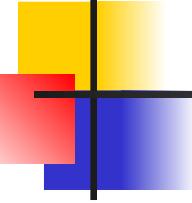


*Standard
Client-Server
Protocols*



1 *WORLD-WIDE WEB AND HTTP*

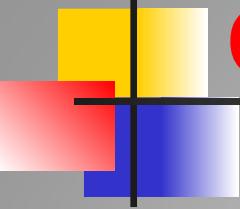
2 *FTP*

3 *ELECTRONIC MAIL*

4 *TELNET*

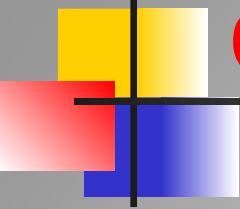
5 *SECURE SHELL (SSH)*

6 *DOMAIN NAME SYSTEM (DNS)*



Objective

- *The first section introduces the World Wide Web. It then discusses the HyperText Transfer Protocol, the most common client-server application program used in relation to the World Wide Web.*
- *The second section discusses the File Transfer Protocol, which is the standard protocol provided by TCP/IP for copying a file from one host to another.*
- *The third section discusses electronic mail, which involves two protocols: SMPT and POP. As we will see, the nature of this application is different from the other two previous applications. We need two different protocols to handle electronic mail.*



Objective (continued)

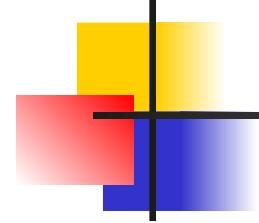
- *The fourth section discusses TELNET, a general client-server program that allows users to log in to a remote machine and use any application available on the remote host.*

- *The fifth section discusses Secure Shell, which can be used as a secured TELNET, but it can also provide a secure tunnel for other applications.*

- *The sixth section talks about the Domain Name System, which acts as the directory system in the Internet. It maps the name of an entity to its IP address.*

WORLD WIDE WEB AND HTTP

In this section, we first introduce the World Wide Web (abbreviated WWW or Web). We then discuss the Hyper-Text Transfer Protocol (HTTP), the most common client-server application program used in relation to the Web.



World Wide Web

The idea of the Web was first proposed by Tim Berners-Lee in 1989 at CERN, the European Organization for Nuclear Research, to allow several researchers at different locations throughout Europe to access each others' researches. The commercial Web started in the early 1990s.

Example 26.1

Assume we need to retrieve a scientific document that contains one reference to another text file and one reference to a large image. Figure 26.1 shows the situation.

The main document and the image are stored in two separate files in the same site (file A and file B); the referenced text file is stored in another site (file C). Since we are dealing with three different files, we need three transactions if we want to see the whole document.

