2 NETWORK STANDARDS

LEARNING OBJECTIVES

By the end of this chapter, you should be able to:

- Provide the definitions of network; standards and protocols; message syntax, semantics, and order.
- Discuss message ordering in general and in HTTP and TCP.
- Discuss message syntax in general and in Ethernet frames, IP packets, TCP segments, UDP datagrams, and HTTP request and response messages.
- Explain how to encode application messages into bits.
- Explain vertical communication on hosts.
- Discuss major standards architectures: TCP/IP, OSI, and the hybrid TCP/IP–OSI standards architecture.

INTRODUCTION

We looked at network standards briefly in Chapter 1. In this chapter, we will look at standards at a more conceptual level, developing taxonomies of standards types. Much of the rest of this book focuses on specific standards; you will need to understand standards broadly to understand where those specific standards fit into the overall standards picture. This chapter also looks in some detail at the most important standards on the Internet and in corporate networks. These include Ethernet, IP, TCP, UDP, and HTTP.

Standard = Protocol

In this text, we use the terms *standard* and *protocol* to mean the same thing. In fact, standards often have *protocol* in their names. Examples are the Hypertext Transfer Protocol, the Internet Protocol, the Transmission Control Protocol, and the User Datagram Protocol.

Network Standards

WHAT ARE NETWORK STANDARDS? As Figure 2-1 illustrates, **network standards** are rules of operation that govern the exchange of messages between two hardware or software processes. To give a human analogy, in my classes, the standard language is American English. Not all of the first author's students are native English speakers, but we are able to communicate using a standard language.

In this chapter, we will see that network standards govern a number of message characteristics, including semantics, syntax, message order, reliability, and connection orientation.

Network standards are rules of operation that govern the exchange of messages between two hardware or software processes. This includes message semantics, syntax, message order, reliability, and connection orientation.

NETWORK STANDARDS BRING COMPETITION Standards are important because they allow products from different vendors to **interoperate** (work together). In Figure 2-1, an Internet Explorer browser from Microsoft works with an open-source Apache webserver. Although these software sources are very different and may not even like each other, their products work together because they exchange messages using the Hypertext Transfer Protocol (HTTP) network standard.

With network standards, it is impossible for any company to maintain a monopoly by refusing to allow others to use its proprietary communication protocols. Competition drives down prices. It also spurs companies to add new features so that their products will not be pure commodities that can only compete on price. These new features often appear in the next version of the standard.

Network standards are not only the key to competition. They are also the key to networking in general. To work in networking, you need to understand individual standards so that you can design networks, set up network components, and troubleshoot problems. Learning networking is heavily about learning standards. In this chapter, we will look broadly at the general characteristics of standards and will also look at some key network standards.

Recap of Chapter 1 Standards Concepts

In Chapter 1, we saw that standards can be described in terms of their layer of operation.

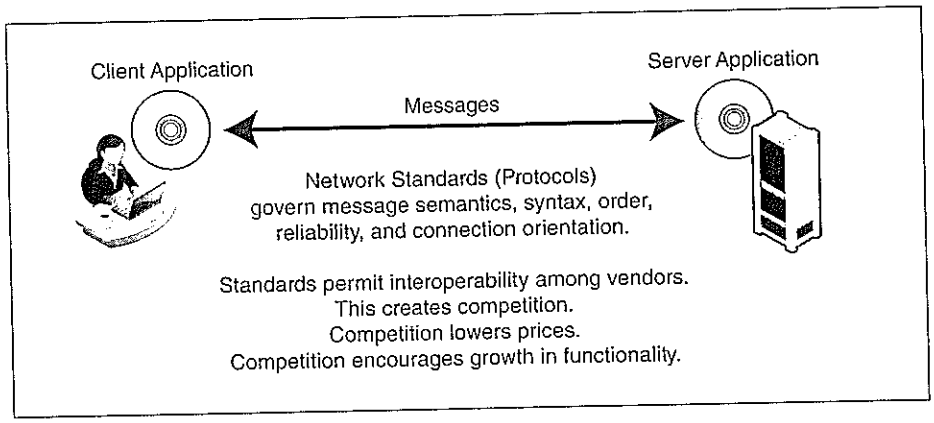


FIGURE 2-1 Network Standards

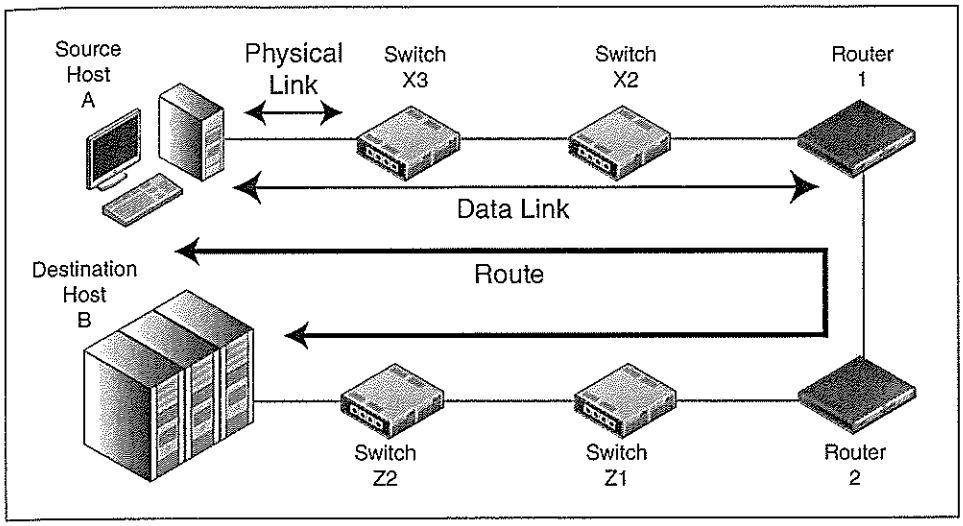


FIGURE 2-2 The Physical, Data Link, and Internet Layers

DELIVERY LAYERS As Figure 2-2 shows, three layers are involved in the transmission of packets between source hosts and destination hosts.

- Physical links are connections between adjacent devices, such as a host and a switch, two switches, two routers, a host and a switch, and so forth. Physical layer standards are not concerned with messages. Their job is to turn the bits of data link layer messages (frames) into signals.
- Data link layer standards govern the transmission of frames between two hosts, two routers, or a host and a router *across a single switched or wireless network*. The path that a frame takes is called its data link. This layer governs switch operation and frame organization.
- Internet layer standards govern the transmission of packets from the source host to the destination host, across multiple networks in an internet. The path that a packet takes between the two hosts is called its route. This layer governs router operation and packet organization.

A common source of confusion is that concepts are repeated at the data link and internet layers but with different terminology. This occurs because internetworking required the adding of a second layer of standards to those needed for transmission through single networks.

Also, recall that packets are carried inside frames. When a source host sends a packet to a destination host, the packet travels within a frame in each network along the way. If there are 19 single networks on the route between the source and destination hosts, a single packet will travel in 19 different frames.

THE TRANSPORT AND APPLICATION LAYERS The physical, data link, and internet layers are for standards that move packets along their way between the source host and the

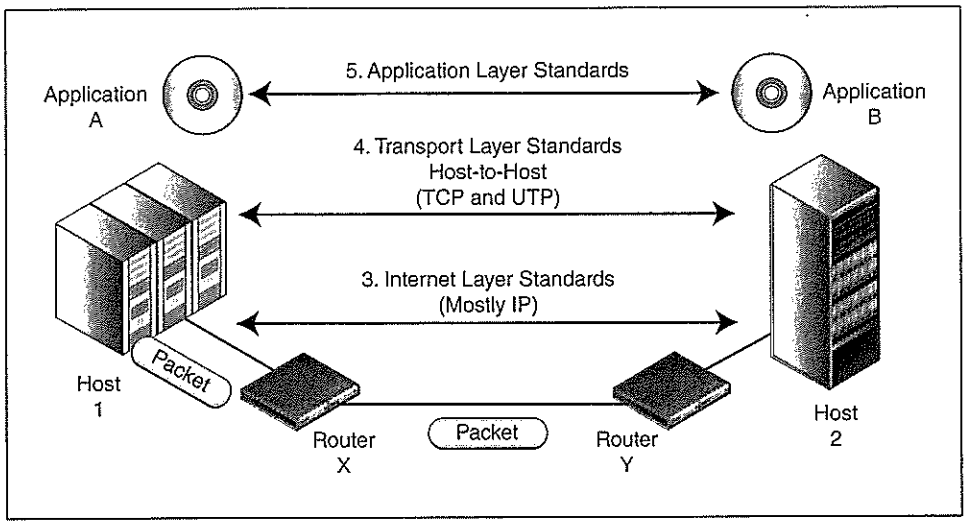


FIGURE 2-3 Transport and Application Processes

destination host. In contrast, Figure 2-3 shows that transport and application processes govern processes that exist only on the two communicating hosts.

- The transport layer supplements the internet layer. Internet layer operation typically is a best-effort service that does not guarantee that packets will be delivered. The transport layer is a "fix-up" layer that can add reliability and other desirable characteristics to transmission across an internet. In addition, the source host transport layer process fragments application messages. These fragments are sent in individual packets. The destination host transport process reassembles the segments and passes the application message to the application.
- The application layer is for application standards. When two e-mail programs need to work together, they use an e-mail application standard. For webservice, HTTP is the application layer standard. There are more application layer standards than there are standards at all other layers combined because there are so many applications and because different applications usually need different application standards.

THE FIVE LAYERS Figure 2-4 recaps the five layers.

- Speaking broadly, the physical and data link layers govern transmission through single networks.
- Also speaking broadly, the internet and transport layers together govern transmission through an internet. The internet layer governs packet organization and raw packet delivery. The transport layer fixes up problems and does fragmentation and assembly.
- Finally, the application layer governs how two applications work together.

Test Your Understanding

1. a) Give the definition of network standards that this chapter introduced. b) In this book, do *standards* and *protocols* mean the same thing?

Broad Function	Layer	Name	Specific Function
Interoperability of application programs	5	Application	Application layer standards govern how two applications work with each other, even if they are from different vendors.
Transmission across an internet (routed network)	4	Transport	Transport layer standards govern aspects of end-to-end communication between two end hosts that are not handled by the internet layer. These standards also allow hosts to work together even if the two computers are from different vendors or have different internal designs.
	3	Internet	Internet link layer standards govern the transmission of packets across an internet—typically by sending them through several routers along the route. Internet layer standards also govern packet organization, timing constraints, and reliability.
Transmission across a single switched or wireless network	2	Data Link	Data link layer standards govern the transmission of frames across a single switched network—typically by sending them through several switches along the data link. Data link layer standards also govern frame organization, timing constraints, and reliability.
	1	Physical	Physical layer standards govern transmission between adjacent devices connected by a transmission medium.

FIGURE 2-4 Layers Recap

Network Standard Characteristics

Network standards govern communication. Figure 2-5 notes, more specifically, that standards govern five specific things about message exchanges: semantics, syntax, order, reliability, and format. In this section, we will focus on message order, semantics, and syntax, but we will also introduce the concept of whether a standard is reliable.

MESSAGE ORDERING In medicine and many other fields, a *protocol* is a prescribed series of actions to be performed in a particular order. In cooking, recipes work this way. If you do not put together the ingredients of a cake in the right amounts and prepare them in the right order, the cake is not likely to turn out very well.

In this same way, network standards govern **message ordering**. For the Hypertext Transfer Protocol standard that we saw in Chapter 1, message ordering is very simple. The client sends an HTTP request message, and the server sends back an HTTP response message. Many protocols, including the Transmission Control Protocol (TCP) standard, which we will see in this chapter and in Chapter 9, involve many messages being sent in precise order.

Human beings are intelligent, so message ordering in human conversations tends to be informal and even chaotic. Software is not intelligent, so message ordering in protocols has to be very rigid and exact.

SEMANTICS To limit the complexity of software, protocols usually define only a few message types, and these types usually have only a few options. Put another way, network protocols greatly limit the **semantics** (meaning) of their messages. For example, the

Network Standards

Network standards are rules that govern the exchange of messages between hardware or software processes on different hosts, including messages (ordering, semantics, and syntax), reliability, and connection orientation.

Message Order

Turn taking, order of messages in a complex transaction, who must initiate communication, etc.
In the World Wide Web, the client program sends an HTTP request message
The webserver program sends back an HTTP response message
The client must initiate the interaction
Other network standards have more complex turn-taking; for instance, TCP
Human turn taking is loose and flexible
Message order for network standards must be rigid because computers are not intelligent

Message Semantics

Semantics = the meaning of a message
HTTP request message: "Please give me this file"
HTTP response message: Here is the file. (Or, I could not comply for the following reason.)
Network standards normally have a very limited set of possible message meanings
For example, HTTP requests have only a few possible meanings
GET: Please give me a file
PUT: Store this file (not often used)
A few more

Message Syntax (Organization)

Like human grammar, but more rigid
Header, data field, and trailer (Figure 2-8)
Not all messages have all three parts
Field lengths are measured in bits or bytes
Bytes are also called octets

FIGURE 2-5 Network Standards Concepts

most common HTTP request message is a GET message, which requests a file. There is also a POST request message, which uploads a file to the webserver. Similarly, the HTTP response message is essentially, "Here is the file," or "Sorry, I can't deliver the file."

Semantics is the meaning of a message.

SYNTAX In addition, while human grammar is very flexible, network messages have very rigid **syntax**, that is, message organization. A little later in this chapter, we will look at the syntaxes of several important protocol messages.

Syntax is how a message is organized.

Test Your Understanding

2. a) What three aspects of message exchanges did we see in this section? b) Give an example not involving networking in which the order in which you do things can make a big difference. c) Distinguish between syntax and semantics.

