission path to an Ethernet switch, which manages the traffic to each connected computer.

It's Fast

Ethernet is fast. It supports speeds from 10 Megabits per second (Mb/s) to 10 Gigabits per second (Gb/s). Ten Mb/s is adequate for many embedded systems. The hardware to support slower speeds is generally less expensive, but the higher speeds are there if needed.

It Can Span Long Distances

A single twisted-pair cable between two computers or between a repeater hub or switch and a computer can be 100 meters. A half-duplex segment of fiber-optic cable in a 10-Mb/s system can be as long as 2000 meters, while a full-duplex segment can be as long as 5 kilometers. With repeater hubs or switches, a network can span even longer distances. A router can enable a network to communicate with other networks, including the entire Internet.

Interfaces are Electrically Isolated

Every Ethernet interface must be electrically isolated from its network cable. The isolation protects the computer's circuits from damaging voltages that may occur on the network. Isolation transformers meeting the standard's requirements are readily available. Fiber-optic cable doesn't conduct electricity, so connections to fiber-optic networks are isolated by definition.

The Cost Is Reasonable

Because Ethernet and TCP/IP are popular, hardware and software are available from a variety of sources at reasonable cost, and sometimes for free. Support for Ethernet and TCP/IP is built into or easily added to computers of all types, including development boards for embedded systems.

Limits

Ethernet isn't the answer for every embedded system's communications needs. For some systems, there are simpler, cheaper, or otherwise more appropriate ways to network.

Cost

If keeping the cost to an absolute minimum is essential, there are cheaper interfaces that are suitable for some applications. For example, the EIA-485 interface, popularly known as RS-485, requires only very inexpensive transceivers and can use the asynchronous communications port available in most microcontrollers. RS-485 supports communications at up to 10 Mb/s and distances of up to 4000 feet, though the maximum distance decreases with speed.

A downside is that the RS-485 specification doesn't define protocols for addressing or for determining when a computer can transmit on the network, so the developer needs to provide these.

Another alternative for inexpensive short-distance networks is synchronous interfaces such as I^2C, Microwire, and Serial Peripheral Interface (SPI). Philips Semiconductor's P82B715 I^2C bus extender chip adds buffering that enables longer cables in an I^2C network. Maxim Semiconductor's 1-Wire network is another option for shorter links. Interfaces such as the Universal Serial Bus (USB) and IEEE-1394 (Firewire) provide a way for PCs to communicate with multiple peripherals.

While each of these interfaces is appropriate for some applications, none matches the combination of flexibility, speed, ease of use, and wide support that Ethernet offers.

Real-time Limits

Ethernet alone doesn't guarantee real-time transfers, or transfers that will occur with minimal delay or at precise times or intervals. Because a device may have to wait to transmit on the network, a device can't know exactly when a message will transmit. Generally though, Ethernet transmissions have minimal delays unless the network is extremely busy.

If an application requires greater control over when a transmission takes place, there's nothing in the Ethernet standard that prevents adding a protocol to support greater control within a local network. For example, a master interface could query each of the other interfaces in a local network in turn, with these interfaces transmitting only when requested by the master. With this arrangement, the master can ensure that the network is idle when an interface tries to transmit.

For some applications, the computers in the network must collect or act on data with minimal delays, but the transfer of the data in the network doesn't have to be in real time. For example, an embedded system may collect periodic measurements, then transmit a block of measurements at its leisure to a PC.

Efficiency

Ethernet isn't very efficient when transferring small amounts of data. All Ethernet data travels in structures called frames. Each frame must have between 46 and 1500 data bytes. Along with the data, each frame includes 26 bytes of synchronizing, addressing, error-detecting, and other identifying information. So to transmit a single byte of data, the frame that contains the byte must also include 26 bytes of overhead plus 45 bytes of padding. Other protocols such as TCP and IP add more overhead that a specific application may not need.

Still, all that really matters is that messages get to their destination on time, and unless the network traffic is very heavy, it generally doesn't matter if the data format isn't as efficient as possible.

