PART 1

PROTOCOL ANALYSIS: A LOOK AT THE PLAYERS

In order to properly perform network monitoring and analysis, we need to understand what we are looking at. This is part of what keeps many from using these important tools. However, by looking at the more common protocols we are likely to run into and the common things associated with them, we will be able to understand what it is we are in fact looking at, and therefore be able to more effectively troubleshoot our networks.

CHAPTER 1

Basic Network Models

In this chapter we will look at the Open Systems Interconnection (OSI) Model and see how we can use it in our network monitoring activities. We will also look at modifications made to the OSI model by the IEEE 802 group. Then, we will look at how data makes it onto the wire, and finally, we will conclude with the role of Protocols in all this.

Network Models assist us in visualizing the way that computers talk to one another with our network monitoring tools. We will use them in our efforts to first understand the protocols and the data flow, and we will revisit them as we examine network optimization and the troubleshooting scenarios.

THE OSI MODEL

Our first step in network monitoring is to look at the OSI model. This will provide a framework for understanding the way protocols work. The OSI model was developed in 1984 by the International Standards Organization (ISO) to serve as a guide for network communication. It updated a 1978 specification the organization had developed to allow dissimilar equipment to talk back and forth to exchange data while using similar protocols. The 1984 OSI model is an international "classic" to which nearly all network architectures answer. While most implementations do not necessarily lay out cleanly like the OSI model does, the functions described by the model must be performed (or at least taken into consideration) in some manner. It is the description of activities, or functions, that make the OSI model so valuable as a trouble-

shooting tool. For instance by describing what a scenario is doing, "yes I can do this, this and this, but not that," you can isolate where exactly the problem is occurring.

The OSI model clearly describes functions and is often used by hardware manufactures as guidelines to incorporate features into their designs. In this manner we see how hardware and software work together to provide for reliable and efficient communication between devices. We will be referring back to this model again and again during the course of the book.

The OSI model is a layered approach to communication, and each layer provides services to the layer below and above it in the hierarchy. Figure 1–1 lists the seven layers of the OSI model.

The highest layer, layer 7, is normally depicted on top because it is where the applications reside and where the interface to the user may be. The lowest two layers, layers 1 and 2, are where the wire and network adapters reside.

We can use the telephone system as an analogy to the way data communication takes place in a networked environment. When I pick up the telephone and call the office, I am using services provided by the telephone company. The telephone uses the wire in my house, and goes out onto the local and, eventually, long-distance carriers. Neither the people in my office nor I am aware of the wires and equipment involved. It is as if we're talking di-

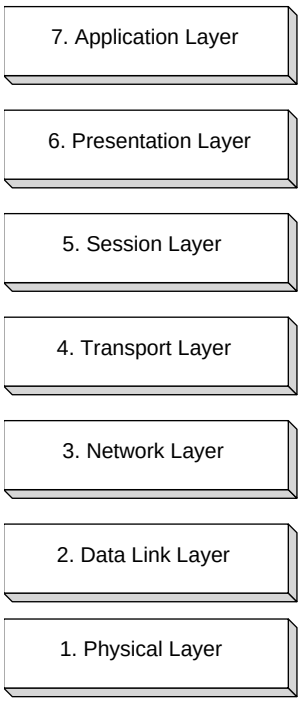

Fig. 1–1 The seven-layer Open Systems Interconnection model is the basis for both understanding and troubleshooting modern computer networks.

rectly to each other. The OSI model works in the same way. The application layer on one machine establishes a virtual conversation with the application layer at the other machine, and each does not know that the other layer is involved.

Figure 1–2 illustrates that each layer in the OSI model thinks it is talking to its counterpart on the other machine. In reality, each layer talks only to the layer above or below it in the hierarchy. Each layer is responsible for performing certain functions according to the protocol involved.

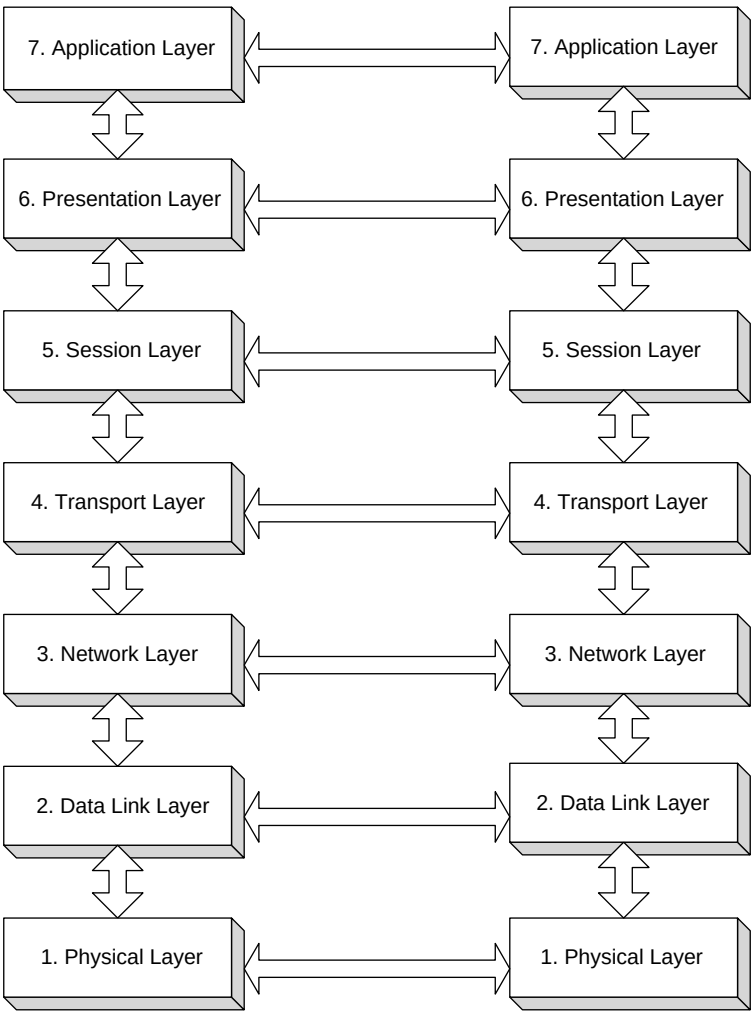

Fig. 1–2 The perceived communication, and the actual path of communication.

When an application wants to send data to another application, the data is passed from one layer to the next down the entire protocol stack. Each layer will add header information to assist it in performing its specific function. In some instances the layer may also add trailer information as well. The data is broken into pieces to assist it in its journey across the network. Data received from the TCP layer is called a segment, but as the data passes to the IP layer and its header information is added, it is called a packet or a datagram. Once the unit of data is on the wire flowing towards its destination, it is called a frame. The terms packet and frame are sometimes (although technically incorrect) used interchangeably. To be completely accurate, data received from the network layer should be referred to as a packet, and data received from the wire should be referred to as a frame. The data then passes from layer to layer, and each layer will add something or take something away from the packet according to where it is in the process. On the receiving end of the communication, the process occurs in the reverse; therefore, it is taken off the wire at the physical layer and moves back up the different layers until arriving at the application layer as it is sent by the other application. Only the physical layer talks to its counterpart on the other machine. Interaction between adjacent layers occurs through an interface that defines which services it will provide to the layer above or below in the hierarchy. With this basis of understanding, let us look in more detail at the services provided by each layer of the OSI model.

Application Layer

Layer seven, the application layer, is the highest level in the OSI model, and it serves as the window for applications to access network services. This layer directly supports user applications, but it is not where applications such as Word or Excel reside. They are not part of the OSI model. Applications that do reside here are Telnet, FTP, and the like. This layer handles general network access, flow control, and error recovery.

The application layer interacts with software applications that implement or support the communicating component of the network application. These network applications are things such as file transfer, e-mail, network management, remote access, and the like. Application layer functions typically include the following:

- Identifying communication partners—The application layer identifies and determines the availability of communication partners for an application with data to transmit.
- Determining resource availability—The application layer must determine whether sufficient network resources for the requested communication are available.

- Synchronizing communication—Communication between applications requires cooperation that is managed by the application layer.

The application layer is the OSI layer closest to the end user. That is, both the OSI application layer and the user interact directly with the software application. Some examples of application layer implementations:

- TCP/IP applications—TCP/IP applications are protocols in the Internet Protocol suite, such as Telnet, File Transfer Protocol (FTP), and Simple Mail Transfer Protocol (SMTP).
- OSI applications—OSI applications are protocols in the OSI suite such as File Transfer, Access, and Management (FTAM), Virtual Terminal Protocol (VTP), and Common Management Information Protocol (CMIP).

Presentation Layer

Layer six, the presentation layer, determines the format used to send data among computers. It can be called the network translator because it is involved in translating data from the format sent down from the application layer into a commonly recognized intermediate format. At the receiving computer, the presentation layer translates data back into the format in which the Application layer on the other computer had sent it. Some of the common activities performed by the presentation layer include protocol conversion, translating the data, encrypting the data, changing or converting the character set, and expanding graphics commands. The presentation layer also manages data compression to reduce the number of bits needed to transmit the data. The redirector, who is responsible for deciding where to obtain data for input or output operations to resources on a server, resides at the presentation layer.

The presentation layer provides a variety of coding and conversion functions that are applied to application layer data. These functions ensure that information sent from the application layer of one system will be readable by the application layer of another system. Some examples of presentation layer coding and conversion schemes follow:

- Common data representation formats—The use of standard image, sound, and video formats allow the interchange of application data between different types of computer systems (JPEG, MPEG, GIF, . . .).
- Conversion of character representation formats—Conversion schemes are used to exchange information with systems using different text and data representations (such as EBCDIC, ASCII and the ever popular Hypertext Markup Language (HTML) that describes how information

should appear when viewed by a web browser such as Internet Explorer or Opera).

- Common data compression schemes—The use of standard data compression schemes allows data that is compressed at the source device to be properly decompressed at the destination.
- Common data encryption schemes—The use of standard data encryption schemes allows data encrypted at the source device to be properly decrypted at the destination.

Presentation layer implementations are not typically associated with a particular protocol stack. Some well-known standards follow:

- Data: ASCII, EBCDIC, Encryption
- Visual Imaging: PICT, TIFF, GIF, JPEG
- Audio and Video: MIDI, MPEG, QuickTime

Session Layer

Layer five, the session Layer, allows applications on different computers to set up, manage, and conclude a conversation, called a session. This layer provides some of the things you would naturally think of as being needed at this level, such as name recognition and security. We need security to prevent anyone from being able to connect to the computer across the network. Some of the other things we would expect to see at this layer are tools to control how the machines communicate during the session. We also find data recovery options such as checkpoints between the communicating applications. This allows the machine to know what was sent and when. It is crucial this function is performed—particularly on unreliable networks (such as the Internet). In this way, if data is lost, then the only thing that must be retransmitted is the data lost since the last checkpoint. This layer also implements dialog control between the communicating processes. This regulates which side transmits, when, and for how long. Thus the session is in charge of managing all service requests and responses that occur between the applications running on two separate computers.