Figure 26.1: Example 26.1

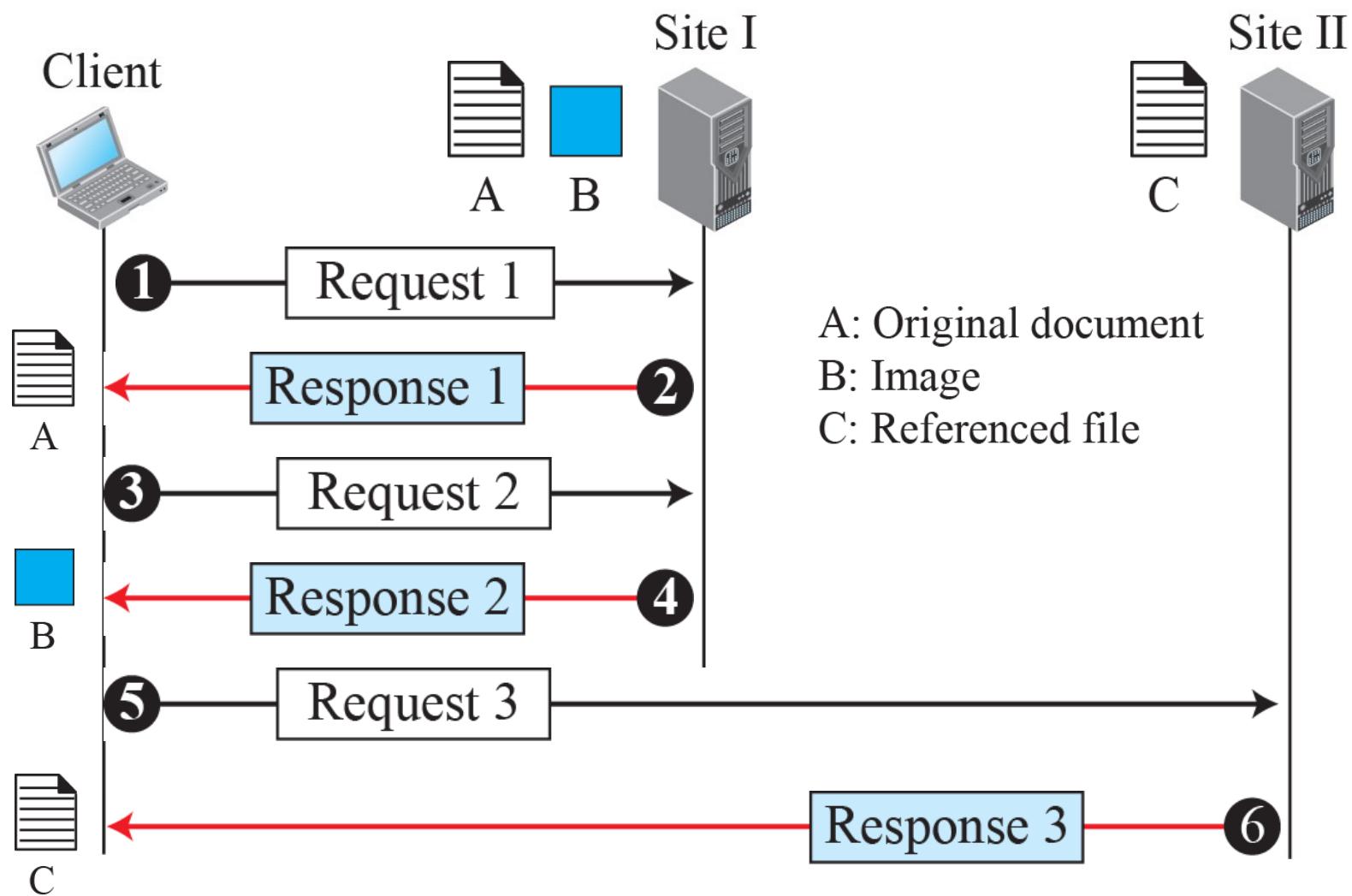
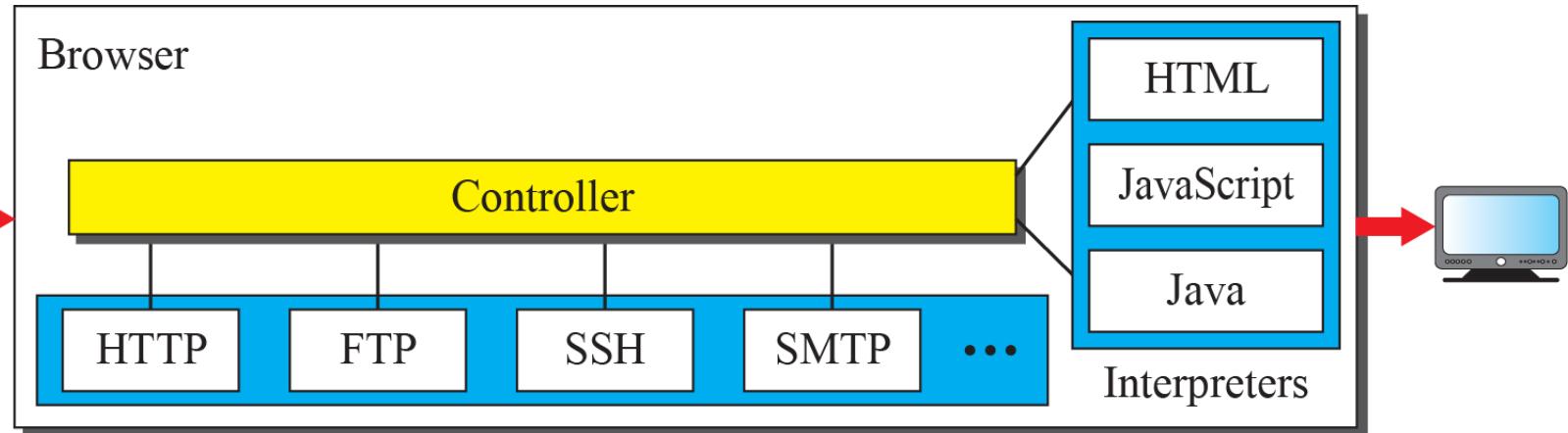
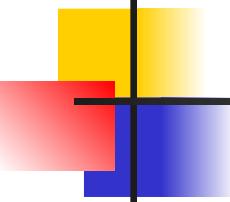


Figure 26.2: Browser



Example 26.2

The URL <http://www.mhhe.com/compsci/forouzan/> defines the web page related to one of the computer in the McGraw-Hill company (the three letters www are part of the host name and are added to the commercial host). The path is *compsci/forouzan/*, which defines Forouzan's web page under the directory *compsci* (computer science).