Power Consumption

Ethernet isn't the best solution if your device must be extremely low power. Power consumption for an Ethernet controller chip can be 50 milliamperes or more at 5 volts. Most chips support a low-power mode that can reduce power consumption when data isn't transmitting. Still, I^2C and some EIA-485 interfaces can have much lower power consumption overall.

Using a PC for Network Communications

An option worth considering for some embedded systems is to let a PC handle the network communications. The embedded system can connect to the PC using any appropriate local interface (USB, RS-232, or parallel port). The PC can then provide the network connection and an application that transfers data between the embedded system and the network. For example, an embedded system might monitor environmental conditions and use a USB connection to send the readings to a PC. The PC might host a Web page that displays the readings. With this arrangement, the embedded system doesn't have to directly support network communications at all.

The IEEE 802.3 Standard

Just about every popular computer interface has a standard, or specification document, that serves as an ultimate reference that defines what circuit designers and programmers need to know in order to use the interface. At minimum, a standard defines the electrical characteristics of the interface's signals. A standard may also specify data formats, software protocols, connectors, and cables.

The Institute of Electrical and Electronics Engineers (IEEE) is responsible for the specification popularly known as Ethernet. The IEEE's members participate in developing and maintaining many computer-related standards. The Ethernet standard and related documents are available from *www.ieee.org*.

A Brief History

Ethernet originated at Xerox Corporation in the 1970s. An early description appeared in the article *Ethernet: Distributed Packet Switching for Local Computer Networks*, in the July 1976 issue of *Communications of the ACM*, a publication of the Association for Computing Machinery. The article was by Robert M. Metcalfe and David R. Boggs of the Xerox Palo Alto Research Center.

In 1980, Digital Equipment Corporation (DEC), Intel Corporation, and Xerox Corporation formalized Ethernet's description in a document titled *The Ethernet, a Local Area Network: Data Link Layer and Physical Layer Specifications*. Another name for this standard is DIX Ethernet, from the first letters of the companies involved. Xerox gave up its trademark rights to Ethernet, allowing the DIX standard to be an open standard not under the control of a single company.

In 1985, the IEEE released its own edition of the standard. The interface described in the IEEE standard is very similar to the DIX interface and is backward-compatible with it, so networks that comply with the DIX standard also comply with the IEEE standard.

The *Ether* in Ethernet refers to luminiferous ether, which is the name given to a hypothetical medium that was once thought to serve as the propagation

medium for electromagnetic waves. The existence of ether has since been disproved, but the name lives on in the term Ethernet.

The 802.x Series

Although the Ethernet name continues in popular use, the IEEE standard uses the word sparingly. The document that describes Ethernet is IEEE Std. 802.3, with the unwieldy title of *Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications*. CSMA/CD is the method Ethernet uses for sharing the network, as described later in this chapter.

Ethernet is one in a group of IEEE standards that describe technologies for use in local and metropolitan area networks. A local network typically exists in a single room or building, while a metropolitan area network (MAN) might span a campus or city. Ethernet's main use is in local networks, though recent standards and usage have expanded its scope to larger networks such as MANs and Wide Area Networks (WANs).

All of the standards in the 802 series share the numbering convention of 802.x. The 802 signifies that the standard relates to local or metropolitan area networking, and *x* represents one or more digits that identify the specific standard.

The Ethernet standard is one of several 802-series standards that define alternate approaches for a network's physical layer and method of media-access control, which defines how computers share a network. The physical layer described in the standard includes the electrical specifications of the transceivers and the electrical and physical specifications of the connectors and cables. Media-access control includes how each computer knows when it can transmit and how the computers identify the intended receiver of a transmission.

Over the years, the IEEE has published a variety of supplements to the original Ethernet standard. In periodic updates of the main standard, the IEEE incorporates the supplements into the main standard. For example, the supplement for Gigabit Ethernet, 802.3z, is now part of the 802.3 standard.

The 802.3ae amendment, approved in 2002, adds support for 10-Gigabit Ethernet.

Options for Ethernet cables in the 802.3 standard include coaxial, twisted-pair, and fiber-optic cables. The 802.11 standard is a separate document that covers methods of wireless networking.