EXAMPLES OF MESSAGE ORDERING

We will look at two examples of message ordering. We will look first at the very simple message ordering in HTTP. We will then look at the more complex message ordering in TCP.

Message Ordering in HTTP

Figure 2-6 illustrates an HTTP request–response cycle. As we have just noted, the client sends a request, and the server sends a response. Note that the cycle is always initiated by the client, not by the server. The server cannot transmit unless the client has sent it an HTTP request message. This is a very simple type of message ordering.

Message Ordering and Reliability in TCP at the Transport Layer

Many protocols have much more complex rules for message ordering. We will look at the Transmission Control Protocol at the transport layer to see an example of this complexity.

THE SITUATION Figure 2-7 shows the transport layer processes on Host A and Host B. They are communicating via HTTP at the application layer. The Hypertext Transfer Protocol requires the use of TCP at the transport layer. The figure shows a sample communication session, which is called a connection.

SEGMENTS In TCP, messages are called **TCP segments** because each carries a segment of an application message (or is a control segment that does not carry application data).

THE THREE-STEP HANDSHAKE OPENING The communication begins with a **three-step handshake** to begin the communication.

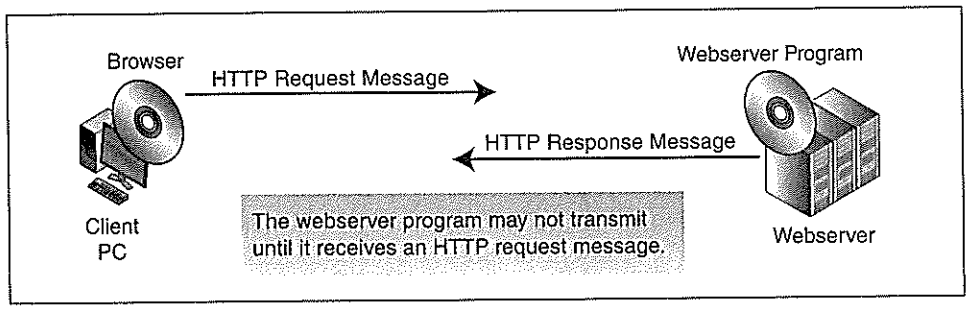


FIGURE 2-6 An HTTP Request–Response Cycle

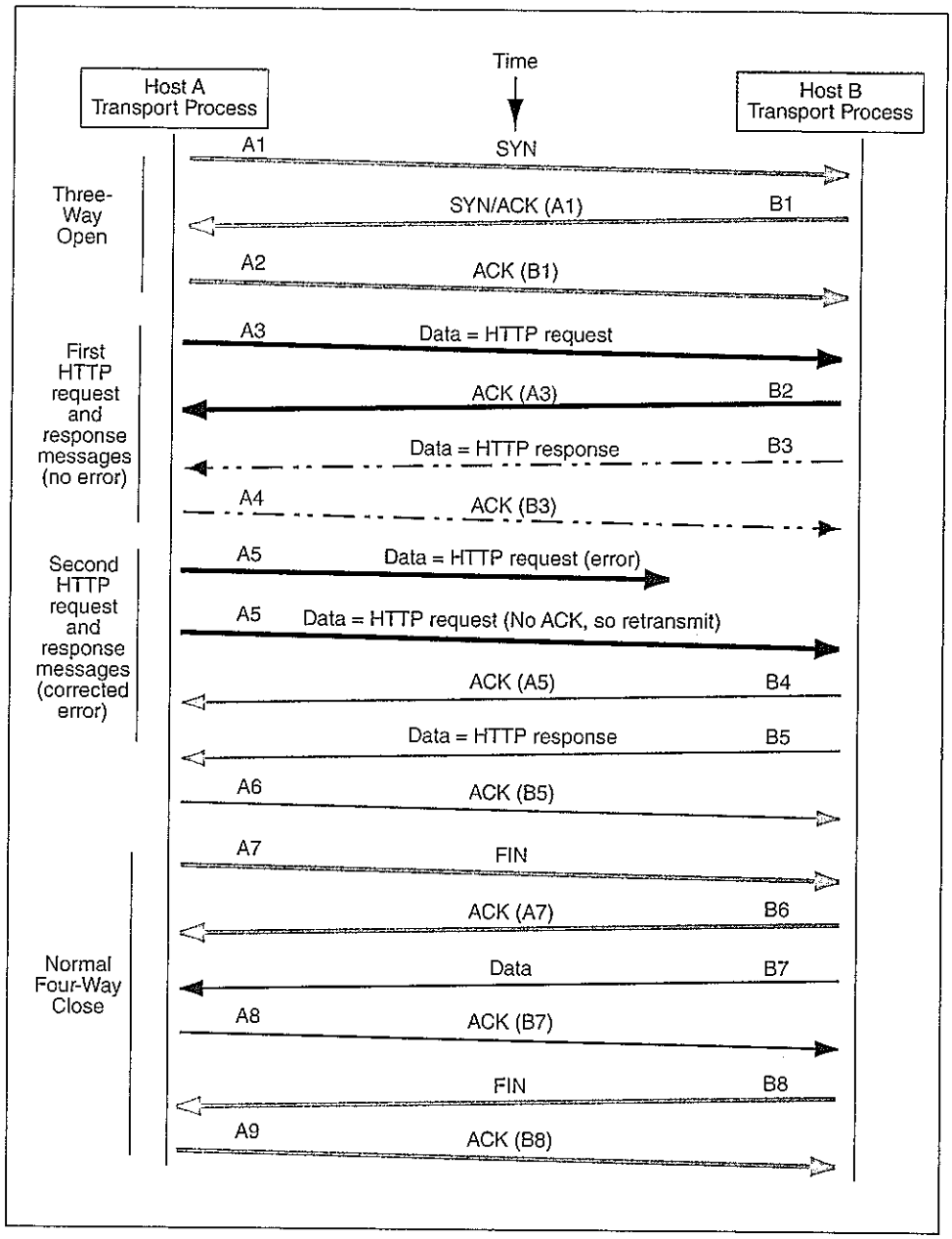


FIGURE 2-7 A TCP Connection

- Host A, which is the client in the HTTP exchanges, initiates the communication. It transmits a TCP SYN segment to Host B. This indicates that Host A wishes to communicate.
- Host B sends back a TCP SYN/ACK segment. The SYN indicates that it also is willing to begin the communication. The ACK part is an acknowledgment of

Host A's SYN message. In TCP, all segments are acknowledged, with the primary exception of pure ACKs. (If pure ACKs had to be acknowledged, there would be an endless series of ACKs.)

- Host A sends back a pure TCP ACK segment. This acknowledges Host B's SYN/ACK.

In TCP, all segments are acknowledged, with the primarily exception of pure ACKs.

CONNECTIONS TCP creates connections with distinct openings and closings. This is like a telephone call, in which you informally make sure that the other person can talk at the start of a call and mutually agree to end the call. In technical jargon, TCP is a **connection-oriented** protocol.

SEQUENCE NUMBERS In a connection-oriented protocol, each message is given a sequence number. This allows the receiver to ensure that no message is missing and allows the receiving process to deal with duplicate segments. (It simply discards duplicates.)

Sequence numbers are important because application messages are fragmented and delivered in separate packets. Sequence numbers allow the receiver to place the segments in order and reassemble them.

Note in Figure 2-7 that each side numbers its sequence numbers individually. For simplicity, we have called Host A's sequence numbers A1, A2, A3, and so forth. We have done the same with Host B's messages. So Host A's SYN segment is A1, while Host B's SYN/ACK is B1, and Host A's acknowledgment of the SYN/ACK is A2.

CARRYING APPLICATION DATA The next four segments (A3, B2, B3, and A4) constitute a request–response cycle.

- A3 carries an HTTP request.
- B2 is an ACK of A3.
- B3 carries the HTTP response message.
- A4 acknowledges the receipt of B3.

Usually, HTTP request messages are small enough to fit in a single TCP segment. However, most HTTP responses are long and must be sent in a number of TCP segments. This does not change the basic picture, however. There would simply be several more exchanges like B3 and A4.

RELIABILITY TCP is a **reliable** protocol. This means that it corrects errors.

- Segment A5 is sent but never reaches Host B.
- Host B does not send an acknowledgment, because ACKs are only sent when a segment is received correctly.
- Host A realizes that A5 has not been acknowledged. After a certain period of time, it retransmits A5.
- This time, the segment arrives correctly at Host B. Host B sends B4, which is an acknowledgment of A5.
- Finally, Host B sends an HTTP response message (B5) and receives an ACK (A6). Again, sending an HTTP response message tends to take several TCP data/acknowledgment cycles.

In this example, Segment A5 never reached the receiving transport process. What would have happened if A5 had reached the transport process but was damaged during transmission? In this case, the receiving transport process would discard the segment. It would not send an ACK. So there is a simple rule for ACKs. Unless a transport process receives a segment correctly, it does not send an acknowledgment.

Unless a transport process receives a segment correctly, it does not send an acknowledgment.

THE FOUR-STEP HANDSHAKE CLOSING Host A has no more HTTP request messages to send, so it closes the connection. It does so by sending a FIN segment (A7), which Host B acknowledges (B6). This means that Host A will not send new data. However, it will continue to send ACKs to segments sent by Host B.

- Host B has one more data segment to send, B7. When it sends this segment, Host A's transport process responds with an ACK (A8).
- Now, Host B is finished sending data. It sends its own FIN segment (B8) and receives an acknowledgment (A9).
- The connection is closed.

PERSPECTIVE TCP is a fairly complex protocol. It uses connections so that it can apply sequence numbers to segments. This allows it to fragment long application messages and deliver the segments with an indication of their order. It also uses connections so that it can provide reliable data to the application layer program above it.

We will see that almost all other protocols are unreliable. Many standards check for errors, but if they find an error, they simply discard the message. Discarded messages never get to the transport process on the other host, so they are never acknowledged. Receiving no acknowledgment, the sender resends them.

Why make only TCP reliable? There are two answers. First, TCP sits just below the application layer. This allows it to send clean data to the application program regardless of errors at lower levels, which are corrected by TCP resends.

Second, as Figure 2-3 shows, only the two hosts have transport layer processes, so error correction is done only once, on the two hosts. It is not done at each packet hop between routers or in each frame hop between switches. Error correction is a resource-consuming process, so it should be done as little as possible. Doing error correction at the transport layer processes on the two hosts accomplishes this.

Test Your Understanding

3. a) Describe the simple message ordering in HTTP. b) In HTTP, can the server transmit if it has not received a request message from the client? c) Describe the three-step handshake in TCP connection openings. d) What kind of message does the destination host send if it does not receive a segment during a TCP connection? e) What kind of message does the destination host send if it receives a segment that has an error during a TCP connection? f) Under what conditions will a source host TCP process retransmit a segment? g) Describe the four-step handshake in TCP connection closes. h) After a side initiates the close of a connection by sending a FIN segment, will it send any more segments? Explain. i) In Figure 2-7, suppose

Host A had already sent A6 before it realized that it would need to resend A5. When it then resent A5, A6 would arrive before A5. How would Host B be able to put the information in the two segments back in order?

EXAMPLES OF MESSAGE SYNTAX

We have just looked at message ordering. Now we will turn to message syntax. In this book, we will be looking at the syntax of many different types of messages. To give you a feeling for message syntax, we will look at the syntax of five important message types.

Syntax: General Message Organization

Figure 2-8 shows that message syntax in general has three parts—a header, a data field, and a trailer.

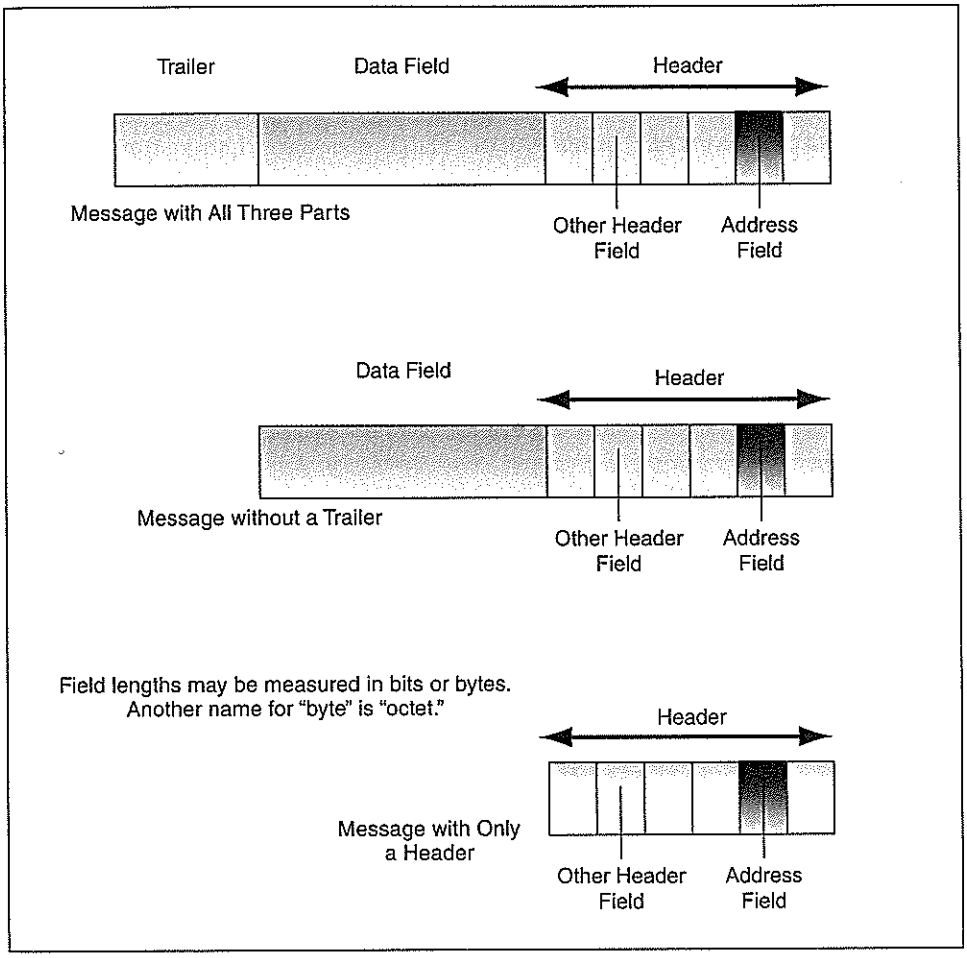


FIGURE 2-8 General Message Organization

DATA FIELD The **data field** is the heart of the message. It contains the content being delivered by the message. In an HTTP response message, the data field contains the file that the response message is delivering.