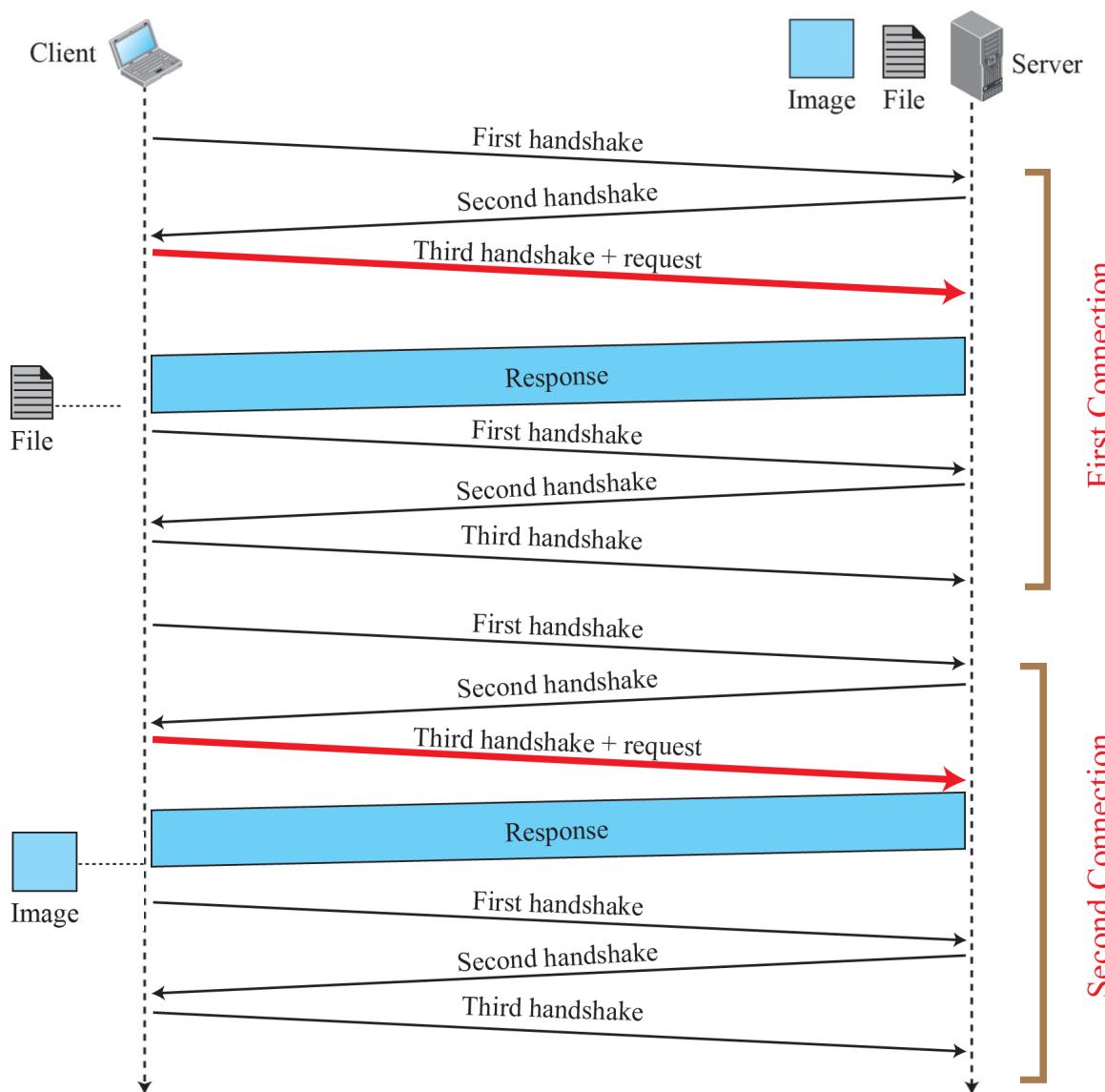
HyperText Transfer Protocol

The HyperText Transfer Protocol (HTTP) is used to define how the client-server programs can be written to retrieve web pages from the Web. An HTTP client sends a request; an HTTP server returns a response. The server uses the port number 80; the client uses a temporary port number. HTTP uses the services of TCP, which, as discussed before, is a connection-oriented and reliable protocol.

Example 26.3

Figure 26.3 shows an example of a *nonpersistent* connection. The client needs to access a file that contains one link to an image. The text file and image are located on the same server. Here we need two connections. For each connection, TCP requires at least three handshake messages to establish the connection, but the request can be sent with the third one. After the connection is established, the object can be transferred. After receiving an object, another three handshake messages are needed to terminate the connection, as we will see in Chapter 23.