The session layer establishes, manages, and terminates communication sessions between presentation layer entities. Communication sessions consist of service requests and service responses that occur between applications located in different network devices. These requests and responses are coordinated by protocols implemented at the session layer. Some examples of session layer implementations are the following: AppleTalk Session Protocol, DEC SCP, NFS, SQL, RPC, and X Windows implementations.

Transport Layer

Layer four is the transport layer, which provides an additional connection level beneath the session layer and thereby defines end-to-end connectivity between host applications. The transport layer is responsible for ensuring that the packets are delivered error free, in sequence, and with no loss or duplication. This layer repackages messages received from the upper layers, dividing long messages into smaller packets when required, and gathering up several small messages and putting them into larger packages. This repackaging allows the packets to be transmitted more efficiently across the network. At the receiving end, the Transport layer reassembles the original messages and typically sends an acknowledgement back to the originator.

The transport layer is responsible for establishing end-to-end operations. This is a logical relationship. The functions provided by the transport layer are flow control, error handling, and problem solving related to the transmission and reception of packets. This is where we find the TCP (Transmission Control Protocol) part of TCP/IP. The transport layer implements reliable internetwork data transport services that are transparent to upper layers. Transport layer functions typically include the following:

- Flow control—Flow control manages data transmission between devices so that the transmitting device does not send more data than the receiving device can process. This is especially important on busy networks where many machines could flood the network with information destined for the same host. It would be very easy for packets to be dropped if there were not some way to control the congestion. When datagrams arrive, they are stored in memory in a buffer. If this gets overloaded before the data can be processed, TCP has the ability to send a not-ready indicator. This acts like a red light to allow the computer time to catch up. Once able to handle additional segments, the receiver sends a ready indicator and the senders can again resume transmitting.
- Multiplexing—Multiplexing allows data from several applications to be transmitted onto a single physical link. This is achieved through the layering approach of the OSI model. Different applications send segments on an as-received basis, and they can be directed to one or many destinations. When the data stream arrives at the destination hosts transport layer, it is reassembled and passed back up to the appropriate application. TCP uses port numbers to enable multiplexing. Since several different applications could conceivably use TCP (or UDP) at the same time, the transport layer protocols store an identifier in the header. This identifier is a 16-bit port number stored in the header. Both TCP and UDP use this to identify the source and destination port number. A list of well-known port numbers is included in Appendix A.

These well-known ports are assigned by the Internet Assigned Numbers Authority (IANA) and have the numbers between 1 and 1023 for various server services such as telnet, FTP, SMTP and the like. Numbers between 1024 and 5000 can be used for applications creating ephemeral ports (short lived ports). IPX/SPX uses a similar concept, although they are called sockets in that suite.

- Virtual circuit management—Virtual circuits are established, maintained, and terminated by the transport layer. This is a connection-oriented session. In order for data transfer to occur, both the sending and the receiving machine are involved. You cannot send data to a machine at this layer without the involvement of both machines. The initiating machine will send a synchronization message to begin the transfer, the receiving machine acknowledges the synchronization message and includes a synchronize sequence request, and once this is acknowledged the data transfer begins.
- Error checking and recovery—Error checking involves various mechanisms for detecting transmission errors. Error recovery involves taking an action (such as requesting that data be retransmitted) to resolve any errors that occur. TCP and SPX require a positive acknowledgment of the receipt of the data that has been transmitted. When data is transmitted, a timer is started. If an ACK is not received before the timer expires, then the segment is resent.

Network Layer

Layer three is the network layer, which is responsible for addressing messages and translating logical addresses and names into physical addresses. This layer also decides the route from the source to the destination computer. It determines which path the data takes in order to arrive at the destination computer, based on such factors as traffic congestion, priority of service, routing and the like. In addition to these functions, if the network adapter on the router cannot transmit a data chunk as large as the source computer sends, then it will compensate for it by breaking the data into smaller units. On the destination end, the receiving computer will reassemble the data as required. The Network layer manages device addressing and tracks the location of devices on the network. As a result, at this layer we typically find routers. The IP (Internetwork Protocol) performs Layer 3 functions.

The network layer provides routing and related functions that allow multiple data links to be combined into an internetwork. This is accomplished by the logical addressing (as opposed to the physical addressing) of devices. The network layer supports both connection-oriented and connectionless service from higher-layer protocols.

Data Link Layer

Layer two is the data link layer, which is responsible for sending data frames from the network layer to the physical layer. On the receiving end, it packages the raw bits received from the physical layer into data frames. A data frame is an organized, logical structure in which data can be placed. Figure 1–3 shows a typical data frame. In it we see the destination address, the sender address, the frame type, length of both the frame and data, as well as how much information is remaining.

The data link layer is responsible for providing the error-free transfer of these frames from one computer to another through the physical layer. One method it uses is the CRC (Cyclical Redundancy Check) that represents error correction and verification information to ensure that the data frame is received properly. It is in effect a magic number that lets the receiving computer know the packet is valid. This allows the network layer to assume virtually error-free transmission over the network connection. In general, the data link layer sends a frame and then waits for an acknowledgement from the recipient. The recipient's data link layer detects any problems with the frame that may have occurred during transmission. Frames that are not ac-

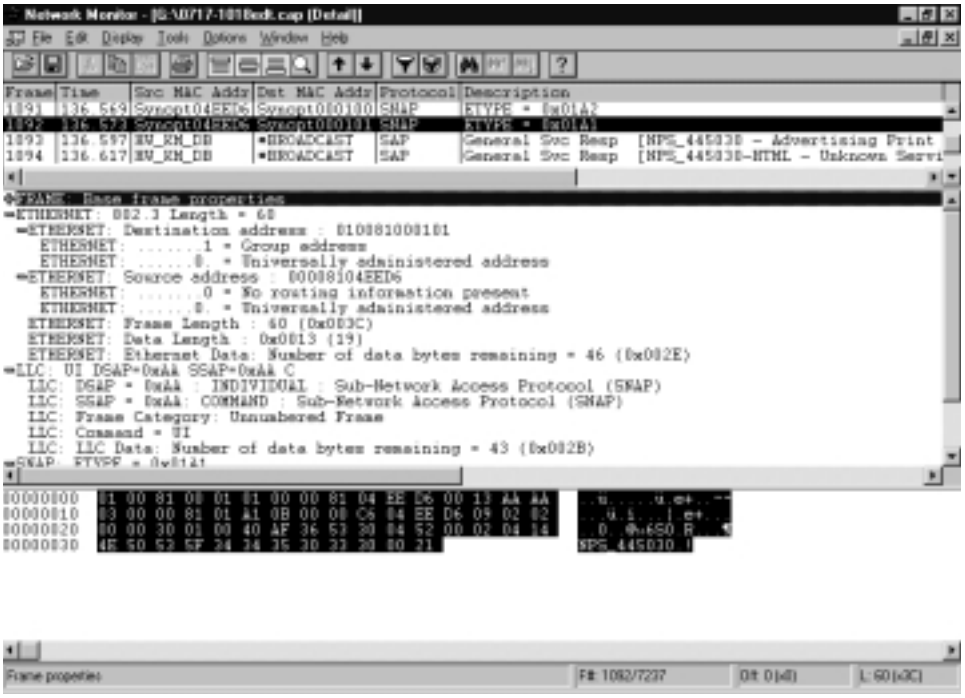

Fig. 1–3 A typical data frame includes destination- and error-checking information added at the data link layer.

knowledged, or frames that are damaged, are retransmitted. Thus the data link layer is responsible for notification, network topology, and flow control.

The data link layer provides access to the physical media and ensures reliable transit of data across a physical network link. This is where Ethernet, Token Ring, and FDDI media access methods are defined. Each of the different data link layer specifications define different network and protocol characteristics, including the following:

- Physical addressing—Physical addressing (as opposed to network addressing) defines how devices are addressed at the data link layer.
- Network topology—Data link layer specifications often define how devices are to be physically connected (such as in a bus or a ring topology).
- Error notification—Error notification involves alerting upper layer protocols that a transmission error has occurred.
- Sequencing of frames—Sequencing of data frames involves the reordering of frames that are transmitted out of sequence.
- Flow control—Flow control involves moderating the transmission of data so that the receiving device is not overwhelmed with more traffic than it can handle at one time.

Some of the physical devices that typically operate at layer two include bridges and switches (although some of the more intelligent switches now are capable of operating at layer three).

Physical Layer

Layer one is the physical layer that transmits the data stream between the machines. As the name implies, this layer incorporates the means to connect the two machines together. It is related to the electrical, optical, mechanical media, procedural, or the interfaces required to make the connection whether it is physical wiring, infrared, laser, or other methods. The physical layer is responsible for carrying the data that has been generated by the higher layers.

This layer also defines how the cable is attached to the network adapter card, for instance, the number of wires used and the like. It is responsible for carrying the bits (the ones and zeros) from one computer to another. The bits do not have any meaning at the physical layer; instead they are merely ones and zeros, electrical pulses, or flashes of light. Layer one is responsible for ensuring that when a one is transmitted, it is received as a one and not as a zero. This layer also defines what a one is, how long the pulse lasts, how long it has to be, how strong, and the like and so pays attention to the voltage levels, timing of voltage changes, physical data rates, and maximum distances each can travel. This is where we find devices such as repeaters.

THE IEEE 802 PROJECT

Around the same time the ISO was working on the OSI model, the Institute of Electrical and Electronics Engineers (IEEE) worked on LAN standards as well. These standards became known as the 802 project named from the year and month it began (February 1980). As these models were emerging at around the same time, and since the two organizations shared information between them, the two standards are compatible. The 802 project defines the standards for the physical components of a network—the interface cable and the cabling—that are contained in the first two layers of the OSI model, the physical and the data link layers. These standards contain specifications for Network Interface Cards, Wide Area Network components, and components used to create both coaxial and twisted pair networks. There are twelve categories in the 802 project, which are listed in Table 1–1.

Enhancements Made to the OSI model

The 802 project split the data link layer into two parts: the Logical Link Control (LLC) layer and the Media Access control layer. In Figure 1–4, we can see how the 802 project achieves a more granular approach to the standards by implementing the data link layer subsections.

Logical Link Control Layer (LLC)

Table 1–1 802 Project

802 number	Title
802.1	Internetworking
802.2	Logical Link Control
802.3	Carrier-sense Multiple Access with Collision Detection (CSMA/CD) LAN (Ethernet)
802.4	Token Bus LAN
802.5	Token Ring LAN
802.6	Metropolitan Area Network (MAN)
802.7	Broadband Technical Advisory Group
802.8	Fiber-Optic Technical Advisory Group
802.9	Integrated Voice/Data Networks
802.10	Network Security
802.11	Wireless Networks
802.12	Demand Priority Access LAN, 100BaseVg-AnyLan

The LLC sub layer manages data-link communication between devices on a single network link. The LLC defines the use of logical interface points, which are called service access points (SAPs) that allow higher-level protocols to use the data link. Other computers can also refer to and use SAPs to transfer information from the LLC sublayer to the upper OSI layers.