The 802.3 standard allows four network speeds. The original standard supported only 10 Mb/s. The standard now also supports 100 Mb/s, often called Fast Ethernet, 1 Gb/s (Gigabit Ethernet), and 10 Gb/s (10-Gigabit Ethernet, also called 10GbE).

Frames

All data in an Ethernet network travels in structures called frames. An Ethernet frame has defined fields for data and other information to help the data get to its destination and to help the destination computer determine whether the data has arrived intact.

The Ethernet controller's hardware places information to be sent in frames for transmitting, and extracts and stores the information in received frames.

Table 1-1 shows the fields in an IEEE 802.3 Ethernet frame. The fields add synchronizing bits, addressing information, an error-checking sequence, and additional identifying information to the data being sent.

Preamble and Start Frame Delimiter

The Preamble and Start Frame Delimiter fields function together. They provide a predictable bit pattern that enables the interfaces on a 10-Mb/s network to synchronize to, or match the timing of, a new frame being transmitted.

In any data link, the receiving interface needs to know when to read the bits in the transmitted data. Some interfaces, such as I^2C, are synchronous interfaces that include a clock line shared by all of the devices. With I^2C, the transmitting device writes bits when the clock is low, and a receiving device reads the bits when the clock is high.

Table 1-1: An IEEE 802.3 Ethernet frame has seven fields.

Field	Length in bytes	Purpose
Preamble	7	Synchronization pattern.
Start Frame Delimiter	1	End of synchronization pattern.
Destination Address	6	Ethernet hardware address the frame is directed to.
Source Address	6	Ethernet hardware address of the sender.
Length or Type	2	If 1500 (05DCh) or less, the length of the data field in bytes. If 1536 (0600h) or greater, the protocol used by the contents of the data field.
Data	46 to 1500	The information the source wants to send to the destination.
Frame Check Sequence	4	Error-checking value.

Other interfaces, such as Ethernet, are asynchronous, which means that the interfaces don't share a clock. RS-232 and other serial interfaces that use a UART (universal asynchronous receiver transmitter) are asynchronous. Each transmitted word begins with a Start bit. The receiver uses the leading edge of the Start bit as a timing reference to predict when to read each of the bits that will follow. An RS-232 character typically has eight or nine bits that follow the Start bit.

In contrast, a single Ethernet frame may contain over 1000 bits. Detecting a single voltage change at the beginning of a frame isn't enough to enable the interface to reliably predict when to read all of the bits that follow.

For 10-Mb/s Ethernet, the solution is to begin each frame with a known bit pattern that contains many transitions. Receiving interfaces use the pattern to synchronize to, or lock onto, the transmitted frame's clock.

The Preamble and Start of Frame Delimiter fields provide this pattern. The Preamble consists of seven identical bytes, each with the value 10101010. The Start Frame Delimiter follows the Preamble, and consists of the byte 10101011. After detecting the first transition in the Preamble, a receiving interface uses the transitions of the following bits to synchronize to the timing of the transmitting interface. The final two bits in the Start Frame Delimiter indicate the end of the Preamble.

The faster Ethernet interfaces use different methods to synchronize, but include the Preamble for compatibility.

In the earlier DIX standard, the Preamble frame is 64 bits and includes the Start-of-Frame byte, while the 802.3 standard defines the Start of Frame as a separate field. The transmitted bit patterns are the same in both cases.

Destination Address

Every Ethernet interface has a 48-bit physical, or hardware, address that identifies the interface on the network. The Destination Address field contains the physical address of the intended receiver of the frame. The receiver may be an individual interface, a group of interfaces identified by a multicast address, or a broadcast address to all interfaces in the network.

Every interface in the network reads the destination address of a received frame. If the address doesn't match the interface's physical address or a multicast or broadcast address the interface has been configured to accept, the interface ignores the rest of the frame.

The first two transmitted bits in the address have special meanings. The first bit is 0 if the address is for a single interface, and 1 if the address is a multicast or broadcast address. A broadcast address is all 1s and is directed to every interface in the network. Multicasting provides a way for an interface to communicate with a selected group of interfaces. The interfaces in the multicast group are configured to accept frames sent to a specific multicast address.