The data field contains the content being delivered by a message.

HEADER The message **header**, quite simply, is everything that comes before the data field.

The message header is everything that comes before the data field.

TRAILER Some messages also have **trailers**, which consist of everything coming *after* the data field.

The message trailer is everything that comes after the data field.

NOT ALL MESSAGES HAVE ALL THREE PARTS HTTP messages demonstrate that only a header is present in all messages. Data fields are not always present but are very common. Trailers are not common.

FIELDS IN HEADERS AND TRAILERS The header and trailer usually contain smaller syntactic sections called **fields**. For example, a frame or packet has a destination address header field, which allows switches or routers along the way to pass on the frame or packet they receive. When we look at network standard messages in this chapter and in later chapters, we will be concerned primarily with header fields and trailer fields.

The header and trailer usually contain smaller syntactic sections called fields.

OCTETS Field lengths can be measured in bits. Another common measure for field lengths in networking is the octet. An **octet** is a group of 8 bits. Hey, isn't that a byte? Yes, exactly. *Octet* is just another name for *byte*. The term is widely used in networking, however, so you need to become familiar with it. *Octet* actually makes more sense than *byte*, because *oct* means "eight." We have octopuses, octagons, and octogenarians.[1]

An octet is a group of 8 bits.

[1]What is the eighth month? (Careful!)

Test Your Understanding

4. a) What are the three general parts of messages? b) What does the data field contain? c) What is the definition of a header? d) Is there always a data field in a message? e) What is the definition of a trailer? f) Are trailers common? g) Distinguish between headers and header fields. h) Distinguish between octets and bytes.

The Ethernet Frame Syntax

Messages at the data link layer are frames. In wired local area networks (LANs), the dominant network standard is Ethernet. Actually, Ethernet, like most "standards," is really a family of standards. Ethernet has many different physical layer protocols from which a company can choose. However, generally speaking, it has a single frame standard, which Figure 2-9 illustrates.

The fields in the frame are delimited by their lengths in octets or bits. The destination host or switch first receives the first bit of the preamble field. It then counts bits until it gets to the next field, the field after it, and so forth. Then it can process the frame.

Ethernet has a complex frame syntax. We will look at its components in more detail in Chapter 5. There are only four fields that we need to emphasize at this time: source and destination address fields, data field packet, and Frame Check Sequence field.

SOURCE AND DESTINATION ADDRESS FIELDS Ethernet has a destination MAC address field and a source MAC address field. These are like the address and return address on a postal envelope. Switches use the destination MAC address to forward frames.

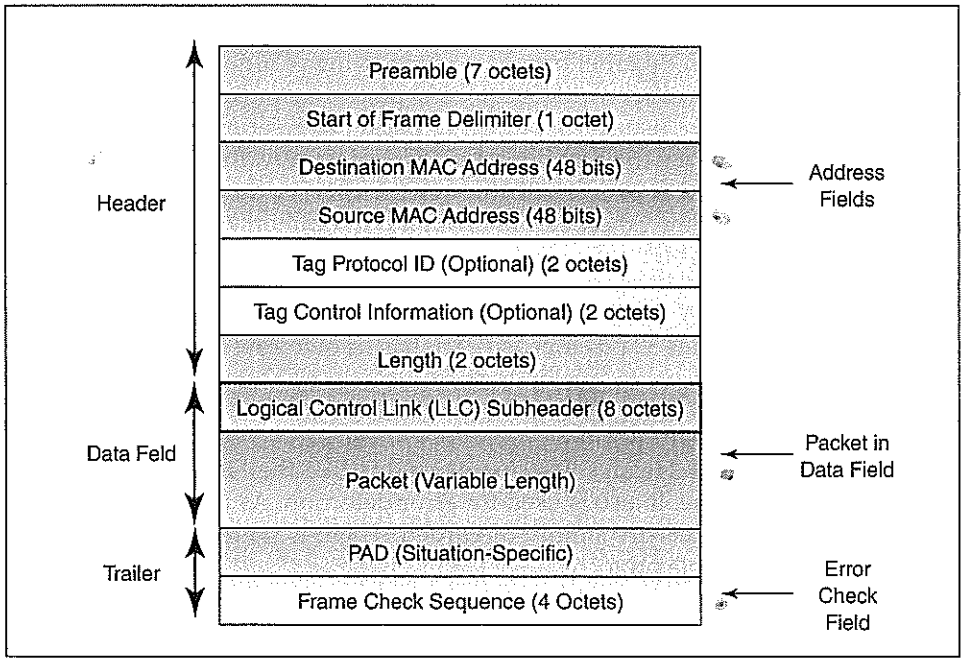


FIGURE 2-9 Ethernet Frame

Note that Ethernet addresses are 48 bits long—not 32 bits long, like IP addresses. Different single network standards have different address lengths. Ethernet addresses are called MAC addresses for reasons we will see in Chapter 6. In that chapter, we will also see that an Ethernet MAC address has six pairs of symbols separated by dashes. An example would be B7-23-DD-6F-C8-AB.

PACKET IN THE DATA FIELD We saw in Chapter 1 that frames carry packets in their data fields. The figure illustrates how this occurs in Ethernet.

FRAME CHECK SEQUENCE FIELD The four-octet Frame Check Sequence field is used for error detection. The sending data link layer process computes a number based on other bits in the frame. The receiver recomputes this 32-bit number. If the recomputed number is the same as the number transmitted in the Frame Check Sequence field, then there have been no errors during transmission, and the receiver accepts the frame. If the numbers are different, there was a propagation error and the frame is corrupted. If so, the receiving data link layer process simply discards the frame. This is error detection but not full error correction. Ethernet is not a reliable protocol.

Test Your Understanding

5. a) How long are Ethernet MAC addresses? b) What devices read Ethernet destination MAC addresses? c) If the receiver detects an error on the basis of the value in the Frame Check Sequence field, what does it do? d) Ethernet does error detection but not error correction. Is Ethernet a reliable protocol? Explain.

The Internet Protocol (IP) Packet Syntax

32 BITS PER ROW Figure 2-10 illustrates the syntax of Internet Protocol (IP) version 4 (IPv4) packets. Later, we will look at the syntax of IP version 6 (IPv6) packets.

An IP packet, like an Ethernet frame, is a long string of bits (1s and 0s). Of course, drawing the packet this way would require a page several meters wide. Instead, Figure 2-10 shows that we usually depict an IP packet as a series of rows with 32 bits per row. In binary counting, the first bit is zero. Consequently, the first row shows bits 0 through 31. The next row shows bits 32 through 63. This is a different way of showing syntax than we saw with the Ethernet frame, but it is a common way of showing syntax in TCP/IP standards, so you need to be familiar with it.

SOURCE AND DESTINATION IP ADDRESS FIELDS Each IPv4 packet has source and destination IP addresses. Each is 32 bits long, so each has its own row in the header. Routers use destination IP addresses to decide how to forward packets so that they will get closer to their destination.

UNRELIABILITY The IPv4 **Header Checksum** field is like the Frame Check Sequence field in the Ethernet frame. It also is used for error detection. As in the case of Ethernet frames, incorrect IP packets are simply discarded. There is no retransmission. So like Ethernet, IP is not a reliable protocol.

Test Your Understanding

6. a) How many octets long is an IPv4 header if there are no options? (Look at Figure 2-10.) b) List the first bit number on each IPv4 header row in Figure 2-10,

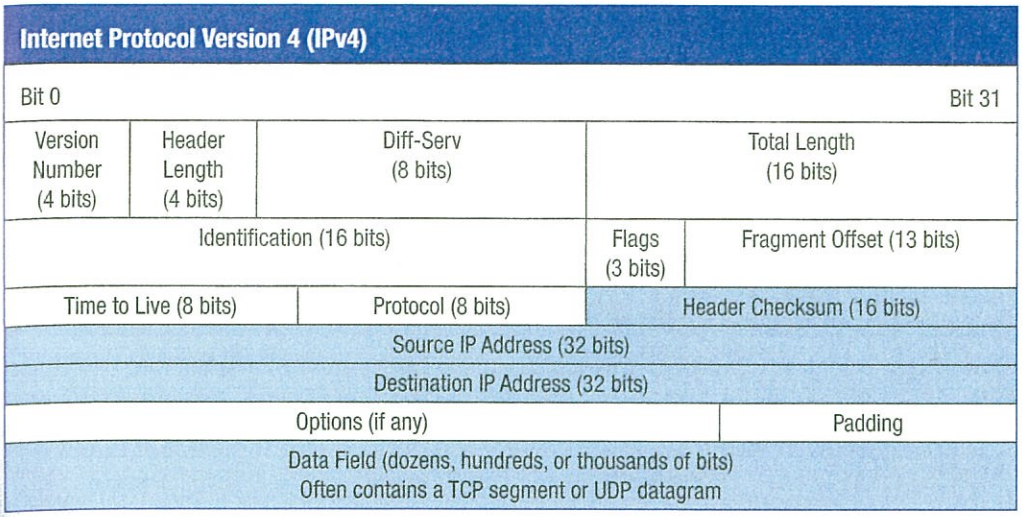


FIGURE 2-10 The Internet Protocol (IP) Packet Syntax in IPv4

not including options. Remember that the first bit in Row 1 is Bit 0. c) What is the bit number of the first bit in the destination address field in IPv4? (Remember that the first bit in binary counting is Bit 0.) d) How long are IPv4 addresses? e) What device in an internet besides the destination host reads the destination IP address? f) What is this device's purpose in doing so? g) Is IP reliable or unreliable? Explain.

Transmission Control Protocol (TCP) Segment Syntax

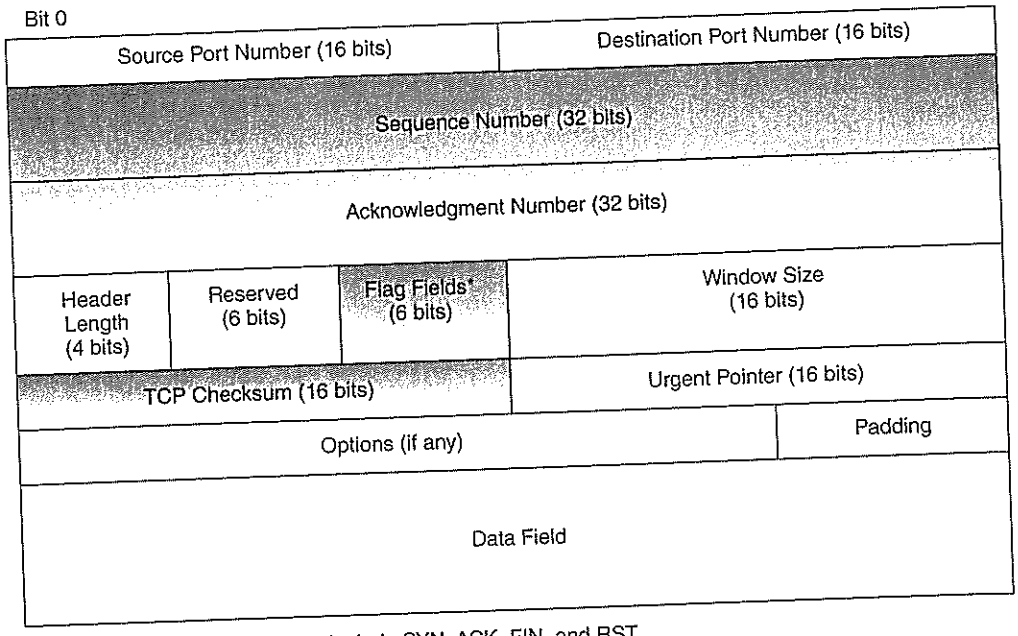
Earlier, we saw message ordering in the transmission of TCP segments. Now we will look at the syntax of TCP segments.

FIELDS IN TCP/IP SEGMENTS When IP was created, it was designed to be a very simple "best effort" protocol (although its routing tables are complex). The IETF left more complex internetwork transmission control tasks to TCP. Consequently, network professionals need to understand TCP very well. Figure 2-11 shows the organization of TCP messages, which are called TCP segments.

FLAG FIELDS TCP has six single-bit fields. Single-bit fields in general are called **flag fields**. If a flag field has the value 1, it is said to be **set**. (If it has the value 0, it is said to be **not set**.) In TCP, flag fields allow the receiving transport process to identify the kind of segment it is receiving. We will look at three of these flag bits:

- If the ACK (acknowledgment) bit is set, then the segment acknowledges another segment. When the ACK bit is set, the acknowledgment field also must be filled in to indicate which message is being acknowledged.
- If the SYN (synchronization) bit is set (has the value 1), then the segment requests a connection opening.
- If the FIN (finish) bit is set, then the segment requests a normal connection closing.