Because the LLC resides on top of the other 802 protocols, we have great flexibility in that upper level protocols can now operate without worrying

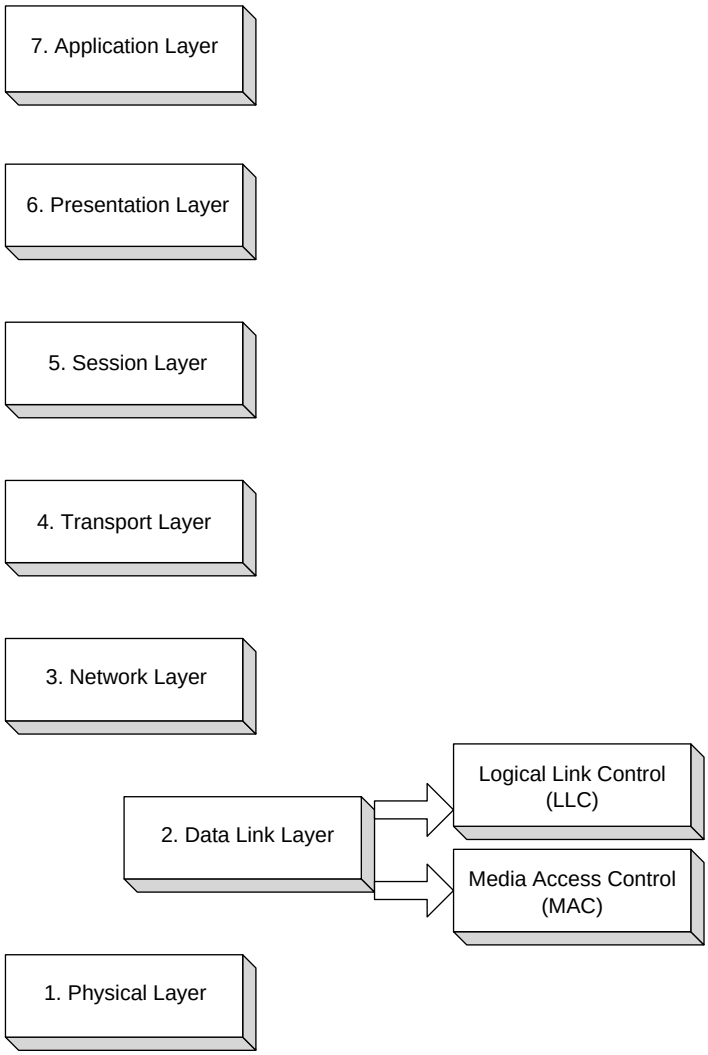

Fig. 1–4 The IEEE 802 project split the data link layer into two sublayers to provide a finer definition of media access methods.

about what type of media we have below. The media access control layer, however, is media-dependent and tied to specific 802 types.

Two service access points (SAPs) are defined in 802.2: the Destination Service Access point (DSAP) and the Source Service Access Point (SSAP). LLC provides support between applications for flow control, through the use of ready-or-not bits and sequence control bits. Figure 1–5 illustrates how these two service access points show up in a trace in Network Monitor.

Media Access Control Layer (MAC)

The lower layer of the two sublayers is the MAC layer, which provides access for the computer's network adapter card to the physical layer. The media access control layer communicates directly with the network adapter card and is responsible for delivering error-free data between computers on a network. Specific media access methods are detailed in 802.3,4,5 and 802.12. In this book we will focus on the access method detailed in 802.3, the Carrier-sense Multiple Access with Collision Detection (CSMA/CD) LAN (Ethernet).

In order for multiple machines to share access to the physical layer in an efficient manner, they need to be able to identify each other with unique addresses. The most important functionality provided by the MAC layer is the unique hardware address, called the MAC address. Every LAN interface must have a MAC address. The MAC address is a 48-bit address comprised of 12 hexadecimal digits. The first six hex numbers are the vendor code (also known as the organizational unique identifier (OUI)). These numbers are administered by the IEEE and can be very useful to network administrators trying to find drivers for a generic Ethernet card, or when sniffing the network. The last six hexadecimal numbers are generally assigned by the manufacturer, and are often burned into the ROM of the device.

A LOOK AT HOW DATA MAKES IT ONTO THE WIRE

Fig. 1–5 The two service access points defined in 802.2 are clearly visible in Network Monitor.

In order to transmit large files in a reliable manner across networks, the files must be broken into pieces called packets. Packets are the basic unit of network communication. Because data is divided up into packets, traffic congestion is reduced, and communication speed is increased as more computers have greater opportunities to access the wire. Special information is entered into each packet that enables the computer to break the data into pieces, put it back together, and check for errors once reassembled. In fact, packets may contain many different types of data such as messages, files, control data, commands or service requests, session control codes, or error messages.

All packets, regardless of the protocol, contain basically the same kinds of information in order to transmit from one computer to another. These are the source address of the sending machine, the destination address of the intended recipient (or recipients), the data itself, the instructions to various networking components it may meet along the way, and some kind of error checking and sequencing information.

Because of this need to contain basic kinds of information, all packets are comprised of at least three basic sections. These sections are the header, the data, and the trailer. In the header you will find information such as the source address, the destination address, what type of frame it is, and how much data it contains. The next section contains the actual data itself, the size of which varies depending on the type of network and the communication protocols being used. For instance, an Ethernet frame has a data load with a minimum size of 46 bytes and a maximum size of 1500 bytes, but that includes the TCP header and the IP header. When you remove the overhead, you are left with a maximum data size of 1460 bytes. The trailer contents will vary depending again on the protocol used, and the access method; however, you will typically see some kind of cyclical redundancy check (CRC) error-checking information. The CRC is one of those magic numbers that is computed by looking at the size of the packet and running it through some algorithm. This is a repeatable calculation, so when the packet arrives at the destination, the receiving computer can look at the CRC number, perform the calculation on the packet, and then compare the CRC number it obtains from its calculation with the one reported in the trailer of the packet. If they match, then the packet is presumed good and passed along up the OSI model to the next layer. If the CRC number computed by the receiving computer does not match the one reported in the packet, then the protocol will report the packet as bad and send a request for a retransmission of the packet.

The Packet Creation Process

How do our network models fall into this process, and what role do they play? Each layer of the OSI model will make its contribution to the format of the packet in the following manner. The packet creation process begins at the application layer. When data generated there needs to be transmitted to an-

other machine on the network, the application layer will add a little header information. From there it goes to the presentation layer, which will add some header information and then pass the packet to the session layer, which also adds some header information. It then goes to the transport layer, where the original data gets broken into the actual packet. Therefore, the actual format of the packet is dependant on the type of transport protocol being used. For instance, the transport protocol used in the TCP/IP protocol stack is the transmission control protocol. The transport control used in the IPX/SPX protocol is the sequenced packet exchange protocol. Each of these transport protocols handles packets a little differently, and we will look at them in more detail in later chapters. Basically, though, after the transport protocol breaks the data into packets, it must add some kind of sequence number to let the receiving computer know how to put the packets back together to reassemble the data. This will happen at the transport level of the receiving computer.

After the transport protocol has broken the data into packets, it is passed down to layer three in the OSI model, the network layer. Here again, specific actions are dependant on the protocol stack being utilized. In the TCP/IP stack, the network protocol is the Internetwork Protocol, and in IPX/SPX, it is the internetwork packet exchange protocol. In general, however, the network layer is responsible for adding routing information, protocol specific source and destination information, checksum information, and timing information as well.

The data link layer also will add its header and a CRC trailer. Finally, the packet arrives at the physical layer ready to make its journey to its destination, and it has not only the data payload, but also the header and trailer information added by the six upper layers to ensure timely and accurate delivery of the data.

On the receiving end, the packet proceeds from the physical layer, up to layer seven—the application layer—with each layer stripping away the information added by the corresponding layer from the sending computer.

Packets are addressed to a specific destination. In most instances, the destination is a particular computer, either a server, or a client machine. Although all the traffic on the wire is visible to each network adapter card, the computer does not pull the data off the wire if it is not addressed to it. There are times when this is not the case. For instance, certain types of information are broadcast that are transmitted to everyone. In this case every network adapter will pull the broadcast off the wire and look at the packet. This is one reason we will look at ways to control broadcast traffic in Chapters 5 and 6.

There are other occasions when a network adapter may pull traffic not destined to itself off the wire, such as in the case of a router. In our examination of network traffic, several of our capture files have routers in them. In these packets, we must look closely to see what the real destination and origi-

nation actually are. Of course all of this has a particular implantation in an Ethernet network, and that leads us into our next section.

ETHERNET COMMUNICATION SPECIFICS

The MAC layer (lower level of OSI layer 2) defines how the computer will gain access to the wire and how it will transmit and receive data. In order to do this, it communicates with the physical layer (OSI layer 1) through a common interface. This interface is composed of basic services or functions that control the exchange of data between the MAC layer on two different machines. By working through the interfaces, the MAC layer is essentially independent of the underlying physical media. It can be copper, fiber, radio, laser, or infrared. The first of these helper interfaces is the physical line signaling. This is a sublayer of the MAC layer, and its main function is to control access to the wire.

Because this is Ethernet, the method used is carrier sense—that is, the computer will sense when the line is free. Before every transmission, a check is made to see whether the line is currently in use. However, there is no carrier signal present on an Ethernet network. So how does it sense activity? It detects that the line is not busy by a timing mechanism between frames of data. This is called the interframe gap, and it is 96-bit periods long (9.6 microseconds). The interframe gap is therefore the smallest permissible space between frames of data on an Ethernet network. The physical line signaling will also detect when the line is occupied, and whether a collision has occurred. These functions are implemented into a part of the Ethernet card called the transceiver. In the old days the transceiver was a separate device, but it has long (thankfully) been implemented as a chip on the Ethernet card. At times this device is referred to as a media access unit (MAU). The transceiver is responsible for four basic functions. These are the transmission of the electrical signals onto the wire, the reception of the signals from the wire, the determination as to whether the wire is busy, or free, and the observation of the data for collisions.

Data is placed onto the wire through a process called the Manchester code. In this process, the first half of the bit value contains the complementary value, while the second half contains the actual value. In this way, each signal placed onto the wire is unique. This even works when transmitting several packets of essentially identical information. In Figure 1–6 we see the period of time used to transmit the information on the wire. The amount of time it would take to change from 1 to 0 on the wire is referred to as a bit period.