The second bit of the destination address is zero if the address was assigned by the manufacturer of the interface, which is the usual case. In the 802.3 standard, the second bit is 1 if the address is administered locally. In the DIX standard, the second bit is always zero.

Chapter 4 has more about how a sending interface learns the destination's address.

Source Address

The Source Address field contains the 48-bit physical address of the transmitting interface. See Destination Address above for more about Ethernet addresses.

Length/Type

The Length/Type field is 16 bits that can have one of two meanings. The field can indicate the number of bytes of valid data in the data field or the protocol used by the data in the field that follows.

If the value is less than or equal to 1500 decimal (5DCh), the value indicates length. The data field must contain between 46 and 1500 bytes. If there are less than 46 bytes of valid, or usable, data, the length field can indicate how many of the bytes are valid data.

If the value is greater than or equal to 1536 decimal (600h), the Length/Type field indicates the protocol that the contents of the data field use. On on-line database at the Internet Assigned Numbers Authority's Web site (*www.iana.org*) specifies values for different protocols. The value for the Internet Protocol (IP) is 800h.

Values from 1501 to 1535 decimal are undefined.

The DIX standard defined this field as a type field only. The original IEEE 802.3 standard defined the field as a length field only. The current 802.3 standard allows either use.

Data

The contents of the data field are the reason why the frame exists. The data is the information that the transmitting interface wants to send.

The data field must be between 46 and 1500 bytes. If there are fewer than 46 bytes of data, the field must include pad bytes to increase the size to 46 bytes. If the transmitting interface has more than 1500 bytes to send, it uses multiple frames.

As explained earlier in this chapter, the data field often contains additional information besides the raw data being sent. This information is typically in

headers that precede the data. The Ethernet frame doesn't care what's in the data field, as long as it meets the length requirements.

Another term for the contents of the data field is the message. The data payload, or message body, is the message minus any headers or other supplemental information in the data field.

Frames with a full 1500 data bytes are the most efficient because they have just 26 bytes, or less than 2 percent, of overhead. At the other extreme, a frame with just one data byte plus 26 bytes of headers and the required 45 bytes of padding has 71 bytes of overhead.

An Ethernet frame must be at least 512 bits (64 bytes) not including the Preamble and Start-of-Frame bits. This is the size of a frame with the minimum 46 data bytes. Receiving interfaces ignore frames that are shorter than this minimum size.

Frame Check Sequence

The Frame Check Sequence (FCS) field enables the receiving interface to detect errors in a received frame.

Electrical noise or other problems in the network can corrupt a frame's contents. A receiving interface can detect corrupted data by using the 32-bit cyclic redundancy check (CRC) value in the frame check sequence field. The transmitting interface performs a calculation, called the cyclic redundancy check, on the bytes to be sent and places the result in the frame check sequence field. The receiving interface performs the same calculation on the received bytes. If the results match, the frame's contents are almost certain to be identical to what was sent.

The Ethernet controller's hardware typically performs the CRC calculations on both ends. On detecting an error in a received frame, the controller typically sets a bit in a status register.

Media Access Control: Deciding When to Transmit

In Ethernet networks that use half-duplex interfaces, only one interface at a time can transmit, so the interfaces need a way of deciding when it's OK to

transmit. The Ethernet standard refers to the method of deciding who gets to transmit as *media access control*.

There are several ways to achieve media access control. In some networks, one computer is the master, and the other computers transmit only after receiving permission from the master. The USB interface uses this type of media-access control. In a token-passing network, the computers take turns. The token can be as basic as a register bit or sequence of bits that a computer sets to indicate possession. Only the computer holding the token can transmit. When a computer finishes transmitting, it passes the token to another computer. The token-ring network described in IEEE standard 802.5 is an example of a token-passing network.

Ethernet uses a media-access control method called *carrier sense multiple access with collision detection*, or CSMA/CD. This method allows any interface to attempt to transmit any time the network is idle. If two or more interfaces try to transmit at the same time, both wait a bit, then retry.