Figure 26.3: Example 26.3



Example 26.4

Figure 26.4 shows the same scenario as in Example 26.3, but using a persistent connection. Only one connection establishment and connection termination is used, but the request for the image is sent separately.

Figure 26.4: Example 26.4

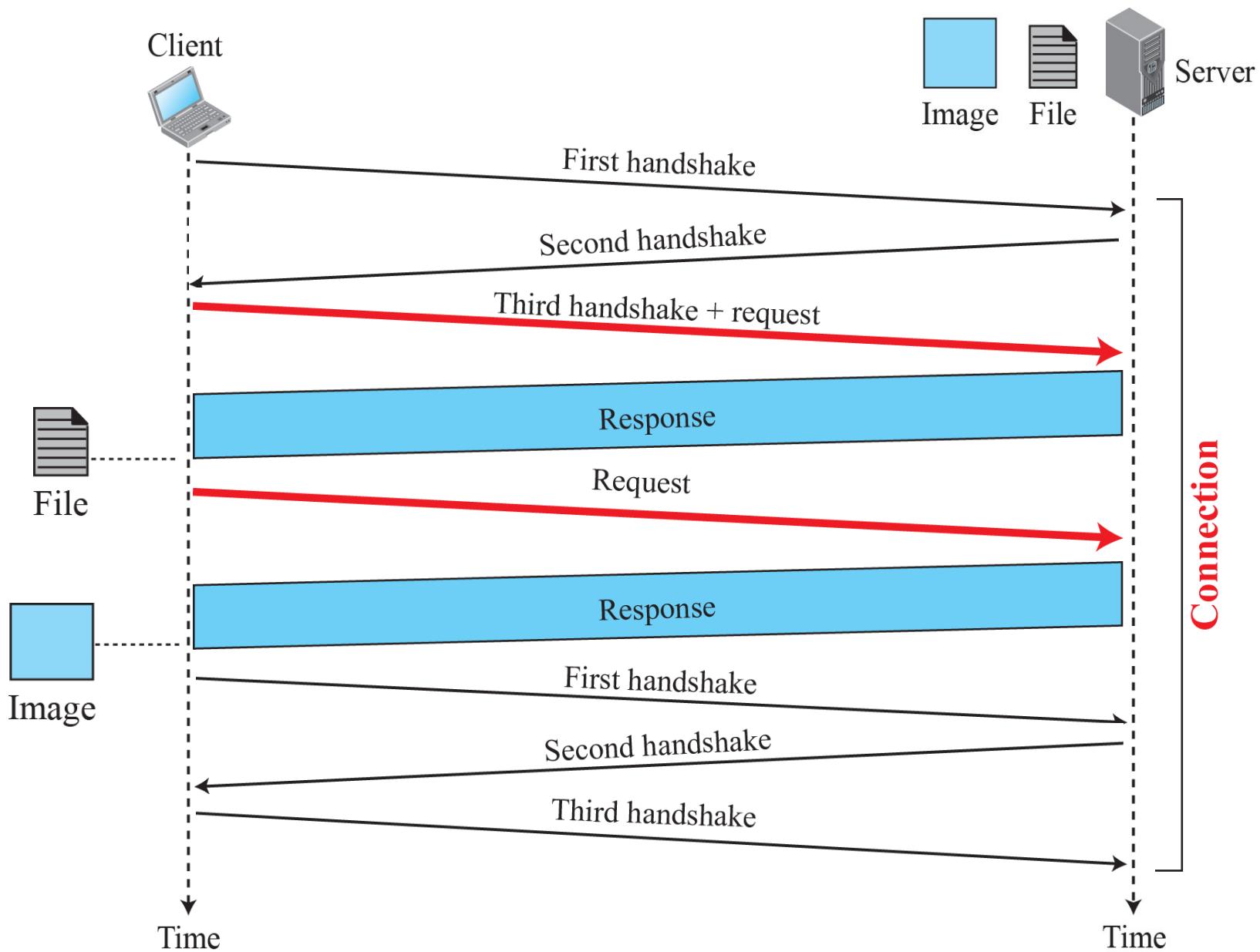
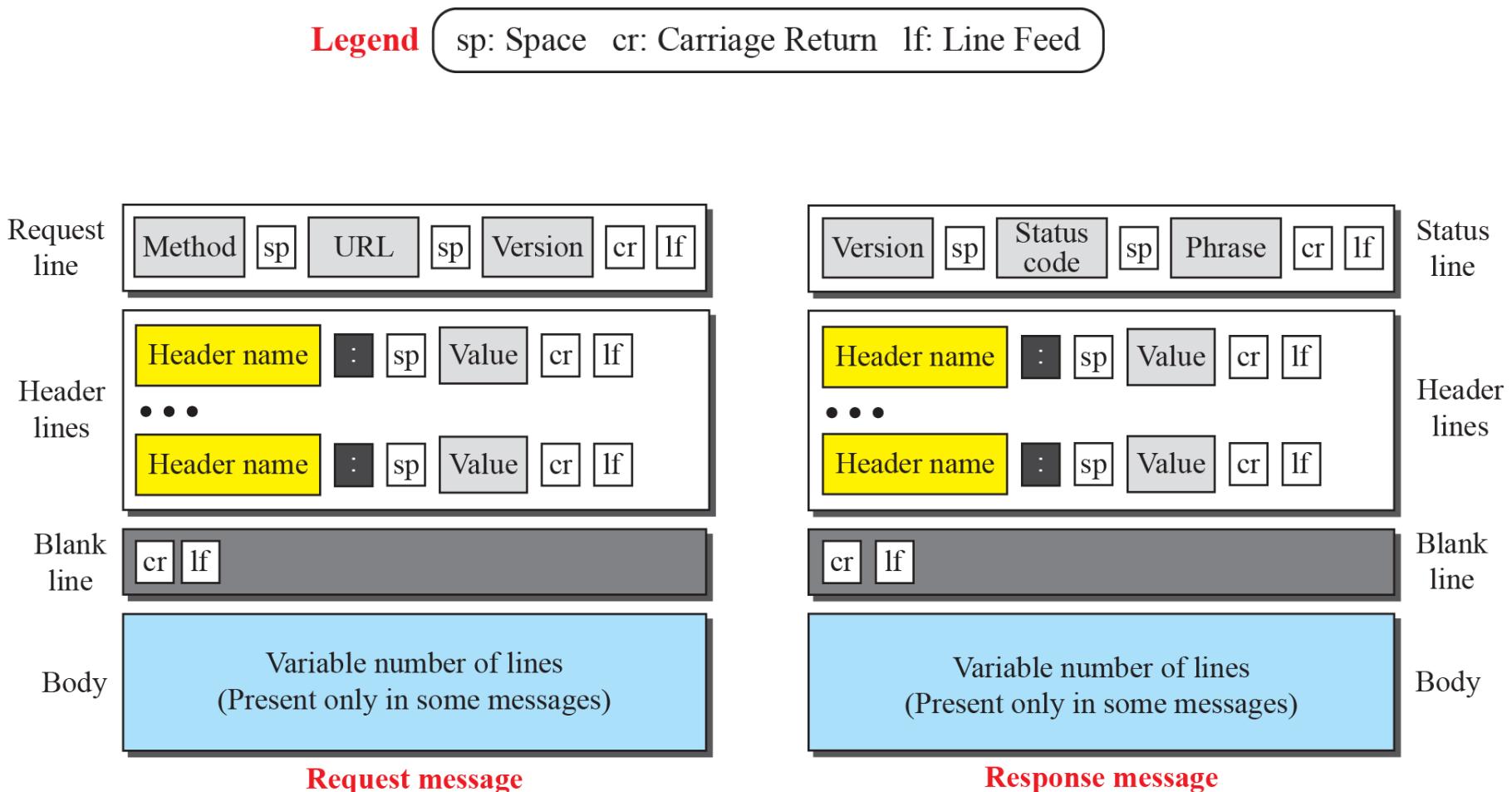