TCP Segment



*Flag fields are 1-bit fields. They include SYN, ACK, FIN, and RST.

UDP Datagram

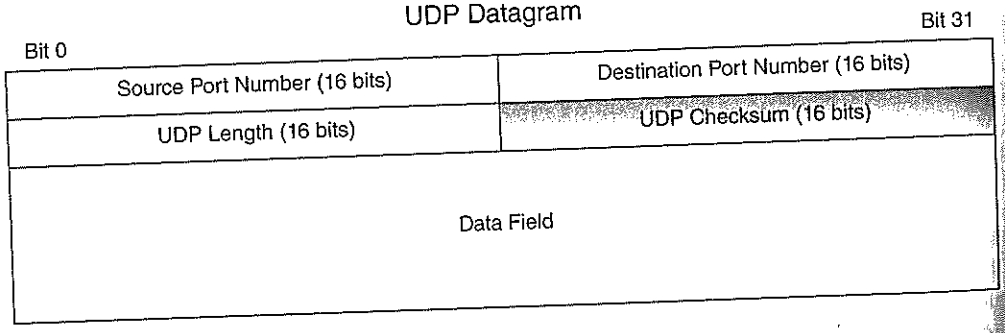
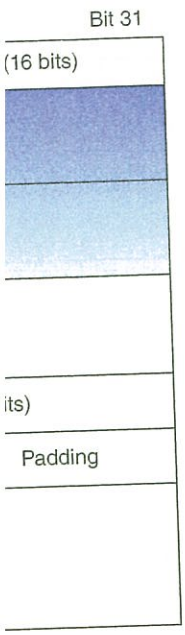


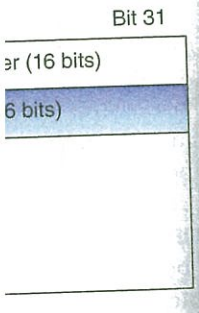
FIGURE 2-11 TCP Segment and UDP Datagram

Single-bit fields are called flag fields. If a flag field has the value 1, it is said to be set. (If it has the value 0, it is said to be not set.)

Earlier, we talked about TCP SYN segments, ACK segments, and FIN segments. These are simply segments in which the SYN, ACK, or FIN bits are set, respectively.

SEQUENCE NUMBERS Each TCP segment has a unique 32-bit **sequence number**. This sequence number increases with each segment. Sequence numbers allow the receiving transport process to put arriving TCP segments in order if IP delivers them out of order.





is said to be set. (If

nd FIN segments.
t, respectively.

equence number.
umbers allow the
f IP delivers them

ACKNOWLEDGMENT NUMBERS Earlier in this chapter, we saw that TCP uses acknowledgments (ACKs) to achieve reliability. If a transport process receives a TCP segment correctly, it sends back a TCP segment acknowledging the reception. We saw earlier that if the sending transport process does not receive an acknowledgment, it transmits the TCP segment again.

The **acknowledgment number** field indicates which segment is being acknowledged. One might expect that if a segment has sequence number X, then the acknowledgment number in the segment that acknowledges it would have acknowledgment number X. Later in this book, we will see that the situation actually is more complex.

The acknowledgment number indicates which segment is being acknowledged.

Test Your Understanding

7. a) Why was TCP designed to be complex? b) Why is it important for networking professionals to understand TCP? c) What are TCP messages called?
8. a) Why are sequence numbers good? b) What are 1-bit fields called? c) If someone says that a flag field is set, what does this mean? d) If the ACK bit is set, what other field must have a value? e) What is the purpose of the acknowledgment number field?

(4) User Datagram Protocol (UDP) Datagram Syntax

Applications that cannot use the high functionality in TCP or that do not need this functionality can use the **User Datagram Protocol (UDP)** at the transport layer instead of TCP. UDP does not have openings, closings, or acknowledgments, and so it produces substantially less traffic than TCP. UDP messages are called datagrams. Because of UDP's simple operation, the syntax of the UDP datagram shown in Figure 2-11 is very simple. Besides two port number fields, which we will see next in this chapter, there are only two header fields.

- There is a **UDP length** field so that the receiving transport process can know how long the datagram is. The packet in the data field will have variable length, so the UDP datagram will have variable length.
- There also is a **UDP checksum** field that allows the receiver to check for errors in this UDP datagram.[2] If an error is found, however, the UDP datagram is discarded. There is no mechanism for retransmission.

Test Your Understanding

9. a) What are the four fields in a UDP header? b) Describe the third. c) Describe the fourth. d) Is UDP reliable? Explain.

Port Numbers

Both TCP and UDP headers begin with two port number fields, one specifying the sender's port number and one specifying the receiver's port number. Servers and clients use these port number fields differently.

[2]If the UDP checksum field has 16 zeroes, error checking is not to be done at all.

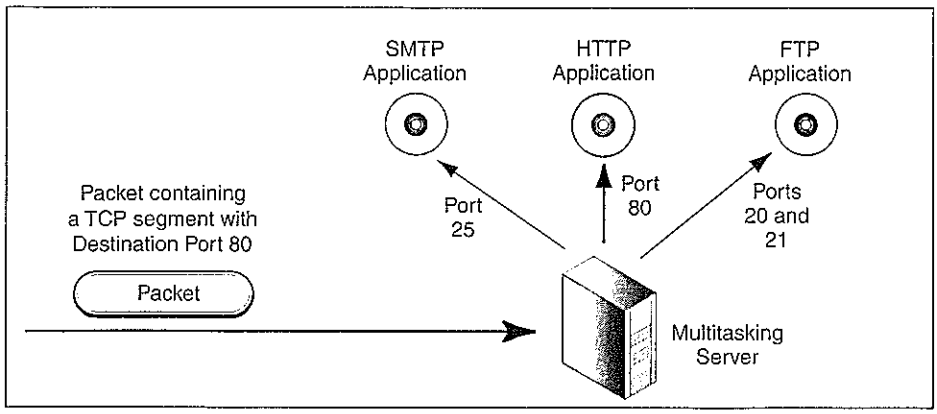


FIGURE 2-12 Server Port Numbers

SERVER PORT NUMBERS Computers are multitasking machines, which means that they can run several programs at the same time. Figure 2-12 shows a server running **SMTP**, **HTTP**, and **FTP** programs. If a packet arrives, how does the TCP or UDP process know which of the application programs running on the server should receive the message?

The answer is that the TCP or UDP process uses the port number. The port number field contains a number that specifies a specific application running on the server. Port 20 or 21 specifies the FTP (file transfer protocol) program, while Port 25 specifies the SMTP (e-mail) program, and Port 80 specifies the HTTP (World Wide Web) application.

A server administrator can choose any port number for a program, but there are **well-known port numbers** that are normally used to specify particular programs. That is the case for the port numbers used by the server. Most webservers use this port number for the webserver program. The well-known port numbers have a port number range reserved for their use—port numbers 0 through 1023.

CLIENT PORT NUMBERS Clients use port numbers in a different way. For every conversation a client initiates or accepts, it randomly generates a number in the **ephemeral port number** range. On Windows computers, this is the range from port 1024 to Port 4999. These port numbers are ephemeral, in the sense that they are discarded when a conversation between the client and a particular webserver ends. If the client communicates with the same server program later, the client will generate a new ephemeral port number.

Figure 2-13 shows a client host (60.171.18.22) communicating with a blue server host (1.33.17.13). The server port number is Port 80, indicating that the client is communicating with the HTTP program on the server. The client has generated ephemeral Port 2707. When the client transmits to the server, the source port number field has the value 2707 and the destination port number 80. When the server replies, the source port number is 80 and the destination port number is 2707.

The client is simultaneously connected to an SMTP server (123.30.17.120), which has the well-known port number 25. For this conversation, the client randomly generates ephemeral Port 4400. When the client transmits, the source port number is 4400 and the destination port number is 256.

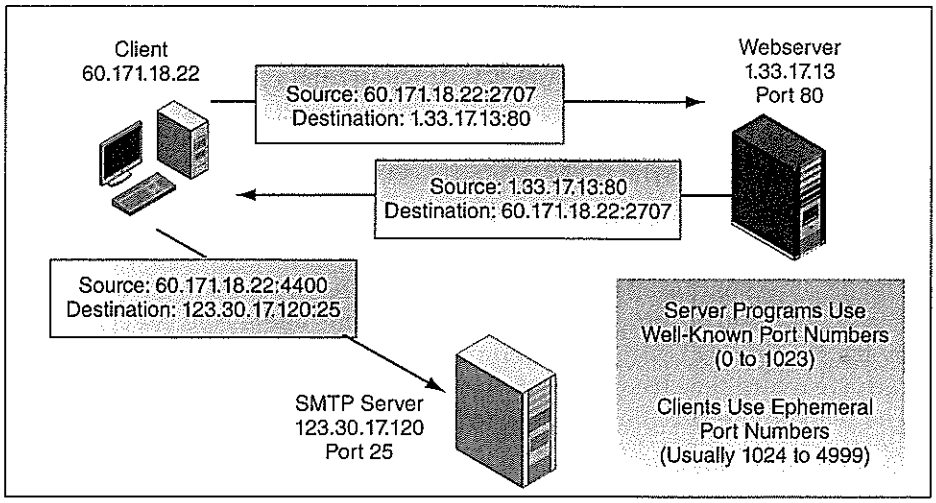


FIGURE 2-13 Client Port Numbers and Sockets

SOCKETS Figure 2-13 shows that a conversation always involves a source IP address and a source part number, plus a destination IP address and a destination port number. It is common to represent each IP address and port number as a socket, which is simply the IP address, a colon, and the port number. When the client transmits to the webserver, the source socket is 60.171.18.22:2707 and the destination socket is 1.33.17.13:80. When the webserver replies, the source socket is 1.33.17.13:80, and the destination socket is 60.171.18.22:2707.

Test Your Understanding

10. a) What type of port numbers do servers use for common server programs? b) What type of port numbers do clients use when they communicate with server programs? c) What is the range of port numbers for each type of port? d) How are ephemeral port numbers generated? e) Why are they called *ephemeral*?
11. a) What is the syntax of a socket? b) In Figure 2-13, when the client transmits to the server host, what is the source port number? c) What is the destination port number? d) What is the source socket? e) What is the destination socket? f) When the SMTP server transmits to the client host, what is the source port number? g) What is the destination port number? h) What is the source socket? i) What is the destination socket?

(5) HTTP Request and Response Message Syntax

The highest layer is the application layer (Layer 5). Standards at this layer govern how application programs talk to one another. We have used HTTP in most of our examples so far. However, there are many application layer standards—more than there are standards at any other layer. There are application layer standards for e-mail, database queries, and every other application. After network professionals master the network and internetwork standards that this course presents, they spend much of the rest of their careers mastering application standards.

Unfortunately, some students become fixated on examples and lose sight of general principles. At the risk of feeding this fixation, we will look at the syntax for the Hypertext Transfer Protocol. Figure 2-14 illustrates the syntax of HTTP request and response messages.

HTTP REQUEST MESSAGE The HTTP request message is particularly simple. It consists of only two lines of text. Some HTTP request messages have additional lines, but it is rare to see an HTTP request message with more than a handful of lines.

- The first line specifies the GET method (which requests a file retrieval), the location of the file to be retrieved (/panko/home.htm), and the version of HTTP used by the sender (HTTP 1.1). This and other lines end with [CRLF]. This stands for carriage return/line feed. It means to start a new line.
- The second line specifies the host to which this HTTP request message should be sent.

These two lines form the header of the message. The sender is transmitting no data, so there is no data field. Nor is there anything after a data field, so there is no trailer.

HTTP

The application layer is the highest layer.
It has more standards than at any other layer.
HTTP is not the only application layer standard; it is one of many.
Many application layer protocols, such as SMTP for e-mail, are much more complex than HTTP.

HTTP Request Message

GET /panko/home.htm HTTP/1.1[CRLF]
Host: voyager.cba.hawaii.edu

HTTP Response Message

HTTP/1.1 200 OK[CRLF]
Date: Tuesday, 20-MAR-2012 18:32:15 GMT[CRLF]
Server: *name of server software* [CRLF]
MIME-version: 1.0[CRLF]
Content-type: text/plain[CRLF]
[CRLF]
File to be downloaded. A string of bytes that may be text, graphics, sound, video, or other content.

Notes

A relatively old feeling protocol
Fields ended by CRLF, which starts a new line
Based on e-mail (an old protocol) for rapid development

FIGURE 2-14 Hypertext Transfer Protocol Message Syntax

HTTP RESPONSE MESSAGE Figure 2-14 also shows the syntax of a relatively simple HTTP response message that responds to the HTTP request message we have just seen.

Header First, there is a header. The header is everything that comes before the data field, which is the file being delivered.

- The first line begins with HTTP/5 to show that it is willing to speak in this version of the standard. The 200 is a code that describes the response. The 200 code states that the message is delivering the requested file. The browser uses the code to know how to react. What about the final OK? The browser ignores it. HTTP is humanly readable, and the "OK" is designed to tell humans, who may not know what the 200 code means, that everything is alright.
- After the first line, other fields in the header have a very specific syntax. There is a keyword, a colon, and then a value for the keyword.
- The Date keyword, for instance, is followed by a colon and the date and time the HTTP response message was sent.
- The Server keyword, to give another example, describes the webserver software sending the response.
- The line containing the [CRLF] is a blank line. It indicates the end of the header.

Data Field Following the header is the file being sent to the browser. This is a long byte stream that constitutes the text document, photograph, video clip, or other type of file being delivered.

Trailer As usual, there is no trailer.

A TEXT PROTOCOL In contrast to the other protocols we have looked at, HTTP is a fairly primitive protocol. It delimits fields with carriage return/line feeds instead of having fields that end at a certain number of bits. Separating the header from the data field with a blank line also seems rather crude. Most application protocols are much more complex than HTTP.