Since there is no external clock on an Ethernet network, the Manchester encoding enables the transmitters and receivers to synchronize with one an-

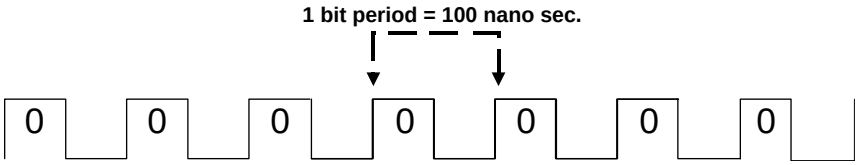

Fig. 1–6 The bit period for Ethernet is 100 nanoseconds.

other. The voltage on the Ethernet cable is between –2.2 volts and 0 volts, and it has a current between –90 mA and 0 mA.

The physical medium attachment performs several important tasks such as the transmitting and receiving data, identifying collisions, and controlling jabber. We will look at several of these briefly.

The Ethernet card must be able to take a parallel data stream and transmit it as a serial stream of data onto the wire. This is like driving down a multilane highway and coming upon a one-lane bridge. The transmit function cannot lose more than two complete bit cells. The delay between bit cells must be less than 50 nanoseconds. When the physical medium attachment receives the bit stream, it cannot lose more than five bit cells. So how does the MAC layer transmit data? Let's look at it in a little more detail.

First, the MAC layer receives the packet form the LLC. Next it adds the following information onto the packet: the preamble, start frame delimiter, destination address, source address, type field (for Ethernet), the length field, and data field with the protocol information form the higher layers. This is illustrated in Figure 1–7 below. If the data is less than 48 bytes, then the data field is padded with zeros to the minimum length.

After the data packet is generated, the transmission module calculates the CRC and puts this into the CRC field. The entire packet is transferred to the transmission MAC module. At this point the wire is checked for activity. If the wire is free, then MAC module waits for the interframe gap (9.6 microseconds) and then transmits. While transmitting, the MAC module checks to see whether a collision is detected. If there are no collisions, then transmission completes, and the next frame can be sent. If a collision is detected, then the collision function is called.

The collision function is used to detect collisions on the wire. When two computers transmit at the same time, the combined voltage of the resulting collision exceeds the amount normally on the wire, and the detecting machine will generate a collision signal. This will last for nine bit periods, or 900 nanoseconds, at which time, the machines attempting to use the wire will retransmit their packets at a random interval to avoid an additional collision. If this fails 16 times in a row, then an error is returned to the higher-level protocols.

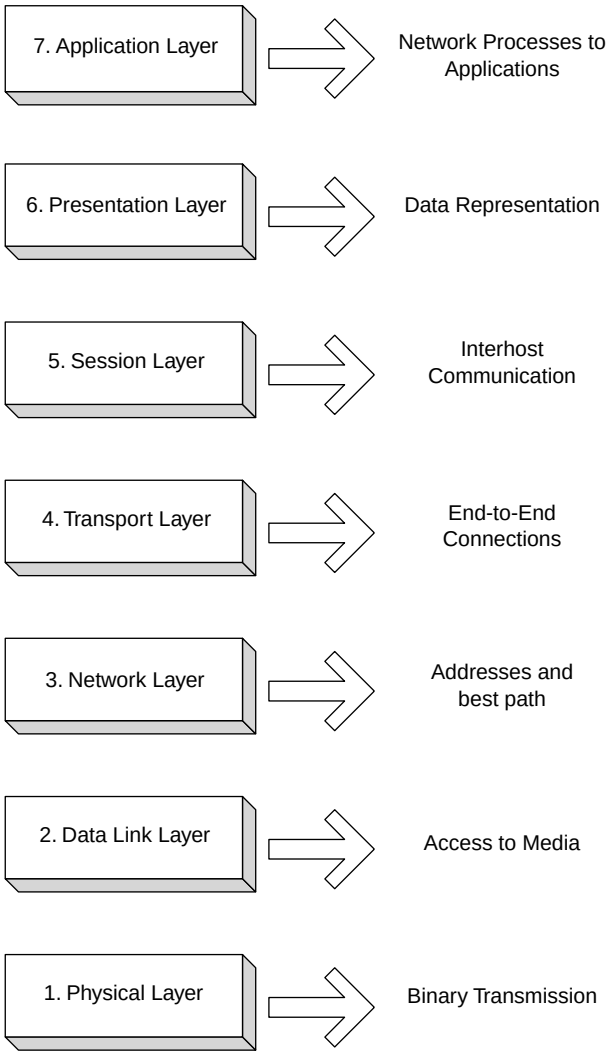

Fig. 1–7 The MAC layer adds header information onto the data packet.

The jabber function was put into the 802.3 standard to prevent one machine from monopolizing the wire. Each machine can use the wire for no more than 30 milliseconds at a time before it must allow another machine the chance to use the wire. If the machine does not cease transmitting during that period, the card assumes it is jabbering, and it should cease transmitting.

The MAC layer is also in charge of receiving data from the wire. All traffic is visible on the wire, but only packets addressed to the local machine are processed by the MAC layer. If it is not addressed to the local machine, then

the data packet is simply discarded by the machine. However, once it is received off the wire, then the first thing to do is to check it for errors. This is done by the MAC module verifying the CRC value. If the packet fails this initial check, then it is discarded as trash. Once the packet has passed the CRC check, the MAC layer begins the process of stripping off the preamble, the start frame delimiter, the destination address, the source address, the type field, and any padding that may have been added to the data field to make the packet meet the minimum size requirement. After this has been done, two more checks are performed against the packet. First the MAC layer will check to make sure the data field has the correct length and is a multiple of eight bits. If both of these additional checks are passed, the data is declared intact, and passed up the protocol layers. If the packet fails either of these checks, it is declared trash, discarded into the bit rubbish heap, and the process has to start over again.

What Is the Role of Protocols in All This?

You can think of protocols as languages. For instance, if I went to France and spoke French, I would get along just fine. But if I went to France and spoke Greek, I might have problems communicating unless I found someone who also spoke Greek. In the same way, protocols need to match between two computers wishing to communicate. Just as there are rules for the way in which languages function—for example subject and verb agreement, each of which must match the language—there are rules for the way that protocols work.

There are commands that the protocols support, and each also has a specific purpose. Just as with the English language, in which there is a difference between spoken English and written English, there are also differences in the way the protocols function. As there are many different languages, there are many different protocols. As certain languages work in certain countries, certain protocols work at different levels of the OSI model and are used for certain purposes. When several protocols work together at various levels of the OSI model, then they are referred to as a protocol stack. A network works at all levels of the OSI model. In the same way, a protocol stack will have protocols that work at different levels of the OSI model as well.

Protocol Stack

Protocols govern how communication takes place over the network between two or more computers. There are many different protocols that can be used to manage the exchange of information between machines on a network. The protocols define how they are going to interact, such as how and when a user is allowed to send, how errors are reported, and how much information can be sent at one time. If it were not for these rules, communication across a network would be impossible. Depending on which layer of the OSI model the protocol is designed to operate, it may be tasked with breaking down data

into small enough pieces to transmit over the wire to another computer. In doing this, it will have to know into how many pieces to break the data, in which order to transmit them, how to put them back together, and how to make sure they arrive in the correct order and in the same shape as when they were transmitted. It will also need to know the location of the machine where the data is supposed to go to, and how to get to the machine with the data. It then needs to know how to talk to the Ethernet card to get it actually put on the wire.

On the receiving computer, the protocol does some of the same things as the sending computer does, only in reverse. First, it must recognize the packet as addressed to the computer. Next it has to pull the data off of the wire and on into the computer through the Ethernet card. The protocols have to put the data back together in the same order they were in originally, and finally present the information to the application so that it can act on the information contained in the packet.

As we can see from the example above, there were many levels of the OSI model involved in this simple transaction. The application layer protocol talks to the application layer protocol on the other machine, the presentation layer talks to the presentation layer on the other machine, the session is responsible for establishing a session, the transport is responsible for getting the data to the other machine, and the internet layer finds the route to the destination and the LLC and physical layers.

As you might imagine, in order to be able to perform its tasks, each layer adds information to the basic packet of information for tracking purposes, so to speak. The sending machine adds as the information flows to layer one from the layer seven applications. As the receiving machine picks the data off the wire, at layer one, it will strip the added information off each layer on its way back up to the application layer. The presentation layer then presents the reassembled data back to the application layer in a reusable format.

One thing to keep in mind about protocols is that they have to be designed to route. While nowadays that is not really an issue, in the past it was. For instance, NETBEUI is a fast protocol good for small workgroups, but it does not route, and therefore is not suited very well for an Enterprise solution. Of course, there are other reasons one would not want to use NETBEUI on a WAN, but we will look at those reasons later. When data has to travel from one LAN to another LAN across a router that ties them together, there may be several ways to get there. Determining how to get there is the function of layer three protocols.

A Layered Approach

There are many tasks that must be performed in order for network communication to take place. As we see many of these activities take place on the network, it is important to understand what is going to aid our troubleshooting efforts. Our protocol stack is a layered approach. There are at least four reasons

for using a layered approach. They reduce complexity, standardize interfaces, facilitate modular engineering, and ensure interoperable technology.

The information has to be prepared, transferred, received, and acted upon. The different protocols that handle these chores form the discussion for the rest of the chapter. The way they work together to handle the task of transferring data from one machine to another is called layering. A protocol stack is a combination of protocols, and each layer in the stack has a different function that is handled by a different protocol—each with rules that govern the way it behaves in various situations. Each layer has its own duties. As we see in Figure 1–8, each layer performs certain duties, tasks and responsibilities.

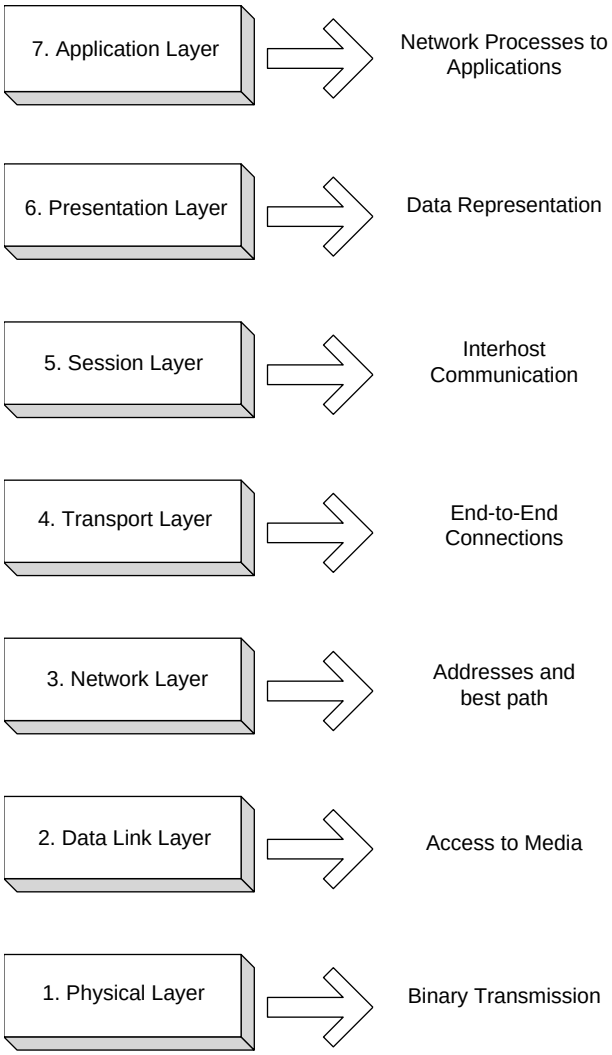

Fig. 1–8 Each layer does its own job, blithely ignoring the existence of the other protocols on the team.