Figure 26.5: Formats of the request and response messages



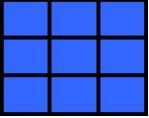


Table 26.1: Methods

<i>Method</i>	<i>Action</i>
GET	Requests a document from the server
HEAD	Requests information about a document but not the document itself
PUT	Sends a document from the client to the server
POST	Sends some information from the client to the server
TRACE	Echoes the incoming request
DELETE	Removes the web page
CONNECT	Reserved
OPTIONS	Inquires about available options



Table 26.2: Request Header Names

<i>Header</i>	<i>Description</i>
User-agent	Identifies the client program
Accept	Shows the media format the client can accept
Accept-charset	Shows the character set the client can handle
Accept-encoding	Shows the encoding scheme the client can handle
Accept-language	Shows the language the client can accept
Authorization	Shows what permissions the client has
Host	Shows the host and port number of the client
Date	Shows the current date
Upgrade	Specifies the preferred communication protocol
Cookie	Returns the cookie to the server (explained later)
If-Modified-Since	If the file is modified since a specific date

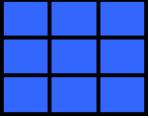


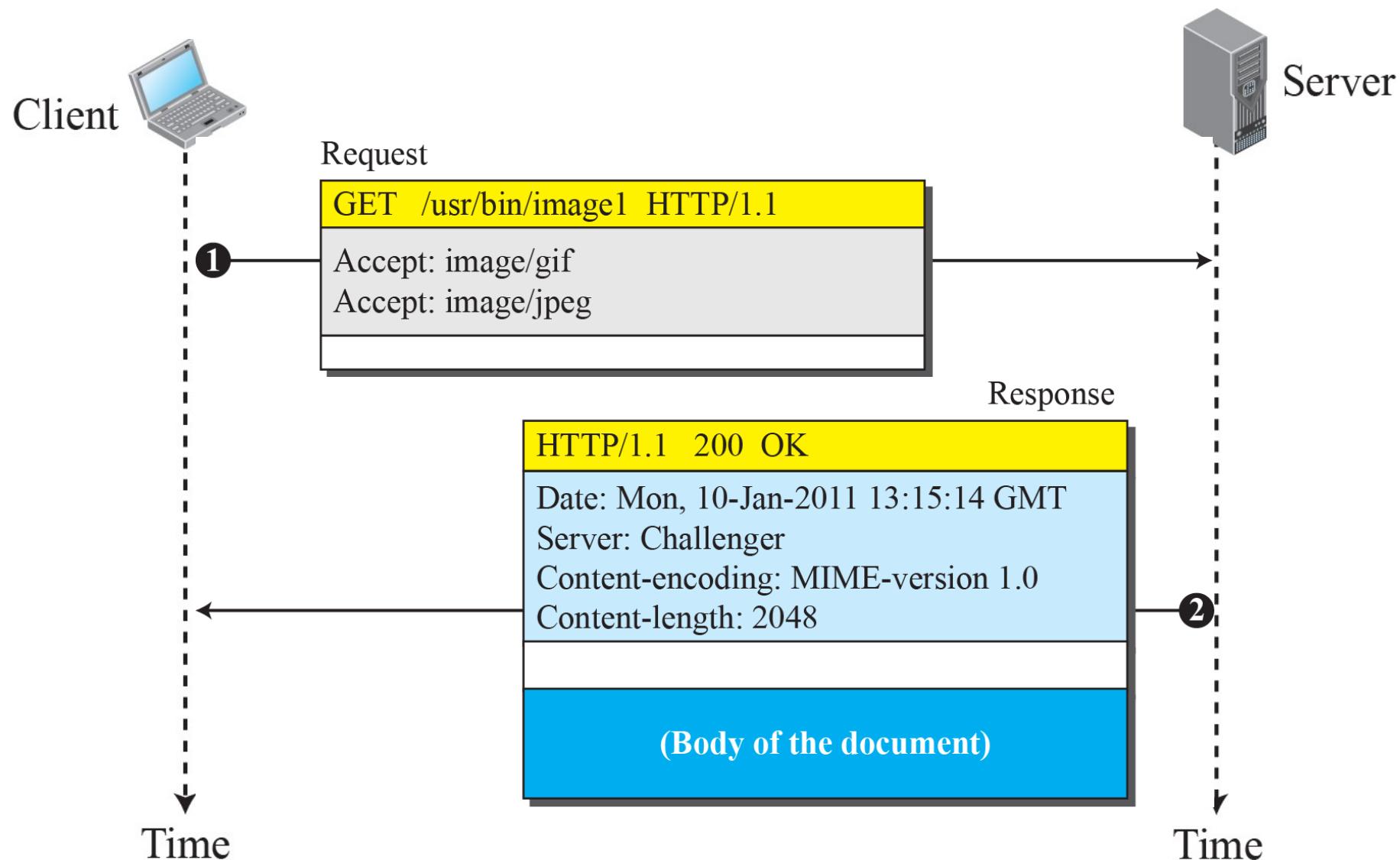
Table 26.3: Response Header Names

<i>Header</i>	<i>Description</i>
Date	Shows the current date
Upgrade	Specifies the preferred communication protocol
Server	Gives information about the server
Set-Cookie	The server asks the client to save a cookie
Content-Encoding	Specifies the encoding scheme
Content-Language	Specifies the language
Content-Length	Shows the length of the document
Content-Type	Specifies the media type
Location	To ask the client to send the request to another site
Accept-Ranges	The server will accept the requested byte-ranges
Last-modified	Gives the date and time of the last change

Example 2656

This example retrieves a document (see Figure 26.6). We use the GET method to retrieve an image with the path **/usr/bin/image26**. The request line shows the method (GET), the URL, and the HTTP version (26.1). The header has two lines that show that the client can accept images in the GIF or JPEG format. The request does not have a body. The response message contains the status line and four lines of header. The header lines define the date, server, content encoding (MIME version, which will be described in electronic mail), and length of the document. The body of the document follows the header..