Tim Berners-Lee, who created HTTP, based this standard on e-mail standards. An e-mail header has a number of keywords (e.g., To and From) followed by a colon, the value for the keyword, and a new line. E-mail standards were already very old when HTTP was created, but they got the job done. HTTP also got the job done. In particular, new keywords can be added very easily, given the robust way HTTP has of ending a field and starting a new field.

Test Your Understanding

12. a) Is the application layer standard always HTTP? b) Which layer has the most standards? c) At which layer would you find standards for voice over IP? (The answer is not explicitly in this section.) d) Are all application layer standards simple like HTTP? e) In HTTP response headers, what is the syntax of most lines (which are header fields)? f) In HTTP request and response message, how is the end of a field indicated? g) Do HTTP request messages have headers, data fields, and trailers? h) Do HTTP response messages that deliver files have headers, data fields, and trailers?

CONVERTING APPLICATION MESSAGES INTO BITS

Encoding

One function of application layer programs is to convert messages into bits. This conversion is called **encoding**. At the transport layer and lower layers, all messages consist of bits. Original application layer messages, in contrast, may have text, numbers, graphics images, video clips, and other types of information. It is the application layer's job to convert all of these into bits before putting them in the application layer message.

Test Your Understanding

13. a) What is encoding? b) At what layer is encoding done?

Encoding Text as ASCII

To convert text data to binary, applications use the **ASCII code**, whose individual symbols are each 7 bits long. Seven bits gives 128 possibilities. This is enough for all keys on the keyboard plus some extra control codes.

Figure 2-15 shows some ASCII codes. It shows that uppercase letters and lowercase letters have different ASCII codes. This is necessary because the receiver may need to know whether a character is an uppercase or lowercase letter. ASCII can also encode the digits from 0 through 9, as well as punctuation and other characters. There are even ASCII control codes that tell the receiver what to do. For example, when we looked at HTTP, we saw carriage returns and line feeds. A carriage return is 0101110, and a line feed is 0100000.

For transmission, the 7 bits of each ASCII character are placed in a byte. The eighth bit in the byte is not used today.[3]

Category	Meaning	7-Bit ASCII Code	Eighth Bit in Transmitted Byte
Uppercase Letters	A	1000001	Unused
Lowercase Letters	a	1100001	Unused
Digits (0 through 9)	3	0110011	Unused
Punctuation	Period	0101110	Unused
Punctuation	Space	0100000	Unused
Control Codes	Carriage Return	0001101	Unused
Control Codes	Line Field	0001010	Unused

FIGURE 2-15 Encoding Text as ASCII

[3]Early systems used the eighth bit in each byte as a "parity bit" to detect errors in transmission. This could detect a change in a single bit in the byte. At today's high transmission speeds, however, transmission errors normally generate multibit errors rather than single-bit errors. Consequently, parity is useless and is ignored.

Test Your Understanding

14. a) Explain how many bytes it will take to transmit "Hello World!" without the quotation marks. (The correct total is 12.) b) If you go to a search engine, you can easily find converters to represent characters in ASCII. What are the 7-bit ASCII codes for "Hello world" without the quotation marks? (Hint: H is 1001000.)

Whole Binary Numbers

Some application data consists of whole numbers. The sending application program represents whole numbers (integers) as whole **binary numbers**. Figure 2-16 shows how this is done.

With binary numbers, counting begins with 0. This is a source of frequent confusion. In counting, add 1 to each binary number to give the next binary number. There are four simple rules for addition in binary.

- If you add 0 and 0, you get 0.
- If you add 0 and 1 (or 1 and 0), you get 1.
- If you add 1 and 1, you get 10 (carry the one).
- If you add 1, 1, and 1, you get 11 (carry the one).

These rules are simple to use.

- In binary numbers, 8 is 1000.
- The number 9 adds a 1 to the final 0, giving a 1, so 9 is 1001.
- Adding 1 to the final 1 gives us 10, so 10 is 1010.
- The number 11 adds a 1 to the final 0, giving 1011.
- The number 12 adds a 1 to the final 1. With carries, this gives 1100.[4]

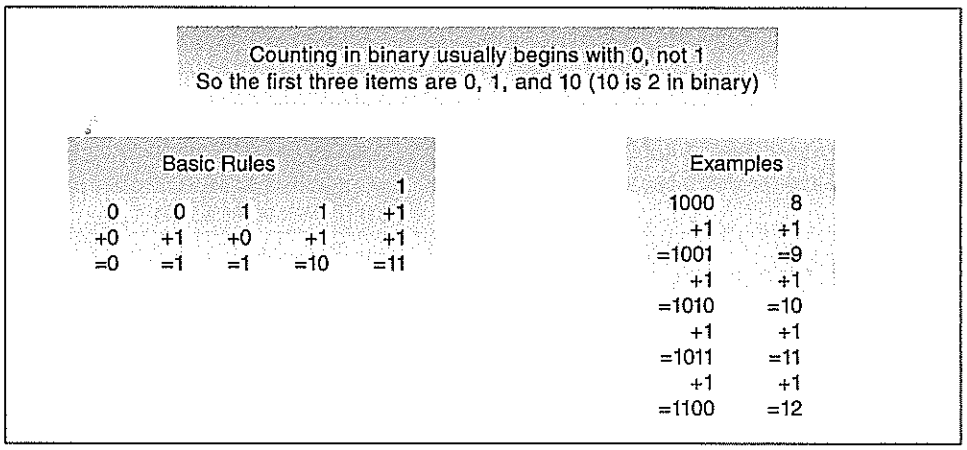


FIGURE 2-16 Binary Representation of Whole Numbers

[4]If you read Chapter 1a, you saw how to use the Microsoft Windows Calculator in programmer mode to convert between decimal and binary. You can also use it to check your binary calculations (obviously not on tests). First click on the Bin button to put the calculator in binary mode. Then use the calculator normally. For example, to add two binary numbers, click on Bin, enter the first binary number, hit the plus button, type the second binary number, and click on the equal sign to see the total.

Test Your Understanding

15. a) Does binary counting usually begin at 0 or 1? b) Give the binary representations for 13, 14, 15, 16, and 17 by adding one to successive numbers (12 is 1100).

Encoding Alternatives

Some application data can be expressed as alternatives, such as North, South, East, or West. The application layer process will create a field in the application layer message and represent each alternative as a group of bits. For instance, the four cardinal compass points can be represented by a 2-bit field within the application message. North, South, East, and West can be represented as 00, 01, 10, and 11, respectively. (These are the binary numbers for 0, 1, 2, and 3.) There is no order to the alternatives, so any choice can be represented by any pair of bits.

We just saw that having four alternatives requires a 2-bit field. More generally, if a field has N bits, it can represent 2^N alternatives. We have just seen that a 2-bit field can represent 2^2 alternatives, or 4. What if you need to represent three alternatives? One bit won't do it, because 2^1 is 2, and we need 3. We will have to use a 2-bit field, and one of the alternatives will go unused.

If a field has N bits, it can represent 2^N alternatives.

Figure 2-17 illustrates how alternatives encoding is done for 1-, 2-, 4-, 8-, 16-, and 32-bit fields. It shows that with one bit, you can encode yes or no, male or female, or any other dichotomy. Two bits, as we just saw, are good for the four cardinal compass points. With 4 bits, you can have 16 alternatives. You need 4 bits to represent the top 10 security threats, because 3 bits will encode only eight alternatives. Using 4 bits to

Bits in Field	Number of Alternatives That Can Be Encoded	Possible Bit Sequences	Examples
1	$2^1 = 2$	0, 1	Yes or No, Male or Female, etc.
2	$2^2 = 4$	00, 01, 10, 11	North, South, East, West; Red, Green, Blue, Black
4	$2^4 = 16$	0000, 0001, 0010, …	Top 10 security threats. Three bits would only give 8 alternatives. (With 4 bits, 6 values go unused)
8	$2^8 = 256$	00000000, 00000001, …	One byte per color gives 256 possible color levels
16	$2^{16} = 65{,}536$	0000000000000000, 0000000000000001, …	Two bytes per color give 65,536 color levels
32	$2^{32} = 4{,}294{,}967{,}296$	000000000000000 0000000000000000, etc.	Number of Internet Protocol Version 4 addresses
If a field has N bits, it can represent 2^N alternatives.			

FIGURE 2-17 Binary Encoding to Represent a Certain Number of Alternatives

represent 10 threats will waste six alternatives, but this is necessary. With 8 bits, you can represent 256 alternatives.

The 2^N rule is not only used at the application layer. In many layer messages, fields represent alternatives. A one-octet field has 8 bits, so it can represent 2^8 possible alternatives (256).

You should memorize the number of alternatives that can be represented by 4, 8, and 16 bits, because these are common field sizes. Each added bit doubles the number of alternatives, while each bit subtracted cuts the number of alternatives in half. So if 8 bits can represent 256 alternatives, 7 bits can represent 128 alternatives (half as many), while 9 bits can represent 512 alternatives (twice as many). How many alternatives can 6 and 10 bits represent?

Test Your Understanding

16. a) If a field is *N* bits long, how many alternatives can it represent? b) How many alternatives can you represent with a 4-bit field? c) For each bit you add to an alternatives field, how many additional alternatives can you represent? d) How many alternatives can you represent with a 10-bit field? (With 8 bits, you can represent 256 alternatives). e) If you need to represent 128 alternatives in a field, how many bits long must the field be? f) If you need to represent 18 alternatives in a field, how many bits long must the field be? g) Come up with three examples of things that can be encoded with 3 bits.

Encoding Voice

Increasingly, applications involve voice and even video. Figure 2-18 illustrates how voice encoding is done. Video encoding is done similarly. When you speak into a landline or mobile telephone, your voice loudness rises and falls thousands of times per

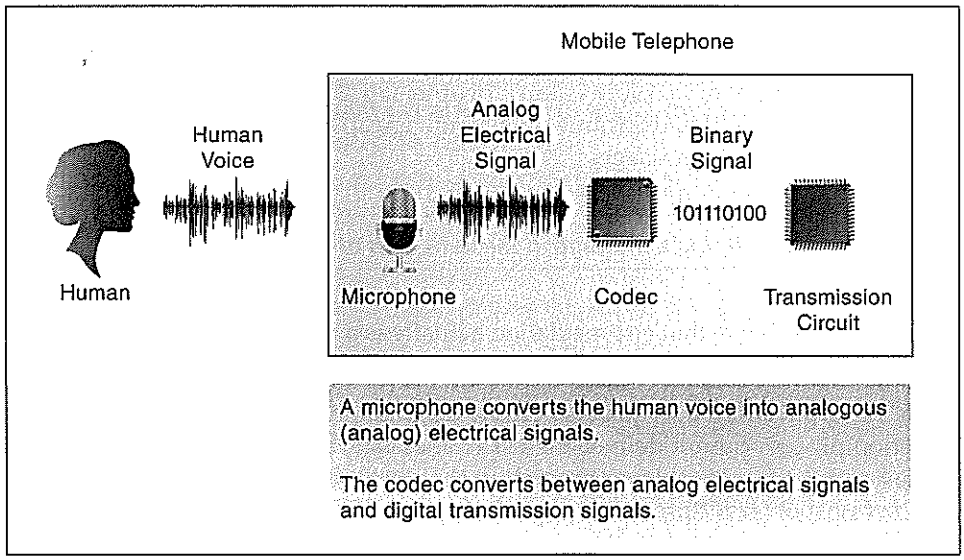


FIGURE 2-18 Encoding Voice

second at different frequencies. This sends pressure waves into your phone's microphone. The microphone converts this into increases and decreases in electricity. The resultant electrical signal rises and falls when your voice loudness rises and falls. To put it another way, the electrical signal is analogous to the voice signal. Therefore, we call this electrical signal an **analog signal**.

The analog signal must be converted into 1s and 0s. This is done by an electrical circuit called a **codec**. Converting outgoing analog signals into digital signals is called *encoding*. Converting incoming digital signals into analog signals to cause the earpiece to vibrate is called *decoding*. Codec is a merciful shortening of these two terms. As we will see in Chapter 11, there are many codec standards. Some specify a higher traffic burden in terms of bits per second; these have higher voice quality. Others are more parsimonious about bandwidth but give lower voice quality.

A codec converts between analog microphone signals and digital transmitted signals.

Test Your Understanding

17. a) Why is the electrical signal generated by a microphone called an analog signal? b) What two things does a codec do? c) Is there a single codec standard?

VERTICAL COMMUNICATION ON HOSTS

So far, we have talked about what happens at individual layers. For instance, the transport process on the sending host sends TCP segments or UDP datagrams to the transport process on the receiving host.

Obviously, however, there is no direct connection between the two hosts at the transport layer. Barring software telepathy, all communication must somehow travel through the physical layer.

In Chapter 1, we saw that Layer 3 packets are carried in the data fields of Layer 2 frames in single networks. Networking people say that the packet is encapsulated (placed) in the data field of the frame. In general, **encapsulation** is placing a message in the data field of another message.

Encapsulation is placing a message in the data field of another message.

Figure 2-19 shows that encapsulation actually is a process that occurs repeatedly.

- In the figure, the source host's application layer process sends an HTTP message to the application layer process on the destination host. The source host's application process cannot deliver the HTTP message, so it passes the HTTP message down to the transport layer process on the source host.
- The transport layer process encapsulates the HTTP message in the data field of a TCP segment. The transport layer then passes the TCP segment down to the internet layer process.