Each layer talks to the protocol in the layer above and below it in the stack but does not know about the other protocols. In this way, protocols are able to achieve virtual communication between layers on different machines. For instance, the application thinks it is talking to the application layer on another machine. Little does it know it is only talking to its own presentation layer. Who says computers are so smart?

The lower layers of the model specify how equipment manufacturers enable their products to connect to the network and to the computers with which they are associated. The upper layers determine how applications will talk to each other, the rules for their communication, and error correction. The higher in the protocol stack, the more sophisticated the tasks and protocols become.

So How do I Tie All This Together?

All these things get tied together in a process called binding. This gives us a great deal of flexibility in that now the protocol and the network card adapter are independent, and therefore we can change a protocol stack without also having to change a driver for the network adapter. We can also change media access methods without having to change protocols as well. Therefore, we can run TCP/IP over Ethernet, token-ring, or whatever method we choose to implement. In addition to the above, if there are multiple network adapters, we can bind the protocol to one or all of the network adapters as we see fit. It simply does not matter how we do it, at least not from the protocol stack point of view. We may, however, need to pay attention to the binding order of the protocols, and we will look at that next.

The binding order is the order in which the computer will try a protocol in an attempt to communicate with other machines on the network, or in the world. Thus the order in which the protocols are bound to the network adapter card determines the order in which they will attempt to communicate with others. This order can be manually determined in Windows NT as seen in Figure 1–9.

The binding process also carries over to more than just the protocol stack to the network adapter; for instance, the Ethernet card driver must be bound to the Ethernet card, and the transport protocol must also be bound to the NetBIOS session layer above it in the stack.

There are several protocol stacks that have been developed over the years, some of which are listed below:

1. The ISO/OSI protocol suite
2. IBM Systems Network Architecture (SNA)
3. Digital DECnet

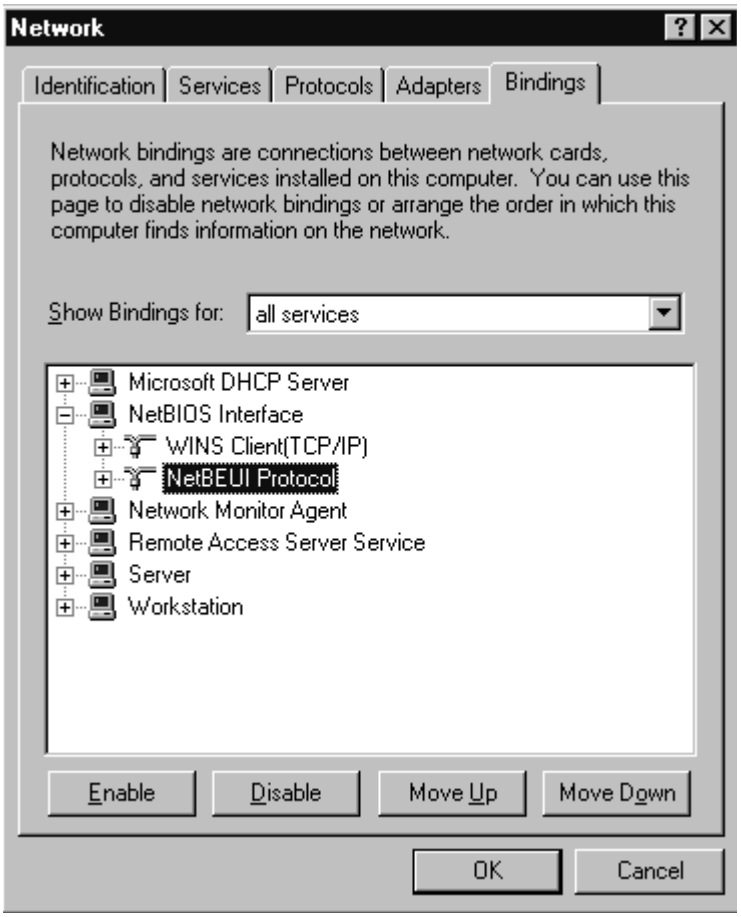

Fig. 1–9 On Windows NT machines, the binding order can be manually configured for optimal performance.

4. Novel Netware
5. Apple AppleTalk
6. The Internet protocol suite, TCP/IP

In each of these stacks, there are specific protocols that reside at the different layers in the OSI model. They may overlap one or more layers in the suite, but their functionality will be present. It may be helpful for us at this time to group some of these levels of functionality together into areas of service as seen in Table 1–2. There are three areas of function: application, transport and network.

Table 1–2 The OSI model provides three essential services

Layer	OSI layer	Functionality provided
Seven	Application	Application level, network services
Six	Presentation layer	Application level, network services
Five	Session layer	Application level, network services
Four	Transport layer	Transport services
Three	Network layer	Network services
Two	Data link layer	Network services
One	Physical layer	Network services

Application Protocols

Now we can talk about application level protocols, transport protocols, and network protocols. The application level protocols incorporate the top three layers of the OSI model, that is, the application, presentation, and session layers. They provide application to application interaction and data exchange. Some of the application level protocols we may find include those listed below:

1. APPC (advanced program-to-program communication), IBM's peer-to-peer SNA protocol, used mostly on AS/400's.
2. FTAM (file transfer access and management), an OSI file access protocol.
3. X.400, a CCITT protocol for international e-mail transmissions.
4. X.500, a CCITT protocol for file and directory services across several systems.
5. SMTP (simple mail transport protocol), an internet protocol for transferring e-mail.
6. FTP (file transfer protocol), an internet file transfer protocol.
7. SNMP (simple network management protocol), an internet protocol for monitoring networks and network components.
8. Telnet, an internet protocol for logging on to remote hosts and processing data locally.
9. Microsoft SMBs (server message blocks) and client shells or redirectors.
10. NCP (Novell Netware Core Protocol) and Novell client shells or redirectors.
11. AppleTalk and Appleshare Apples networking protocol suite.
12. AFP apples protocol for remote file access.
13. DAP (data access protocol), a DECnet file access protocol.

Transport Protocols

Transport protocols reside at the fourth or middle layer of the OSI model. They provide communication sessions between computers to ensure that data is able to move reliably between computers. Popular transport protocols include those listed below.

1. TCP (transmission control protocol), the TCP/IP protocol for guaranteed delivery of sequenced data.
2. SPX (sequential packet exchange), the transport portion of the IPX/SPX protocol suite for sequenced data.
3. NWLink, the Microsoft implementation of IPX/SPX.
4. NetBEUI (NetBIOS extended user interface) establishes communication sessions between computers and provides the underlying data transport services.
5. ATP AppleTalk transaction protocol, NBP name-binding protocol communication session and data transport protocols.

Network Protocols

Network protocols provide what are called link services and make up the bottom three layers of the OSI model. These protocols handle addressing and routing information, error checking and retransmission requests. Network protocols also define rules for communicating in a particular networking environment such as Ethernet or Token Ring. The more popular network protocols include those listed below:

1. IP (internet protocol), the TCP/IP protocol for packet forwarding routing.
2. IPX (internetwork packet exchange), NetWare's protocol for packet forwarding and routing.
3. NWLink, the Microsoft implementation of the IPX/SPX protocol.
4. NetBEUI, a transport protocol that provides data transport services for NetBIOS sessions and applications.
5. DDP datagram delivery protocol, an AppleTalk data transport protocol.

We can see how the protocols map to the OSI model, and we now have an idea of the functionality of each. We do, however, want to see how all these things map together, and after that, we will begin to explore differences in the way each behaves, and hopefully begin to see how each is used, as well as the different levels of functionality achieved by each.

Connection-Oriented Network Service

Connection-oriented service involves three phases:

- Connection establishment—During the connection establishment phase, a single path between the source and destination systems is determined. Network resources are typically reserved at this time to ensure a consistent grade of service (such as a guaranteed throughput rate).
- Data transfer—During the data transfer phase, data is transmitted sequentially over the path that has been established. Data always arrives at the destination system in the order in which it was sent.
- Connection termination—During the connection termination phase, an established connection that is no longer needed is terminated. Further communication between the source and destination systems requires that a new connection be established.

Connection-oriented service has two significant disadvantages as compared to connectionless network service:

- Static path selection—Because all traffic must travel along the same static path, a failure anywhere along that path causes the connection to fail.
- Static reservation of network resources—A guaranteed rate of throughput requires the commitment of resources that cannot be shared by other network users. Unless full, uninterrupted throughput is required for the communication, bandwidth is not used efficiently.

Connection-oriented services are useful for transmitting data from applications that are intolerant of delays and packet resequencing. Voice and video applications are typically based on connection-oriented services.

Connectionless Network Service

Connectionless network service does not predetermine the path from the source to the destination system, nor are packet sequencing, data throughput, and other network resources guaranteed. Each packet must be completely addressed because different paths through the network might be selected for different packets, based on a variety of influences. Each packet is transmitted independently by the source system and is handled independently by intermediate network devices. Connectionless service offers two important advantages over connection-oriented service:

- Dynamic path selection—Because paths are selected on a packet-by-packet basis, traffic can be routed around network failures.

- Dynamic bandwidth allocation—Bandwidth is used more efficiently because network resources are not allocated bandwidth that they are not going to use.

Connectionless services are useful for transmitting data from applications that can tolerate some delay and resequencing. Data-based applications are typically based on connectionless service.

Data Link Layer Addresses

A data link layer address uniquely identifies each physical network connection of a network device. Data link addresses are sometimes referred to as physical or hardware addresses. Data link addresses usually exist within a flat address space and have a preestablished and typically fixed relationship to a specific device. End systems typically have only one physical network connection, and thus only one data link address. Routers and other internetworking devices typically have multiple physical network connections. They therefore have multiple data link addresses.