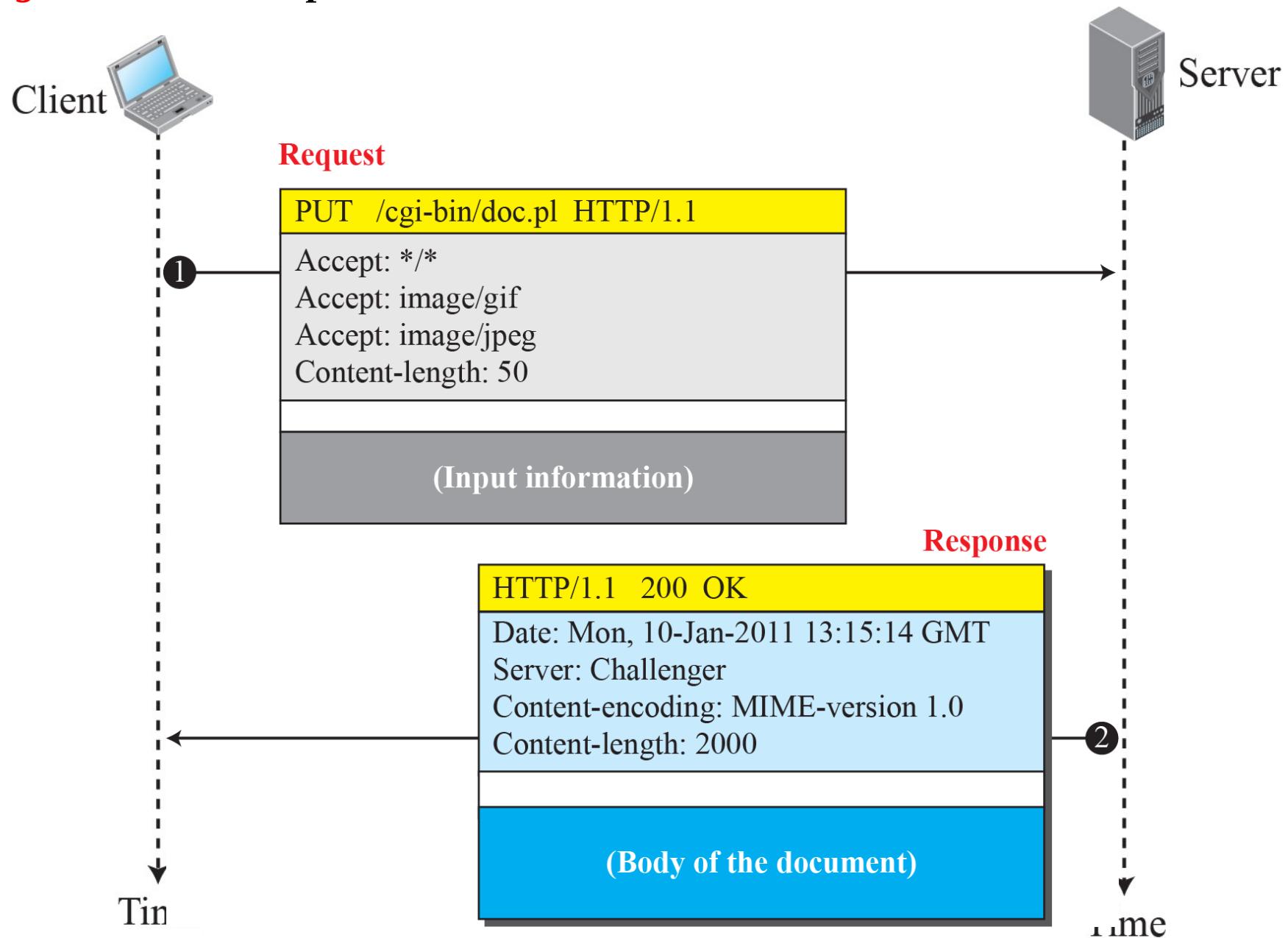
Figure 26.6: Example 26.5



Example 26.6

In this example, the client wants to send a web page to be posted on the server. We use the PUT method. The request line shows the method (PUT), URL, and HTTP version (26.1). There are four lines of headers. The request body contains the web page to be posted. The response message contains the status line and four lines of headers. The created document, which is a CGI document, is included as the body (see Figure 26.7).

Figure 26.7: Example 26.6



Example 26.7

The following shows how a client imposes the modification data and time condition on a request.

GET http://www.commonServer.com/information/file1 HTTP/1.1

Request line

If-Modified-Since: Thu, Sept 04 00:00:00 GMT

Header line

Blank line

The status line in the response shows the file was not modified after the defined point in time. The body of the response message is also empty.