One way to understand how CSMA/CD works is to examine the words that make up the term. *Carrier* comes from the world of radio, where audio broadcasts ride on, or are carried by, a higher frequency called the carrier. Ethernet doesn't have a carrier in this sense. Instead, the carrier is said to be present whenever an interface is transmitting. *Carrier sense* means that an interface that wants to transmit must monitor the network and sense, or detect, when the network is idle, indicated by the absence of a carrier.

Multiple access means that no single interface controls the network traffic. Any interface can attempt to transmit on a network that has been idle for at least the amount of time defined as the interframe gap (IFG). In a 10Mb/s network, the IFG equals 96 bit times, or 9.6 microseconds.

The Ethernet controller's hardware normally handles the sending and receiving of frames, including detecting collisions and deciding when to try again after a collision. The CPU writes the data to send into memory that the controller can access, and the controller stores received data in memory that the CPU can access. The CPU uses interrupts or polling to learn of the success or failure of a transmission and the arrival of received data.

Responding to Collisions

A collision results when two interfaces in the same *collision domain* try to transmit at the same time. In half-duplex Ethernet networks, the computers connect to repeater hubs that provide attachment points for multiple interfaces. All of the interfaces that connect via repeater hubs share a collision domain, which means that every network transmission goes out to all of the interfaces. Interfaces that connect directly via coaxial cable also share a collision domain.

Another option for connecting interfaces is to use switching hubs, popularly called Ethernet switches, or just switches. Like repeater hubs, Ethernet switches provide attachment points for multiple interfaces, but interfaces that connect via switches don't share a collision domain. Instead, the switches are responsible for managing the storing and forwarding of traffic received from connected interfaces. Chapter 2 has more about repeater hubs and switches and the options for connecting interfaces.

On detecting a collision, the transmitting interface doesn't stop transmitting immediately. Instead, it continues long enough to be sure that the other transmitting interface(s) have time to detect the collision. A transmitting interface that has detected a collision always finishes sending the 64 bits of the Preamble and Start of Frame Delimiter if these haven't transmitted yet. Following these, the interface sends an additional 32 bits called the jam signal, then stops transmitting. The jam signal can be any arbitrary data except the previous frame's CRC value.

Delaying before Retransmitting

After an interface stops transmitting due to a collision, the next task is deciding when to try again. If two interfaces wait the same amount of time and then retry, another collision will occur. Instead, the Ethernet standard defines a backoff process where each interface selects a randomly chosen delay time before attempting to retransmit. This reduces the chance that two interfaces will retry at the same time, although multiple retries may be needed at times.

The delays before retrying are multiples of the interface's slot time, which is specified in units of bit times. For 10-Mb/s and Fast Ethernet, the slot time is 512 bit times, which works out to 51.2 microseconds at 10 Mb/s and 5.12 microseconds at 100 Mb/s. For Gigabit Ethernet, the slot time is 4096 bit times, or 4.096 microseconds.

For the first retry, an interface chooses randomly between retrying immediately or waiting one slot time before retrying. If the first retry results in a collision, the interface tries again, randomly selecting a delay of 0, 1, 2, or 3 slot times.

Each new attempt, up to ten tries, selects from a larger range of backoff times, as Table 1-2 shows. The formula for determining the number of slot times to choose from is 2^x, where x is the number of the retry. In the first retry, the interface selects between 2^1, or 2, slot times (0 or 1). The second retry selects from 2^2, or 4, slot times (0, 1, 2, or 3). And so on up to 2^{10}, or 1024 slot times (0 to 1023), which the interface uses for the final seven retries if needed. After 16 tries, the interface gives up and reports a failure, typically with an interrupt.

Network Limits to Ensure Collision Detection

To prevent an interface from trying to use a frame that has experienced a collision, a transmitting interface must be able to detect the collision and abandon the frame before transmitting for one slot time. The IEEE 802.3 standard specifies slot times and maximum cable lengths to ensure that a transmitting interface will always be able to detect a collision in time.