Media access control (MAC) addresses are a subset of data link layer addresses. MAC addresses identify network entities in LANs implementing the IEEE MAC sublayer of the data link layer. Like most data link addresses, MAC addresses are unique for each LAN interface. MAC addresses are 48 bits in length and are expressed as 12 hexadecimal digits: The first six hexadecimal digits are the manufacturer identification (or vendor code), called the organizational unique identifier (OUI). The IEEE administers these six digits. The last six hexadecimal digits are the interface serial number or another value administered by the specific vendor. MAC addresses are sometimes called burned-in addresses (BIAs) because they are burned into read-only memory (ROM) and copied into random-access memory (RAM) when the interface card initializes.

Network Layer Addresses

A network layer address identifies an entity at the network layer of the OSI reference model. Network addresses usually exist within a hierarchical address space. They are sometimes called virtual or logical addresses. The relationship of a network address with a device is logical and unfixed. It is typically based either on physical network characteristics (the device is on a particular network segment) or on groupings that have no physical basis (the device is part of an AppleTalk zone). End systems require one network layer address for each network layer protocol they support. (This assumes that the device has only one physical network connection.) Routers and other internetworking devices require one network layer address per physical network connection for each network layer protocol supported. For example, a router with three interfaces, each running AppleTalk, TCP/IP, and OSI, must have three

network layer addresses for each interface. The router therefore has nine network layer addresses.

Data Encapsulation

Each layer of the OSI model talks to the layer below and is dependant upon it to provide specific services and functionality. As each layer provides the service, it adds information to the data coming from the application in order to be able to transport the information to the destination machine. Of course, the specific information added depends upon the protocol stack in use, but following the general model, we see each layer add header information to the data, and in some cases also a trailer to the end of the data until we have a neatly formed packet that gets put onto the wire.

As we see in Figure 1–10, each layer adds header information to enable it to perform the service needed for the application. As data flows down the OSI model on the sending computer, this information is added. Once at the receiving computer, the header and trailer information is stripped away as the data is passed back up the OSI model.

There are five steps involved in the process of data encapsulation as it is prepared by the transport layer to be placed onto the wire. These are listed below:

1. Information from the user is received from the upper layers. It can be anything actually, from a logon request to a server, a print job, or surfing the web. Information is converted to data so that it can be transported to the destination.
2. The data is prepared for transport to the destination computer. In the case of TCP, a TCP header is added to the data that will include sequencing information to assist with keeping things in order as it is converted to Segments.
3. We are now at layer 3 in the OSI model, where the Internet protocol resides. Now we will add an IP header as we convert the TCP Segment into an IP packet. The IP header includes both source and destination IP addresses which will assist in routing (performed at this layer of the OSI model) the packet to the proper destination.
4. At the data link layer, IP packets are converted to Ethernet frames in our example. Each network device has to be able to put the packet into a frame so that the device can communicate over the local interface to another interface on the network. The frame type has to match or the devices will not be able to communicate. In our traces we will see that the Ethernet framing has been wrapped around our IP packet.

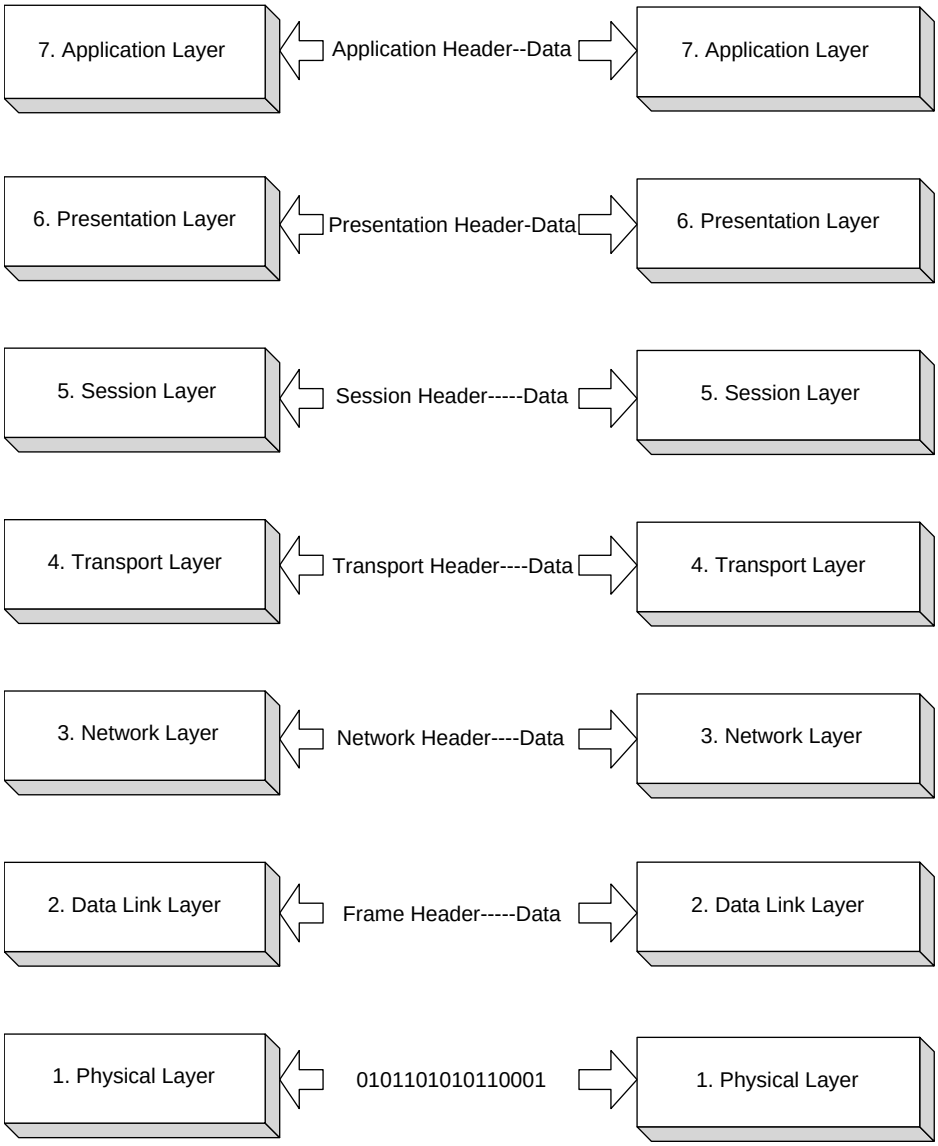

Fig. 1–10 Header information added at the sending computer is stripped off by the same layer at the receiving computer.

5. Now the Ethernet Frame is converted to ones and zeros. As we discussed earlier, these bits travel the network in a particular fashion defined by our access methods until they reach their destination.

IP over LAN Technologies

As we move forward with our troubleshooting skills, we need to be able to understand LAN encapsulations. LAN technologies encompass desktop technologies such as Ethernet, token ring, and arcnet and MAN technologies such as FDDI. In some technologies such as Ethernet, multiple encapsulations may exist, causing possible interoperability problems. In each of these technologies, the IP datagram needs to be delimited, addressed, and identified as an IP datagram.

Ethernet II When sent over an Ethernet network, IP datagrams use either the Ethernet II or the IEEE 802.3 SNAP encapsulation. The Ethernet II encapsulation uses the 2-byte ethertype field to denote upper layer protocols. The ethertype field is set at 0x08-00 to denote IP and 0x08-06 to designate ARP.

IP datagrams sent using an Ethernet II frame have a maximum size of 1,500 bytes and a minimum size of 46 bytes. IP datagrams under 46 bytes in length are padded with zeros to 46 bytes to preserve the Ethernet minimum frame size of 64 bytes (not including the preamble).

As we see in Figure 1–11, the Ethernet II frame is comprised of the preamble, and then the destination address, the source address, the ethertype field code, the payload, and finally the frame check sequence.

IEEE 802.3 SNAP The IEEE 802.3 SNAP encapsulation also begins with a preamble, a destination address, and source address; however, at that point the 802.3 SNAP diverges from the Ethernet II encapsulation. As we can see in Figure 1–12, we have a length field, and a sub-network access protocol (SNAP) header to encapsulate the IP datagram so that it can be sent on an IEEE 802.3 compliant network. The IEEE 802.3 header uses the defined value of 0xAA for both the destination service access point (DSAP) and the source service access point (SSAP) fields to indicate a SNAP frame. The control code is set to 0x03 to denote an unnumbered frame. The organization code field is set at 0x00-00-00 and the ethertype value of 0x08-00 for IP and 0x08-06 for ARP is the same as in the Ethernet II frame type.

IP datagrams sent using an IEEE 802.3 SNAP frame have a maximum size of 1,492 and a minimum size of 38 bytes. This is changed slightly from the Ethernet II size limitations and reflects the larger size of the 802.3 SNAP encapsulation overhead. IP datagrams under 38 bytes in length are padded

Fig. 1–11 The Ethernet II frame encapsulates an IP datagram.

with zeros to preserve the Ethernet minimum frame size of 64 bytes (not including the preamble and the start delimiter).

Ethernet II and IEEE 802.3 compared On a normal everyday network, it is indeed possible that both Ethernet II and IEEE 802.3 may be present on your network. How this works is that the internet protocol standards require all hosts to be able to send and receive Ethernet II encapsulated frames. In addition, most hosts are able to receive IEEE 802.3 encapsulated packets. Some hosts on the network may be able both to send and receive IEEE 802.3 packets. If this is the case, however, the default must be the Ethernet II type. Figure 1–13 has both the IEEE 802.3 and the Ethernet II packets, so we can get a good idea of the differences between the two encapsulation methods.

Ethernet II is the default for most networks. As we see in Figure 1–13, both frame formats have a six-byte (48 bit) destination and source address. This is the MAC address, the Ethernet address, or the hardware address of the host that is physically attached to the network. These addresses are resolved to the four-byte (32 bit) IP address by the address resolution protocol (ARP) that we will look at in the next chapter.