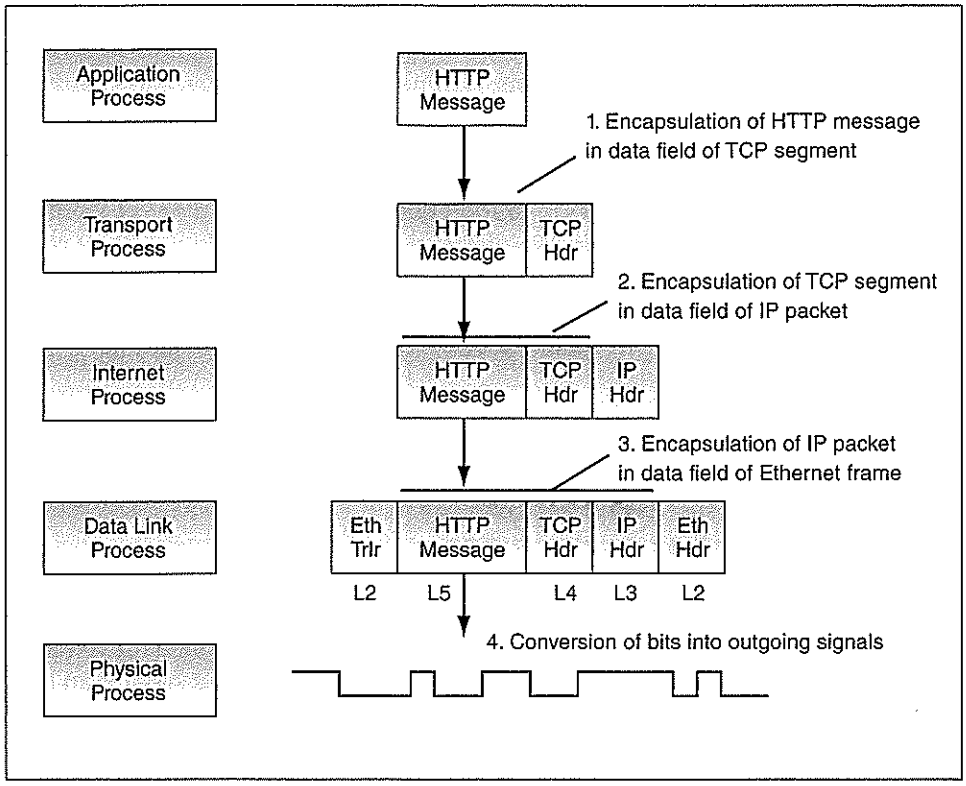


FIGURE 2-19 Layered Communication on the Source Host

- The Internet layer process encapsulates the TCP segment in the data field of an IP packet. It then passes the packet down to the data link layer.
- The data link layer process encapsulates the IP packet in a data link layer frame. If the single network standard is Ethernet, this is an Ethernet frame. If the single network standard is the Point to Point protocol (PPP), this is a PPP frame. The data link layer process may also add a trailer.

The whole process is like a set of Russian nesting dolls. So the frame consists of the following:

- The data link header
- The IP header
- The TCP header
- The application message segment
- The data link trailer if the single network standard has a trailer in its frame standard.

At the physical layer, something very different occurs. When the physical layer process receives the frame from the data link layer process, it does not do encapsulation. It merely converts the bits of the frame into signals and transmits its signal out the physical link connecting it to a switch, router, or host.

The purpose of this section has been to give you a light understanding of how layered communication works on the source host. However, there are many other aspects to vertical communication on source hosts, destination hosts, routers, and switches.

Test Your Understanding

18. a) What is encapsulation? b) Why is encapsulation necessary for there to be communication between processes operating at the same layer but on different hosts, routers, or switches? c) After the internet layer process in Figure 2-19 receives the TCP segment from the transport layer process, what two things does it do? d) After the data link layer process in Figure 2-19 receives the IP packet from the internet layer process, what two things does it do? e) After the physical layer process receives a frame from the data link layer process, what does the physical layer process do? f) If encapsulation occurs on the source host, what analogous process do you think will occur on the destination host? (The answer is not in the text.)

MAJOR STANDARDS ARCHITECTURES

Architectures

When you build a house, you do not design and build a bedroom, then move on to the next room. If you designed and built rooms one at a time, without having an overall plan for the house, you might end up having to go through the bathroom whenever you passed from the dining room to the kitchen. Instead, an architect begins with an overall plan for a house. This plan is called the architecture. The architecture specifies the functions of individual rooms (bedrooms, kitchens, etc.), how many rooms of each type are required, and how the rooms will fit together. Only after the architecture is finished does the architect begin to design individual rooms.

Network standards also need architectures. Network standards architectures break the standards functionality needed for communication into layers and define the functions of each layer. Only after they have done that do they develop individual standards. For instance, the TCP/IP standards architecture shown in Figure 2-20 has four layers. This architecture was created in the late 1970s. Since then, the Internet Engineering Task force has been adding individual standards, beginning with core standards such as IP, TCP, and UDP. It will continue to create individual standards for many years to come.

Network standards architectures break the standards functionality needed for communication into layers and define the functions of each layer. Only after they have done that do they develop individual standards.

Generally, all standards must be taken from the same architecture. For instance, electrical standards architectures in different countries have different voltages, amperages, alternating current cycles per second, plug shapes, and socket shapes. If you took a device built for the U.S. architecture, you could not take it to another country and simply change the plug to fit that country's sockets. Actually you *could* do that

Broad Purpose	TCP/IP	OSI	Hybrid TCP/IP-OSI
Applications	Application	Application (Layer 7)	Application (Layer 5)
		Presentation (Layer 6)	
		Session (Layer 5)	
Internetworking	Transport	Transport (Layer 4)	TCP/IP Transport Layer (Layer 4)
	Internet	Network (Layer 3)	TCP/IP Internet Layer (Layer 3)
Communication within a single switched LAN or WAN	Use OSI Standards Here	Data Link (Layer 2)	Data Link (OSI) Layer (Layer 2)
		Physical (Layer 1)	Physical OSI Layer (Layer 1)

Notes:

The Hybrid TCP/IP-OSI Architecture governs the Internet and dominates internal corporate networks.

OSI standards dominate the physical and data link layers (which govern communication within individual networks) almost exclusively.

TCP/IP dominates the internet and transport layers in internetworking and governs 80% to 90% percent of all corporate traffic above the data link layer.

FIGURE 2-20 The Hybrid TCP/IP–OSI Architecture

physically, but if you plugged the modified device into the wall jack, you should not expect the device to work. You would need a converter that converts between all elements in the U.S. electrical standards architecture and all elements in the foreign electrical architecture.

Although there are several major network standards architectures, two of them dominate actual corporate use: OSI and TCP/IP. Figure 2-20 illustrates layering in both. As you can see, they are quite different. Although it generally is impossible to mix standards for different architectures, this kind of mixing is possible with TCP/IP and OSI. It is possible to use a **hybrid TCP/IP–OSI architecture** that uses TCP/IP standards at some layers and OSI standards at other layers. This hybrid standards architecture is the dominant standards architecture in real corporations.

TCP/IP and OSI are often considered to be competitors. In fact, their standards agencies mostly specialize on different layers. When both are developing standards for a single layer, such as voice over IP standards at the application layer, their standards agencies often work together.

Test Your Understanding

19. a) What does a network standards architecture do? b) In what order are standards and standards architectures developed? c) What are the two dominant network standards architectures? d) What is the dominant network standards architecture in most real firms today? e) Are the two network standards architectures competitors?

The OSI Network Standards Architecture

OSI is an abbreviation for the "Reference Model of Open Systems Interconnection." *Reference model* is another name for *architecture*. An open system is one that is open to communicating with the rest of the world. In any case, OSI is rarely spelled out, which is merciful.

STANDARDS AGENCIES: ISO AND ITU-T Standards architectures are managed by organizations called **standards agencies**. Figure 2-21 shows that OSI has two standards agencies.

- One is the **International Organization for Standardization (ISO)**, which generally is a strong standards organization for manufacturing, including computer manufacturing. By the way, do not confuse OSI and ISO. OSI is an architecture; ISO is a standards agency.
- The other is the **International Telecommunications Union–Telecommunications Standards Sector (ITU–T)**.[5] Part of the United Nations, the ITU–T oversees international telecommunications standards.

Although ISO or the ITU–T must *ratify* all OSI standards, other organizations frequently *create* standards for inclusion in OSI. For instance, we will see in Chapter 6 that the IEEE creates Ethernet standards. These standards are not official until the ITU–T or ISO ratifies them, although this has always been a mere formality.

OSI'S DOMINANCE AT LOWER LAYERS (PHYSICAL AND DATA LINK) Although OSI is a seven-layer standards architecture (see Figure 2-22), OSI standards from Layers 3 through 6 are rarely used.

	OSI	TCP/IP
Standards Agency or Agencies	ISO (International Organization for Standardization) ITU–T (International Telecommunications Union–Telecommunications Standards Sector)	IETF (Internet Engineering Task Force)
Dominance	Dominant at physical and data link layers	Dominant at the internet and transport layers
Documents Are Called	Various	Mostly RFCs (requests for comments)

Note:
Do not confuse OSI (the architecture) with ISO (the organization).
The acronyms for ISO and ITU–T do not match their names, but these are the official names and acronyms.

FIGURE 2-21 OSI and TCP/IP

[5]No, the names and acronyms do not match for ISO and ITU-T, but these are the official names and acronyms for these two organizations.

Layer Number	OSI Name	Purpose	Use
7	Application	Governs remaining application-specific matters.	Some OSI applications are used
6	Presentation	Designed to handle data formatting differences and data compression and encryption. In practice, a category for general file format standards used in multiple applications.	Rarely used as a layer. However, many file format standards are assigned to this layer.
5	Session	Initiates and maintains a connection between application programs on different computers.	Rarely used
4	Transport	Generally equivalent to the TCP/IP transport layer. However, OSI transport layer standards are not compatible with TCP/IP transport layer standards.	Rarely used
3	Network	Generally equivalent to the TCP/IP internet layer. However, OSI network layer standards are not compatible with TCP/IP internet layer standards.	Rarely used
2	Data Link	End-to-end transmission in a single switched network. Frame organization. Switch operation.	Nearly 100 percent dominant
1	Physical	Physical connections between adjacent devices.	Nearly 100 percent dominant

FIGURE 2-22 OSI Layers

However, at the two lowest layers—the physical and data link layers—corporations use OSI standards almost universally in their networks. These two layers govern transmission within a switched or wireless network. Almost all single networks—both LANs and WANs—follow OSI standards at the physical and data link layers, regardless of what upper-layer standards they use. OSI standards are almost 100 percent dominant at the bottom two layers.

Almost all switched and wireless networks—both LANs and WANs—follow OSI standards at the physical and data link layers, regardless of what upper-layer standards they use.

Other standards agencies, recognizing the dominance of OSI at the physical and data link layers, simply specify the use of OSI standards at these layers. They then create standards only for internetworking and applications.

HIGHER LAYERS IN OSI The box *Higher Layers in OSI* discusses the higher layers in OSI. Apart from the application layer, the higher layers in OSI are rarely used in networking.

Test Your Understanding

20. a) What standards agencies are responsible for the OSI standards architecture? Just give the acronyms. b) At which layers do OSI standards dominate usage?

The TCP/IP Network Standards Architecture

The TCP/IP architecture is mandatory on the Internet at the internet and transport layers. TCP/IP is also widely used at these layers by companies for their internal corporate internets.

The TCP/IP architecture is named after two of its standards, TCP and IP, which we have already seen briefly. However, TCP/IP also has many other standards, including the UDP standard we have already seen. This makes the name TCP/IP rather misleading. Another confusing point about names is that TCP/IP is the *standards architecture*, while TCP and IP are individual *standards* within the architecture.

Note that TCP/IP is the standards architecture, while TCP and IP are individual standards within the architecture.

THE INTERNET ENGINEERING TASK FORCE (IETF) TCP/IP's standards agency is the **Internet Engineering Task Force (IETF)**. Traditionally, the IETF has been viewed as being in competition with ISO and ITU–T for creating standards. However, as noted earlier in this chapter, the IETF and these other organizations have begun to cooperate in standards development. For instance, the IETF is working closely with ITU–T on VoIP transmission standards.

Having evolved from the Network Working Group described in Chapter 1, the IETF historically has been rather informal. Its committees traditionally focus on consensus rather than on voting, and technical expertise is the main source of most power within the organization. Although corporate participation has somewhat "tamed" the IETF, it remains a fascinating organization, although its members might argue that "organization" is too strong a word.

A great deal of the IETF's success is due to the fact that the IETF typically produces simple standards and then adds to their complexity over time. In fact, IETF standards often have the word *simple* in their name, for instance, the Simple Mail Transfer Protocol. Consequently, the IETF develops standards quickly. In addition, vendors can develop TCP/IP-based products quickly and inexpensively because of this speed of development and simplicity. "Inexpensive and fast to market" is almost always a good recipe for success. In contrast, OSI standards often take a very long time to be developed and often are so bloated with functionality that they are uneconomical to implement.

Note that the success of TCP/IP is *not* primarily due to its use on the Internet. Many corporations had already shifted many of their networks to TCP/IP before the Internet became a dominant force in the 1990s.

REQUESTS FOR COMMENTS (RFCs) As we saw in Chapter 1, most documents produced by the IETF have the rather misleading name **requests for comment (RFCs)**. Every few years, the IETF publishes a list of which RFCs are **Official Internet Protocol Standards**. Each list of standards adds some RFCs to the list and deprecates some previously listed standards to indicate that they are no longer official standards.

DOMINANCE AT THE INTERNETWORKING LAYERS (INTERNET AND TRANSPORT) As noted earlier, physical and data link layer standards govern the transmission of data within a single *switched or wireless network*. We saw that OSI standards are dominant at these layers.