For 10-Mb/s and Fast Ethernet, one slot time equals the time required to transmit 512 bits, which is the minimum frame size minus the Preamble and Start Frame Delimiter.

For Gigabit Ethernet, the minimum frame size is still 512 bits, but the slot time is 4096 bits and a valid transmission must have at least 4096 bits. To extend a short frame to 4096 bits, the transmitting interface has two options. It can follow the frame with carrier extension bits, which are non-data symbols that keep the carrier present for the required time. Or for

Table 1-2: For each retry after a collision, the Ethernet controller selects from a larger number of delay times.

Retry number	Possible delay times, in units of slot time, chosen randomly
1	0 or 1
2	0, 1, 2, 3
3	0 to, 1, 2, 3, 4, 5, 6, 7
4	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
5	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15....30, 31
6	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15.... 62, 63
7	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15....126, 127
8	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15....254, 255
9	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15....510, 511
10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15....1022, 1023
11	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15....1022, 1023
12	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15...1022, 1023
13	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15....1022, 1023
14	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15....1022, 1023
15	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15....1022, 1023
16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15....1022, 1023
17	report failure

all frames after the first, the interface may send a burst of additional frames to fill the slot time.

Full-duplex Interfaces

An Ethernet interface can be half duplex or full duplex. With a half-duplex interface, the computer can't send data while receiving. With a full-duplex interface, the computer can transmit and receive at the same time.

Full duplex has several advantages. A full-duplex Ethernet segment doesn't need to support collision detecting because there are no collisions to detect. With two data paths available, a full-duplex segment can theoretically support twice the traffic of a half-duplex segment at the same speed. For fiber-optic links, the maximum allowed segment lengths are much greater

for full duplex. In half-duplex segments, the data's round-trip travel time must be short enough to ensure that collisions are detected in time. The length of full-duplex fiber-optic segments are limited only by optical losses in the fiber.

A full-duplex segment can link two computers, a computer and a switch, or two switches. Full-duplex segments are common in high-speed links between switches. Inexpensive Ethernet switches have made full-duplex links popular for segments that connect computers to their networks as well.

The cabling to support full duplex is present in all of the popular twisted-pair and fiber-optic media systems. Most full-duplex Ethernet media systems use a separate cable pair or fiber strand for each direction. Twisted-pair Gigabit Ethernet systems use hybrid circuits that enable simultaneous transmitting and receiving on the same wires.

To use full duplex, the interface's Ethernet controller must support full-duplex mode. Many controllers have options to configure the controller to support half or full duplex or to auto-negotiate, using full duplex if possible and half duplex otherwise. If you're using a module with provided Ethernet-controller software, the software will need to support full-duplex mode and auto-negotiation if you want to use these capabilities in the controller.

Physical Addresses

To send an Ethernet frame on the network, a computer places its physical address in the Source Address field and the places destination's physical address in the Destination Address field. The physical address has two parts, a 24-bit Organizationally Unique Identifier (OUI) that identifies the interface's manufacturer and an additional 24 bits that are unique to the piece of hardware.

For a fee, the IEEE grants the rights to use an OUI. At this writing, it's a one-time fee of $1650. An interface card purchased for a PC or an embedded-system module with an Ethernet interface typically has a physical address programmed into the hardware. If you use an address provided in this way, you can be just about 100% sure that the physical address won't

match the address of any other interface your interface might communicate with. (And if it does, it's not your fault.)

If you want to assign a different, locally administered address, some products enable changing the address.

The physical address is often expressed as a series of six hexadecimal bytes:

00-90-C2-C0-D3-EA

In the above example, 00-90-C2 is the OUI and C0-D3-EA is the unique value assigned by the owner of the OUI to a specific piece of hardware.

Each byte has a decimal value from 0 to 255. The IEEE 802.3 standard and other network standards use the more precise term *octet* instead of byte. In common usage, both terms refer to 8-bit values, but an alternate definition for byte is "the data size sufficient to hold a character," and this value can vary with the computer system.