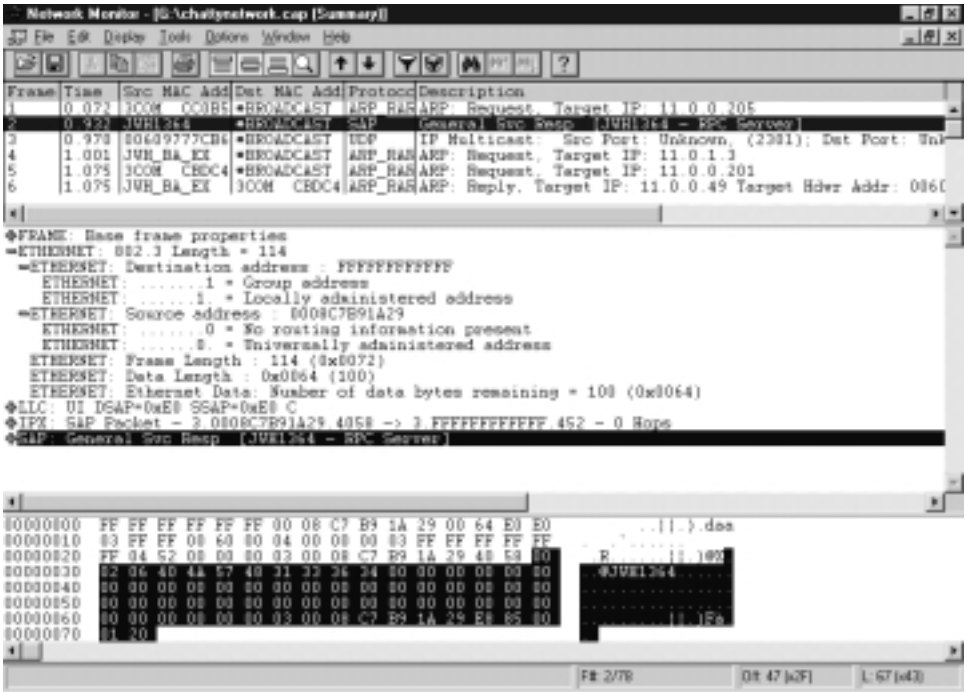

Fig. 1–12 The IEEE 802.3 SNAP encapsulation includes slightly more overhead than the Ethernet II frame.

In the Ethernet II header, the next two bytes are used to indicate the type (0x08-00 for IP, 0x08-06 for ARP). This is the same field that is found later in the 802.3 frame just before the start of the data portion of the frame. The IEEE 802.3 frame has, instead of the type, a two-byte length field. This has no correlation between Ethernet II frame and enables you to distinguish these two types of frame encapsulation methods. Following the type field in the Ethernet II header is the data portion of the frame with a size between 46 and 1500 bytes.

In the IEEE 802.3 encapsulation method after the destination and source addresses, we have a two-byte length field. This field is used to indicate how many bytes follow up to, but not including the CRC at the end of the frame. After the length field, we come to the section defined by the 802.2 chapter of the IEEE project. The first three fields in this section are the LLC fields. The DSAP and the SSAP fields are always set to AA, while the control field is always set to 03. In the next five bytes of SNAP data, the organization code is set to 0x00-00-00, which takes up three bytes. The next two bytes form the type field, which is the same as the type field found in the Ethernet II type field. As a result, it will be 0x08-00 for IP or

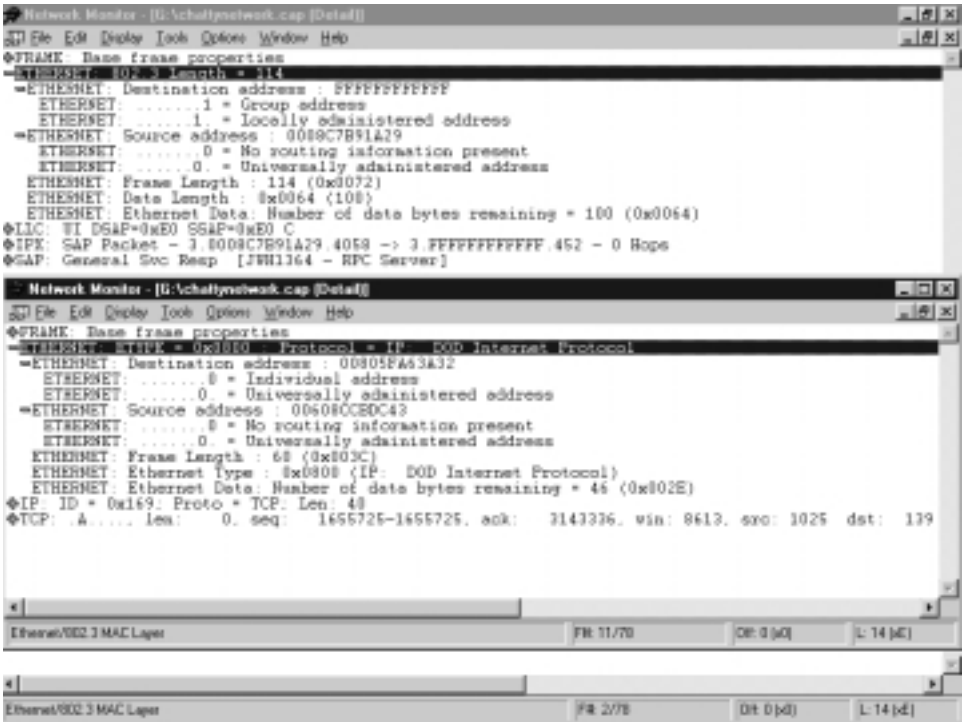

Fig. 1–13 IEEE 802.3 and Ethernet II compared.

0x08-06 for ARP or 0x08-35 for an RARP frame. Other types are listed in Appendix B.

The cyclic redundancy check (CRC) field is a checksum that is used to detect errors in the frame. This is also sometimes referred to as a frame check sequence (FRC).

The minimum size for IEEE 802.3 encapsulated frames and Ethernet II frames is the same. But because of the additional fields inserted before the data in the IEEE 802.3 type of frame (which takes up eight bytes), the minimum and maximum amounts for data load are eight bytes fewer than for the Ethernet II frame type.

Flow Control

Flow control is a function that prevents network congestion by ensuring that transmitting devices do not overwhelm receiving devices with data. There are a number of possible causes of network congestion. For example, a high-speed computer might generate traffic faster than the network can

transfer it, or faster than the destination device can receive and process it. There are three commonly used methods for handling network congestion:

1. Buffering is used by network devices to temporarily store bursts of excess data in memory until they can be processed. Occasional data bursts are easily handled by buffering. However, excess data bursts can exhaust memory, forcing the device to discard any additional datagrams that arrive.
2. Source quench messages are used by receiving devices to help prevent their buffers from overflowing. The receiving device sends source quench messages to request that the source reduce its current rate of data transmission for several reasons. Perhaps the most common occurs when the receiving device begins discarding received data due to overflowing buffers. When this happens, the receiving device begins sending source quench messages to the transmitting device at the rate of one message for each packet dropped. When these begin arriving at the source device, as each receives the source quench messages, it will begin lowering the data rate until it stops receiving the messages. Once the receiving device is no longer sending the source quench messages, the source device will gradually begin increasing the data rate as long as no further source quench requests are received.
3. Windowing is a flow-control scheme in which the source device requires an acknowledgement from the destination after a certain number of packets have been transmitted. With a window size of three, the source requires an acknowledgment after sending three packets. Let's look at how this works. The source device sends three packets to the destination device. After receiving the three packets, the destination device sends an acknowledgment to the source. The source receives the acknowledgment and sends three more packets. If the destination does not receive one or more of the packets for some reason (such as overflowing buffers), it does not receive enough packets to send an acknowledgment. The source, not receiving an acknowledgment, retransmits the packets at a reduced transmission rate.

Internetworking Functions of the OSI Network Layer

The internetworking functions of the network layer are responsible for selecting the best path through a network, establishing the network addresses, and communicating paths.

Routers use a routing protocol between routers, use a routed protocol to carry user packets, set up and maintain routing tables, discover networks, adapt to internetwork topology changes, use a two part address, and contain broadcasts.

WAN Services

X.25 X.25 was originally developed in the 1970's to provide dumb terminals with WAN connectivity across public data networks (PDN's). However, due to its flexibility and reliability, it has emerged as an international standard for sending data across PDN's.

A connection-oriented interface to a packet switched network (PSN) provides both error checking and guaranteed delivery of packets using either switched or virtual circuits. Due to its reliability, it is used for applications that require reliable transmission. A router would connect to a packet assembler/disassembler (PAD) on the X.25 network. The PAD is responsible for breaking down the messages into packets and then addressing them appropriately.

The X.25 specification maps to the physical, data link, and network layers of the OSI model. However, because the X.25 specification predates the OSI model, the layer names are different. For instance, the physical layer is called X.21 and specifies the electrical and the physical interfaces that can be used. Layer two is called the link access procedure-balanced (LAPB) protocol, which takes care of frame composition, flow control, and error checking. The packet layer corresponds to the network layer and is responsible for the set up and addressing of the virtual circuit.

Frame relay Frame relay is an industry standard, switched protocol that operates at the data link layer. It handles multiple virtual circuits using HDLC encapsulation between connected devices. Frame relay uses high-quality digital techniques, making error checking and flow control methods unnecessary. It is therefore more efficient and faster than X.25, the protocol for which it is generally considered a replacement. By using a simplified framing with no error correction, frame relay can send layer information very rapidly. It was originally conceived as a protocol to use over ISDN interfaces and was therefore developed as an independent protocol.

Within the frame relay PDN, the switching is implemented using a statistical rather than a time-division multiplexing method. With statistical multiplexing, available circuits form devices that are not currently allocated. Due to this dynamic characteristic, real time networks that are bursty in nature are ideal candidates for frame relay.

The management of the link is accomplished by using the local management interface. The LMI is responsible for establishing a link and monitoring the PVC's. Because the digital links are less susceptible to errors, frame relay uses only a CRC algorithm to detect bad data but not any mechanism for correcting the bad data. Frame relay relies on upper protocols for flow control over the link.

ATM Asynchronous transfer mode technology is a connection-oriented nonguaranteed delivery service. It scales very well and is used on both LANs and WANs. ATM is different from frame relay in that instead of breaking down messages into variably sized frames, all messages are broken into equally sized cells. Each cell has a five-byte header, and a 48-byte data field. Since all the cells are the same size, the switching can be done very fast, and therefore the need for buffering is eliminated.

Because it is an asynchronous method, we do not have the need for TDM techniques. With ATM, a station can send cells whenever it needs to, rather than waiting for the transmit turn to come up as with TDM.

Although ATM does not map directly to the OSI model, it does map basically to the data link and a network layer. As such, it can run over many different physical mediums including SONET and FDDI. The ATM adaptation layer maps to the OSI session and transport layers. It is responsible for receiving the data for the higher layer protocols such as IP and segmenting the data into the 48-byte cells to transmit over the ATM network.

ISDN Integrated services digital network is a communication protocol offered by telephone companies that permits telephone networks to carry data, voice, and other source traffic over telephone lines. ISDN is built on two main types of communications channels. The first is the Bearer (B channel) channel, which can carry voice, data, or images at a rate of 64 Kbps, and the second is the D (Data channel) channel, which is 16 Kbps and is used for control information, signaling, and link management data. ISDN is typically implemented in two versions: the basic rate ISDN, which is two B channels and a D channel; and the primary rate ISDN, which includes 23 B channels and a D channel.

HDLC The high-level data link control is a bit-oriented, synchronous data link layer protocol developed by ISO. Derived from SDLC, HDLC specifies a data encapsulation method on synchronous serial links using frame characters and checksums. Data is carried in frames that may contain variable amounts of data.