OSI standards are dominant at the physical and data link layers. TCP/IP is dominant at the internet and transport layers.

TCP/IP internet and transport layer standards, in turn, govern transmission across an entire internet, ensuring that any two host computers can communicate. TCP/IP application standards ensure that the two application programs on the two hosts can communicate as well. TCP/IP is dominant in the internetworking layers (internet and transport).[6]

TCP/IP is dominant in corporate networking at the internet and transport layers.

Test Your Understanding

21. a) Which of the following is an architecture: TCP/IP, TCP, or IP? b) Which of the following are standards: TCP/IP, TCP, or IP? c) What is the standards agency for TCP/IP? d) Why have this agency's standards been so successful? e) What are most of this agency's documents called? f) At which layers is TCP/IP dominant?

The Application Layer

OSI is completely dominant at the physical and data link layers, and TCP/IP is very dominant at the internet and transport layers. What about the application layer? The answer here is complex.[7] Overall, it seems best to say that no standards agency or architecture dominates at the application layer, although the IETF is particularly strong, especially for popular standards such as e-mail and file transfer.

Although many applications do not come from the IETF, they almost all run over TCP/IP standards at the internet and transport layers. This is true even for standards created by ISO and ITU–T.

[6]A few old legacy systems use other standards architectures at the internet and transport layers, but this is a minor concern in most firms today. The most important of these legacy architectures is IBM's Systems Network Architecture, which links large mainframe computers to terminals.

[7]Many application protocols come from the IETF. These include such popular standards as e-mail protocols (SMTP, POP, IMAP, etc.), the FTP standards, and the Simple Network Management Protocol (SNMP) in network management.

Some application standards come directly from OSI. This is particularly true of graphics file format standards. However, many OSI standards were too complex for widespread use. The IETF then produced simpler versions of these OSI standards. A good example is the Lightweight Directory Access Protocol (LDAP) for access to directory servers. LDAP evolved (devolved?) from the OSI directory access protocol standard.

Other standards agencies also produce application layer standards. HTTP and HTML standards come from the World Wide Web Consortium (W3C), although the IETF is producing some WWW standards. Most confusingly, incompatible Service Oriented Architecture Web services standards are being produced by several competing standards agencies.

At the application layer, as noted earlier, there is growing cooperation between ISO, the ITU–T, and the IETF. In voice over IP, the ITU–T and the IETF have harmonized several key standards. ISO and the IETF, in turn, have been cooperating in file format standards.

Test Your Understanding

22. a) Is any standards architecture dominant at the application layer? b) Do almost all applications, regardless of what standards architecture they come from, run over TCP/IP standards at the internet and transport layers?

TCP/IP *and* OSI: The Hybrid TCP/IP–OSI Standards Architecture

We have noted neither TCP/IP nor OSI dominates at all layers. Consequently, as discussed earlier, the most common standards pattern in organizations is to use OSI standards at the physical and data link layers, TCP/IP standards at the internet and transport layers, and application standards from several sources. This is very important for you to keep in mind because this **hybrid TCP/IP–OSI standards architecture** in Figure 2-18 forms the basis for most of this book.

The most common standards pattern in organizations is to use OSI standards at the physical and data link layers, TCP/IP standards at the internet and transport layers, and application standards from several sources. This is the hybrid TCP/IP–OSI standards architecture.

Test Your Understanding

23. a) Which layers of the hybrid TCP/IP–OSI standards architecture use OSI standards? b) Which layers use TCP/IP standards? c) Do wireless LAN standards come from OSI or TCP/IP? Explain. (The answer is not explicitly in this section.) d) Do switched WAN standards come from OSI or TCP/IP? Explain. (Again, the answer is not explicitly in this section.)

Higher Layers in OSI

In the section on OSI, we noted that OSI standards are dominant at the physical and data link layers but are not widely used at higher layers. This box discusses higher layers in OSI.

OSI Network and Transport Layers

The network layer functionality of OSI corresponds closely to the internet layer functionality of TCP/IP that we saw earlier in this chapter. The transport layer functionality of OSI, in turn, is very similar to the transport layer functionality of TCP/IP. However, while *functionality* may be similar between OSI and TCP at these layers, actual OSI and TCP/IP *standards* at these layers

are completely incompatible. More importantly, OSI standards are rarely used at the network or transport layers by real organizations.[8]

OSI Session Layer

The **OSI session layer (OSI Layer 5)** initiates and maintains a connection between application programs on different computers. For instance, suppose that a single transaction requires a number of messages. If there is a connection break, the transmission can begin at the last session layer rollback point instead of restarting at the beginning. For example, if communication fails during a database transaction, the entire transaction does not have to be done over—only the work since the last rollback point. Unfortunately, while this is good for database applications and a few other applications, few applications benefit from the overhead added by the OSI session layer.

OSI Presentation Layer

The **OSI presentation layer (OSI Layer 6)** is *designed* to handle data-formatting differences between the two computers. For example, most computers format character data (letters, digits, and punctuation signs) in the ASCII code. In contrast, IBM mainframes format them in the EBCDIC code. The presentation layer can handle this and other format translations.

The OSI presentation layer also is *designed* to be used for compression and data encryption for application data.

In practice, the presentation layer is rarely used for either data format conversion or compression and encryption. Rather, the presentation layer has become a category for general application file format standards used in multiple applications, including MP3, JPEG, and many other general OSI file format standards.

OSI Application Layer

The **OSI application layer (OSI Layer 7)** governs remaining application-specific matters that are not covered by the session and presentation layers. The OSI application layer, freed from session and presentation matters, focuses on concerns specific to the application in use.

Test Your Understanding

24. a) At which layers do OSI standards dominate usage? b) Name and describe the functions of OSI Layer 5. c) Name and describe the intended use of OSI Layer 6. d) How is the OSI presentation layer actually used? e) Beginning with the physical layer (Layer 1), give the name and number of the OSI layers.

[8] Although OSI physical and data link layer standards are dominant, and while many OSI application layer standards are used, almost no systems implement OSI standards at the network, transport, session, or presentation layers. Then why, you may ask, do you need to be able to describe these layers? The answer is that you need to get a job. In a very large percentage of all job interviews, an interviewer noting that you have taken a networking course will ask you to describe the OSI layers. We kid you not.

CONCLUSION

Synopsis

In this chapter, we looked broadly at standards. Most of this book (and the networking profession in general) will focus on standards, which are also called protocols. Standards govern message exchanges. More specifically, they place constraints on message semantics (meaning), message syntax (format), and message order.

Standards are connection-oriented or connectionless. In connection-oriented protocols, there is a distinct opening before content messages are sent and a distinct closing afterward. There also are sequence numbers, which allow fragmentation and are used in supervisory messages (e.g., acknowledgments) to refer to specific messages. In connectionless protocols, there are no such openings and closings. Connectionless protocols are simpler than connection-oriented protocols, but they lose the advantages of sequence numbers.

In turn, reliable protocols do error correction, while unreliable protocols do not. Although unreliable protocols may do error detection without error correction, this does not make them reliable. In general, standards below the transport layer are unreliable in order to reduce costs. The transport standard usually is reliable; this allows error correction processes on just the two hosts to correct errors at the transport layer and at lower layers, giving the application clean data. Figure 2-23 compares the main protocols we have seen in this chapter in terms of connection orientation and reliability.

To discuss message ordering in more detail, we looked at HTTP and TCP. Message ordering in HTTP is trivial. The browser must initiate the communication by sending an HTTP request message; afterward the webserver program may transmit. TCP, in contrast, has complex message ordering. A three-step handshake is needed to open a connection, and four messages are needed to close a connection. Correctly received TCP messages (called segments) are always acknowledged by the receiver. If the sender does not receive an acknowledgment promptly, it retransmits the unacknowledged segment. This gives reliability.

To discuss message syntax in more detail, we looked briefly at the syntax of Ethernet frames, IP packets, TCP segments, UDP datagrams, and HTTP request and response messages. We saw that they represent syntax in three different ways. We will be looking at the syntax of many messages in this course, so you should be familiar with all methods for representing syntax. In the discussion, we saw that octet is another name for byte. We also saw that application programs on multitasking servers are usually

Layer	Protocol	Connection-Oriented or Connectionless?	Reliable or Unreliable?
5 (Application)	HTTP	Connectionless	Unreliable
4 (Transport)	TCP	**Connection-oriented**	**Reliable**
4 (Transport)	UDP	Connectionless	Unreliable
3 (Internet)	IP	Connectionless	Unreliable
2 (Data Link)	Ethernet	Connectionless	Unreliable

FIGURE 2-23 Characteristics of the Protocols Discussed in This Chapter

represented by well-known port numbers, while clients use ephemeral port numbers to represent conversations with servers. A socket consists of an IP address, a colon, and a port number. It represents a particular program (or conversation) on a particular host.

The application layer must convert text, graphics, video, and other application layer content into bits (1s and 0s). In this chapter, we looked at how application programs encode ASCII text, whole numbers, a number of alternatives, and voice and video streams into strings of bits.

We looked at how layer processes work together on the source host. After each layer creates its message, it immediately passes the message down to the next-lower-layer process. The data link, internet, and transport processes take every message they are given and encapsulate it in a message suitable for that layer.

Individual standards are not created in isolation. They are created within broad plans called standards architectures. Only after the broad architecture is designed by specifying the functionality of each layer and ensuring that the architecture overall allows interoperability are individual standards developed.

The TCP/IP architecture has the Internet Engineering Task Force (IETF) as its standards agency. It has a four-layer architecture. The bottom layer basically says, "Use OSI standards for single switched and wireless networks."

The OSI architecture has ISO and ITU–T as its two standards agencies. OSI has a seven-layer architecture. The session layer manages a connection between application programs; if there is an interruption, only communication since the last "rollback point" needs to be repeated. The presentation layer was designed as a layer for converting between application formatting implementations in different operating systems and for providing encryption and compression. In practice, it has become a catch-all category for application standards, such as JPEG.

Most real corporations today use a hybrid TCP/IP–OSI standards architecture that combines layers from TCP/IP and OSI. This architecture, which Figure 2-20 illustrates, will be the focus of this book. At the physical and data link layers, which govern transmission through a single switched or wireless network, OSI standards are almost always used. TCP/IP standards are dominant at the internet and transport layers. Application layer standards come from many sources. OSI and TCP/IP standards agencies frequently collaborate to create application layer standards.

END-OF-CHAPTER QUESTIONS

Thought Questions

1. How do you think TCP would handle the problem if an acknowledgment were lost, so that the sender retransmitted the unacknowledged TCP segment, therefore causing the receiving transport process to receive the same segment twice?
2. a) In Figure 2-13, what will be the value in the destination port number field if a packet arrives for the e-mail application? b) When the HTTP program sends an HTTP response message to a client PC, in what field of what message will it place the value 80?
3. Binary for 47 is 101111. Give the binary for 48, 49, and 50.
4. You need to represent 1,026 different city names. How many bits will this take if you give each city a different binary number?

Brainteaser Questions

1. How can you make a connectionless protocol reliable? (You may not be able to answer this question, but try.)
2. Spacecraft exploring the outer planets need reliable data transmission. However, the acknowledgments would take hours to arrive. This makes an ACK-based reliability approach unattractive. Can you think of another way to provide reliable data transmission to spacecraft? (You may not be able to answer this question, but try.)

Perspective Questions

1. What was the most surprising thing you learned in this chapter?
2. What was the most difficult material for you in this chapter?

2a HANDS-ON: WIRESHARK PACKET CAPTURE

LEARNING OBJECTIVES

By the end of this chapter, you should be able to:

- Use the Wireshark packet capture program at a novice level.
- Capture packets in real time.
- Analyze the packets at a novice level.

INTRODUCTION

A good way to practice what you have learned in this chapter is to look at individual packets. Packet capture programs record packets going into and out of your computer. If you capture a brief webserver interaction, you can look at header fields, TCP three-step connection starts, and other information. There are several good packet capture programs. We will look at Wireshark, which is simple to use, popular, and free to download (at least at the time of this writing).

GETTING WIRESHARK

To get Wireshark, go to wireshark.org. Do *not* go to wireshark.com. Follow the instructions and download the program on your computer.

USING WIRESHARK

Getting Started

After installation, open the Wireshark program. You will see the opening screen. It will look like the screen in Figure 2a-1. There will be controls at the top with a blank area below them. You will soon fill this area with your packet capture.

Starting a Packet Capture

To start a packet capture, click on the *Go* menu item. Then, when the Wireshark: Capture Interfaces dialog appears, as Figure 2a-2 illustrates, select a network interface and click on *Start*.

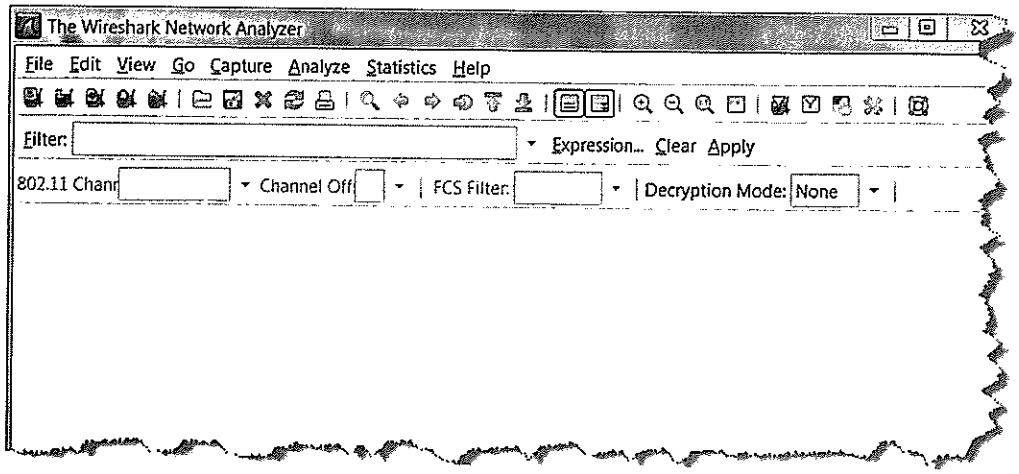


FIGURE 2a-1 Initial Wireshark Screen

Source: Wireshark Foundation.