Sometimes the sending computer doesn't know the receiving computer's physical address. To learn the physical address that corresponds to an IP address in the local network, a computer can send a broadcast message using the address resolution protocol (ARP). To obtain the IP address that corresponds to an Internet domain name, the sending computer can use the DNS protocol to send a request to a domain name server.

When transmitting to a computer outside its local network, including on the Internet, an interface sends the Ethernet frames to a router in the local network, and the router does what is needed to send the message on its way. Chapter 4 has more on ARP, domain name servers, and routers.

Using a Protocol Analyzer to View Ethernet Traffic

In troubleshooting network problems, it's often helpful to be able to view the network traffic to find out exactly what is (and isn't) transmitting. A protocol analyzer makes this possible and can be extremely helpful in tracking down problems. A protocol analyzer may be software only or it may be a hardware device that runs analyzer software.

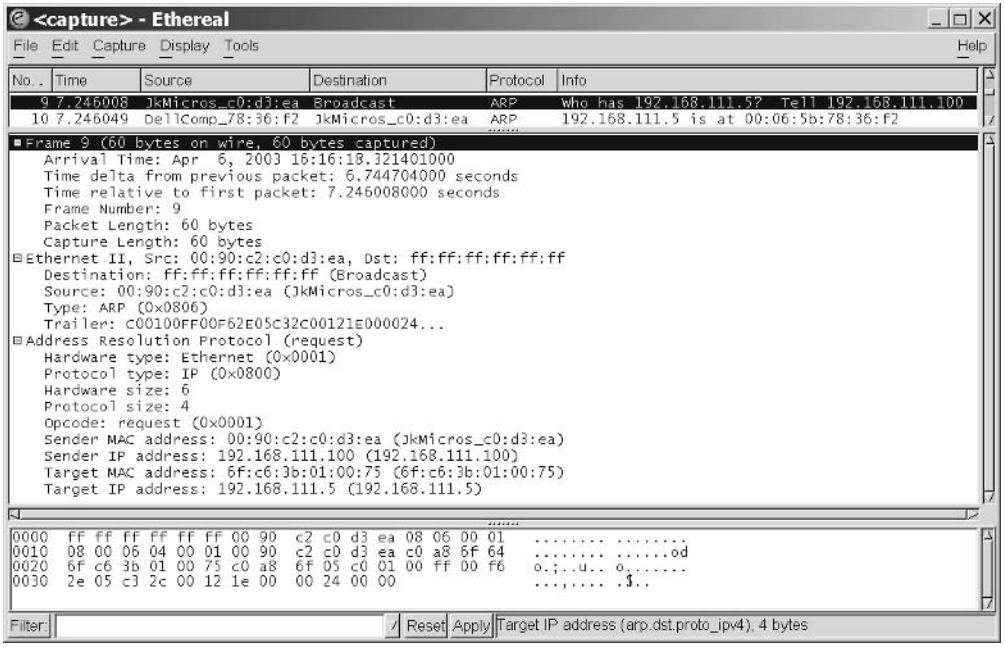

Figure 1-5: The Ethereal Network Protocol Analyzer decodes and displays Ethernet traffic.

The Ethereal Network Protocol Analyzer

The Ethereal Network Protocol Analyzer is a full-featured and free protocol analyzer from Gerald Combs and others, available from *www.ethereal.com*. You can use Ethereal to view the contents of Ethernet frames along with timing and other information. The software decodes data for over 300 network protocols. Figure 1-5 shows data captured by Ethereal during a response to an ARP request for a computer's Ethernet hardware address.

You can run Ethereal from any PC in the network you're monitoring. To ensure that the frames you want to see are visible, the PC running Ethereal should be in the same collision domain as the computer whose traffic you're monitoring.

Other Options

Ethereal is one of a variety of software-based protocol analyzers for Ethernet. Another option for viewing network traffic is to use a hardware analyzer that you plug into an available port in the network. An example is Agilent Technology's J6800A Network Analyzer. The J6800A is a portable system that contains an embedded PC running network analyzer software. You can control the analyzer remotely over a network or dial-up connection. A traffic generator enables you to specify traffic to place on the network. Two time-synchronized data acquisition systems enable comparing two locations at once.