SLIP The serial line IP is used as a serial line encapsulation method and is therefore a simple packet framing protocol. SLIP defines a few characters that frame the IP packets onto a serial line. It is a minimal overhead technique that provides no addressing negotiation, protocol identification, error detection, or compression mechanism. Designed to be used over slow serial lines (such as modems), it works by defining two special characters that are used to provide framing.

The first one is the END character (0xC0), which is used to define the end of the IP datagram. The second character is the ESC character (0xDB), which is used to indicate when the 0xC0 character occurs inside the

IP datagram. The SLIP ESC character is different from the ASCII ESC character (0x1B).

SLIP uses a technique called character stuffing to prevent the occurrence of the END character in the middle of the IP datagram. If the END character (0xC0) occurs inside the original IP datagram, then it is replaced with the sequence 0xDB-DC. If the ESC character (0xDB) occurs within the IP datagram, then it is replaced with the sequence 0xDB-DD. This prevents the modem from hanging up in the middle of a transmission.

The maximum size of an IP datagram over SLIP is typically limited to 1,500 bytes on newer systems, or to 1,006 bytes on Berkeley UNIX version 4.2 implementations. However, this is a common practice, although is not specified in the standards.

To reduce the amount of overhead on potentially slow serial links, a method of compressing the IP and TCP headers into a 3- to 5-byte header is defined in RFC 1144 and is known as C-SLIP or Compressed SLIP.

PPP The point-to-point protocol is a successor to SLIP. PPP provides router-to-router and host-to-network connections over synchronous serial lines such as ISDN or SONET, as well as asynchronous circuits such as typical dial-up phone lines. It addresses the shortcomings of SLIP.

Point-to-point protocol is a family of protocols that provides multi-protocol encapsulation, a link control protocol (LCP) for establishing, configuring and testing the datalink connection, and a family of network control protocols (NCPs) for establishing and configuring different network layer protocols.

As we see in Figure 1–14, PPP sets a flag to 0x7E (01111110) in order to signify the start and end of a PPP frame. In successive PPP frames, only a single flag character will be set. The address field is used to address the packet to the destination node. On a point-to-point link, the destination node does not need to be addressed. Therefore, in PPP the address field is set to 0xFF, the broadcast address.

In an HDLC environment, the control field is used for a data link layer sequencing and acknowledgment function; however, in PPP the control field is set to 0x03 to indicate an unnumbered information (UI) frame. This is the same as the bit we saw set in our earlier discussion of Ethernet framing methods.

The protocol field is a two-byte field used to identify the protocol in the PPP payload. If the field is set to 0x00-21, it indicates an IP datagram. The frame check sequence (FCS) is a two-byte CRC similar to the one used for Ethernet encapsulation.

As in SLIP, PPP has a method to prevent the flag character from appearing in the middle of the data. On a synchronous link such as ISDN, we use bit stuffing to insert extra bits to mark when the flag character occurs. Bit stuffing means that a byte can be encoded as more than eight bits, but the extra bits are added or removed by the synchronous link hardware.

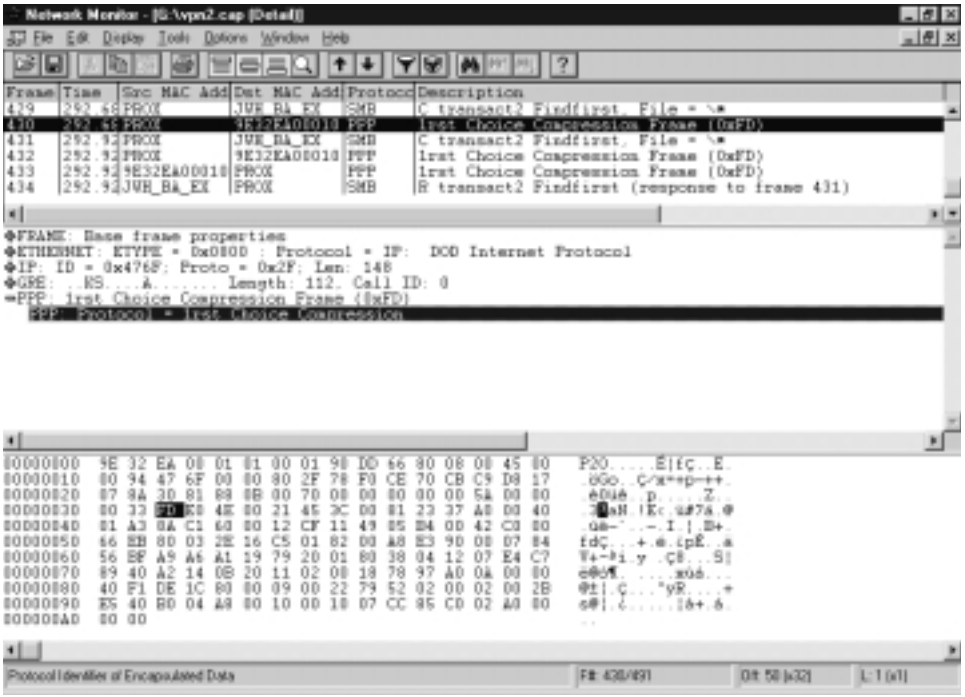

Fig. 1–14 The PPP protocol encapsulates IP for improved transmission over both synchronous and asynchronous circuits.

PPP also uses character stuffing (like SLIP) on asynchronous links to prevent the flag from appearing inside the PPP frame. If the 0x7E character occurs in the original IP datagram, it is replaced with the character string 0x7D-5E. If the ESC (0x7D) character occurs in the original IP datagram, it is replaced with the sequence 0x7D-5D back to 0x7D.

Characters fewer than 0x20 are also modified to prevent the serial drivers from interpreting them as control characters, by sending a 0x7D and then the original character with the sixth bit complemented.

The maximum receive unit (MRU) is 1,500 bytes.

PPTP Point-to-point tunneling protocol is a way to take an existing PPP frame and encapsulate it over an IP internetwork. PPTP allows the creation of secured virtual private networks (VPN) over the Internet. PPTP can be used whether or not the Internet service provider (ISP) is VPN-enabled or not. All that is needed is a server that supports PPTP and a client capable of making a PPTP connection.

The encapsulation of PPP frames is done using the generic routing encapsulation (GRE) protocol, which uses the IP protocol ID of 47 (0x2F). To use with PPTP, the GRE protocol was enhanced to provide an acknowledgment field.

As we see in Figure 1–15, the PPTP frame has a media header and an IP header before getting to the GRE header. After the GRE header comes the PPP header, and then the PPP payload.

The GRE header begins with two bytes for flags. These flags indicate whether a GRE payload is present, and whether sequence and acknowledgment numbers are present. The protocol type field is the Ethertype value that corresponds to PPP (0x88-0B). The payload length is the size of the GRE payload in bytes. Call ID is the session identification information for this PPTP packet, followed by the sequence number, and acknowledgement number, which is the highest sequence number received by the sending peer for this session. The acknowledgment number is used for flow control, not for retransmission of lost PPTP frames.

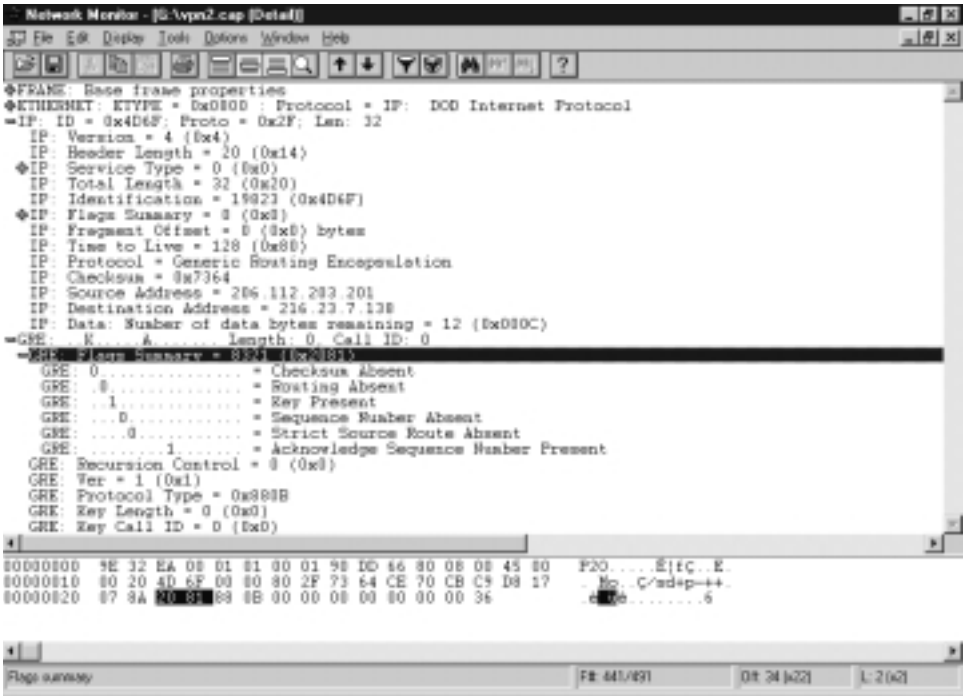

Fig. 1–15 The PPTP frame has several headers before getting to the GRE header information.

CHAPTER REVIEW

In this chapter we have covered much territory. We began by looking at the OSI model in which we looked at the level of functionality provided by each layer in the model. Next, we looked at the modifications made to the OSI model by the IEEE 802 project, and we saw how they split layer two, the data link layer, into two sections: the logical link control layer, and the media access control layer. These changes permit a finer bit of control over the lower layers allowing the logical link control to be media-independent since it sits on top of the MAC layer.

Next, we looked at how packets are formed, and we looked at some of the terminology involved such as IP datagrams, TCP segments, as well as Ethernet frames. After that, we took a pretty good look at how Ethernet actually communicates, how the carrier sensing works with Ethernet, how the multiple access is handled, and the methods of detecting collisions.

We then looked at the role of protocols and examined the concept of the protocol stack and the layered approach to network communications. We saw how binding works to enable us to change protocols without having to change network adapter drivers, and we looked at the concept of the protocol binding order as well.

Finally, we looked at how data encapsulation works and examined many of the common methods for encapsulation that are currently in use today. This was done by examining the way in which the headers work, and looking at each field in many of the more popular network interface layer protocols.

IN THE NEXT CHAPTER

We will begin our look at the TCP/IP protocol stack. We will see how the protocol suite maps to the various layers of the OSI model, and we will examine some of the protocols commonly implemented in a standard TCP/IP protocol stack.

Next, we will begin by looking at the transmission control protocol and the methods used by it to ensure reliable connection oriented communication. We will look at flow control methods and sequencing numbers. Following our discussion of TCP, we will look at its evil twin, the IP protocol.

In our discussion of IP, we examine the way IP encapsulates data in decent detail, as well as the IP header and the like. We look at some of the common commands used with IP and find hints we will use in our troubleshooting efforts later in the book.