Getting Data

Your browser should already be open. Switch to your browser and enter a URL. (In this example, the author went to Wikipedia.org.) This creates a flurry of packets between you and the host specified in the URL. These appear on the window below the controls, as shown in Figure 2a-3.

Stopping Data Collection

To stop the data collection, click on the *Capture* menu item, as Figure 2a-4 shows. When the drop-down menu appears, select *Stop*. You now have a packet stream to analyze.

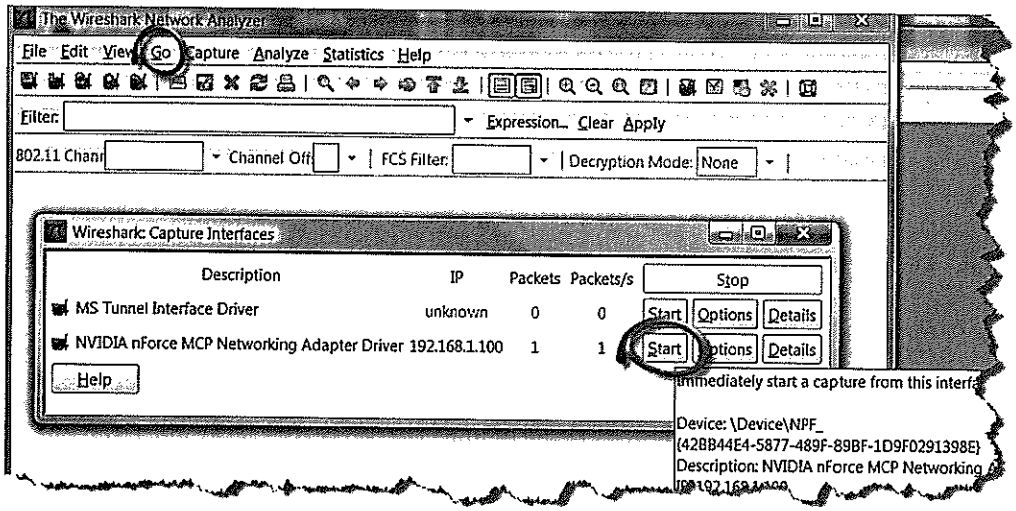


FIGURE 2a-2 Starting a Packet Capture in Wireshark

Source: Wireshark Foundation.

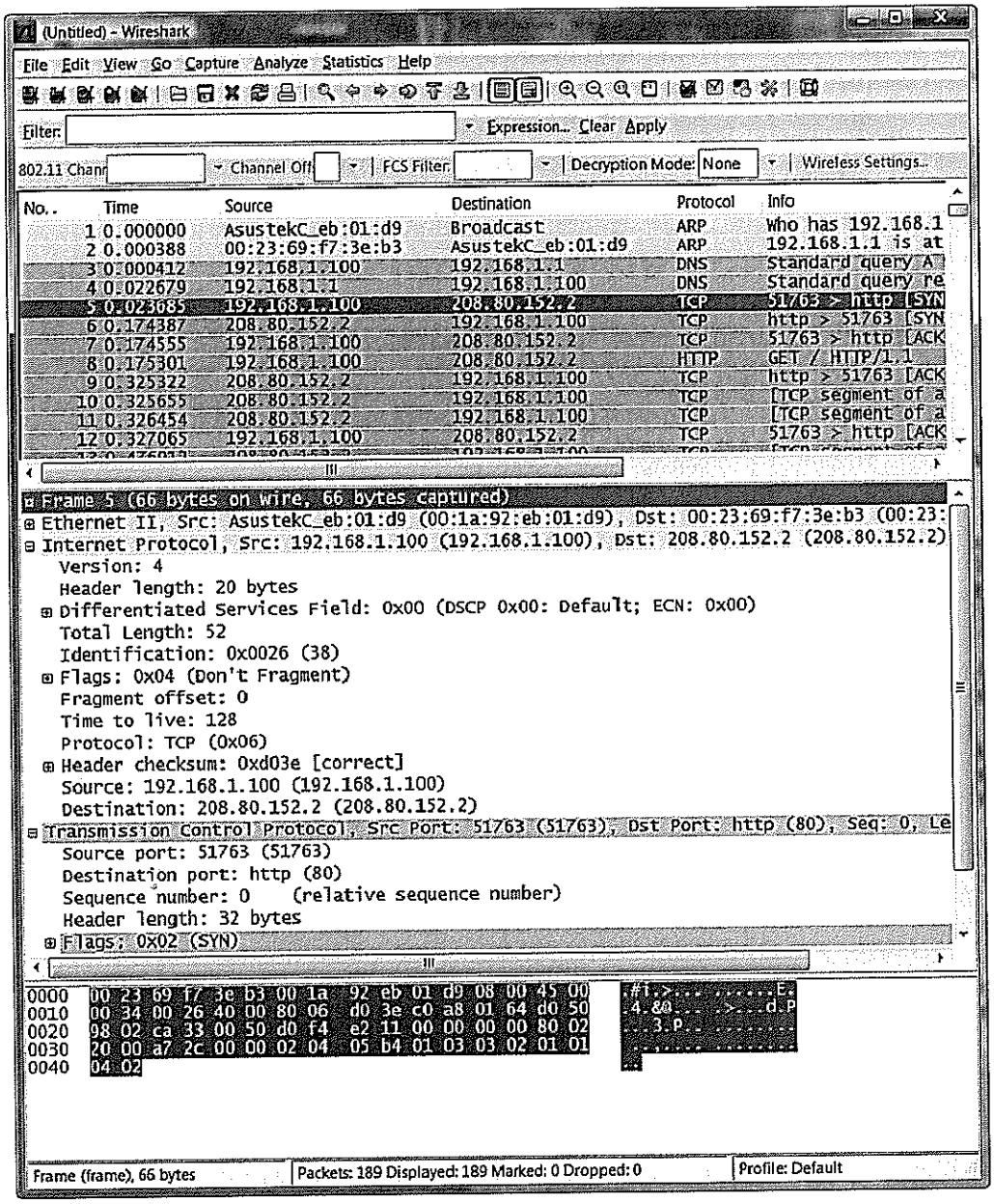


FIGURE 2a-3 Collecting Data

Source: Wireshark Foundation.

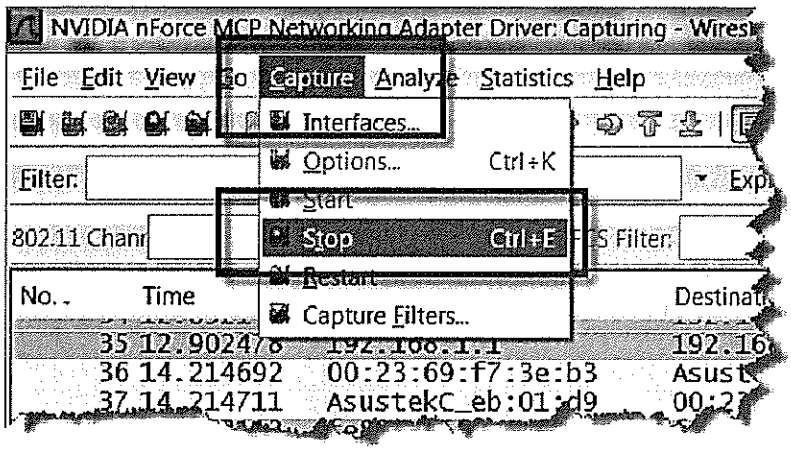


FIGURE 2a-4 Stopping the Data Collection

Source: Wireshark Foundation.

Looking at Individual Packets

Now you can begin looking at individual packets. To see how to do this, look again at Figure 2a-3.

PACKET SUMMARY WINDOW In the upper window in the display area, you can see the packets one at a time. The capture begins with two ARP packets, which we will discuss when we get to the TCP/IP chapters.

Then come two DNS packets. In the example, the author typed the host name Wikipedia.org in the URL. The author's computer (192.168.1.100) sent a DNS request message to its DNS server to get the IP address for Wikipedia.org. The DNS sent back the requested IP address.

Now, the author's computer opened a connection to 208.80.152.2, which is Wireshark.org's IP address.[1] It first sent a TCP SYN segment to 208.80.152.2. This is Frame 5. In Figure 2a-3, the frame has been selected.

Information about the contents of this particular frame is shown in a window below the window showing each frame on a single line. First, the window shows information on the Ethernet header and trailer. Next comes information about the IP packet, followed by information about the TCP SYN segment contained in the packet.

WINDOW WITH DETAILED INFORMATION ON THE SELECTED PACKET The Ethernet information has been minimized. Only the source and destination MAC addresses are shown. However, information about the IP packet has been maximized. You can see the values of the individual fields in the selected packet. For example, note that the Time to Live field in this packet had the value 128. In addition, the protocol field value indicates that the data field contains a TCP segment.

[1]If you try this, you may get a different IP address. Many firms have multiple physical webservers that they associate with a host name. A DNS response message returns the IP address of one of these physical servers.

The TCP segment information is also expanded, although only the first few fields are shown in the window. Note that the destination port is 80, indicating that the author was contacting the Wireshark.org webserver. Note also that the Flag Fields information says that the SYN bit is set, as one would expect.

To make life easier for you, Wireshark does as much translation as possible. For example, it interprets the information in the protocol field as indicating that there is a TCP segment in the packet's data field. It also indicates that Port 80 is HTTP.

The information on sequence number is highly simplified compared to the discussion in Chapter 2. This is the first TCP segment being sent. It is given the value 0 rather than its complex real value.

HEX WINDOW The lowest window shows the contents of the packet in hexadecimal (Base 16) format. Hex is difficult for new analysts to interpret, but it is very compact compared to the information in the middle window. Experienced packet analysts quickly learn the positions of important fields and learn to read the hex symbols for that field.

Options

Figure 2a-5 shows that Wireshark capture options allow you to control what packets are captured. If you are connected to multiple external servers simultaneously, this can allow you to capture only packets for a particular connection.

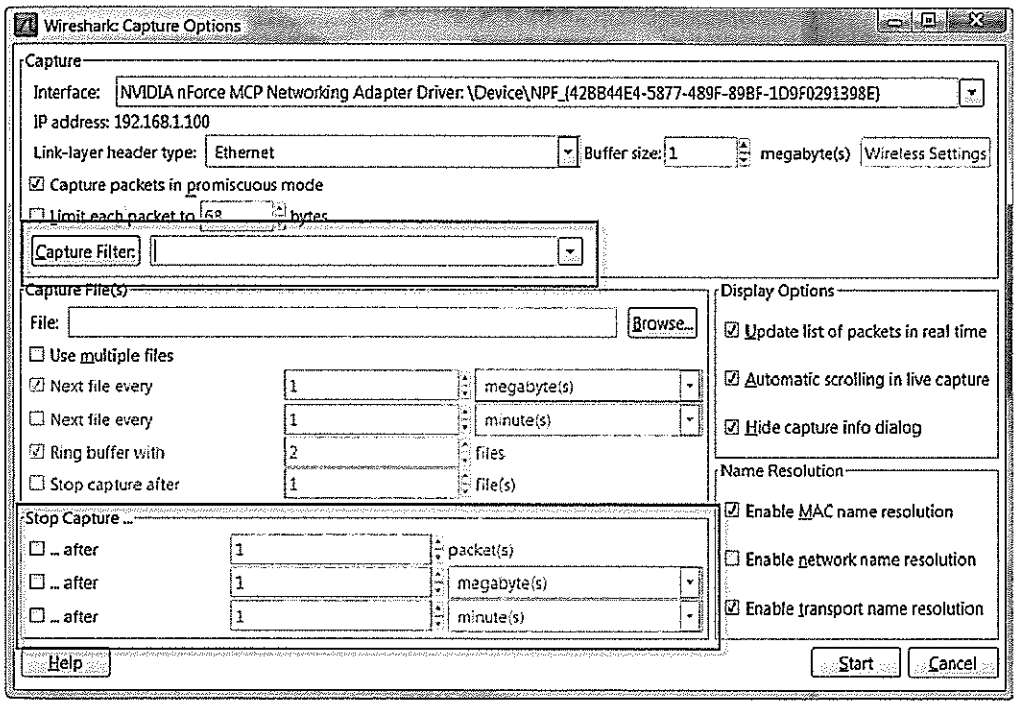


FIGURE 2a-5 Wireshark Options

Source: Wireshark Foundation.

Exercises

1. Do the following:
   - Download Wireshark.
   - Start Wireshark.
   - Turn on Wireshark capture.
   - Type a URL in your browser window (not Wikipedia.org).
   - After a few seconds, stop the capture.
   - Answer the following questions:
     1a. What URL did you use? What was the IP address of the webserver?
     1b. Find the frame in which your PC sent the SYN packet. List the source and destination IP address, the source and destination port numbers, and the header checksum.
     1c. Select the SYN/ACK packet. List the source and destination IP address, the source and destination port numbers, and the header checksum.
     1d. Select the packet that acknowledges the SYN/ACK segment. List the source and destination IP address, the source and destination port numbers, and the header checksum.
2. Change the options so that only packets you send are recorded. Do a capture. Click on the window containing Wireshark and hit *Alt-Enter*. This captures the window to your clipboard. Paste it into your homework.