HTTP/1.1 304 Not Modified

Status line

Date: Sat, Sept 06 08 16:22:46 GMT

First header line

Server: commonServer.com

Second header line

(Empty Body)

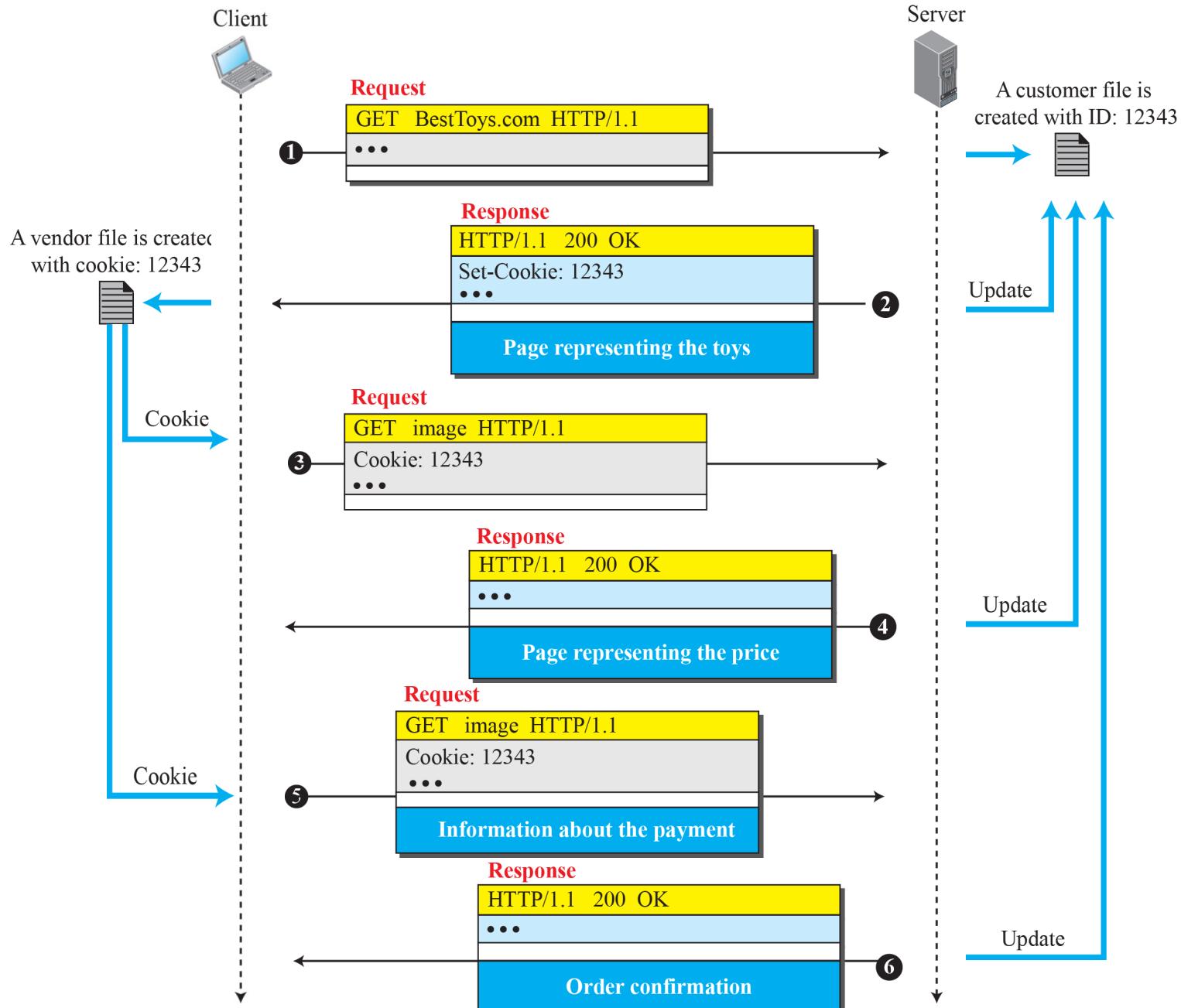
Blank line

Empty body

Example 26.8

Figure 26.8 shows a scenario in which an electronic store can benefit from the use of cookies. Assume a shopper wants to buy a toy from an electronic store named BestToys. The shopper browser (client) sends a request to the BestToys server. The server creates an empty shopping cart (a list) for the client and assigns an ID to the cart (for example, 12343). The server then sends a response message, which contains the images of all toys available, with a link under each toy that selects the toy if it is being clicked. This response message also includes the Set-Cookie header line whose value is 12343. The client displays the images and stores the cookie value in a file named BestToys.

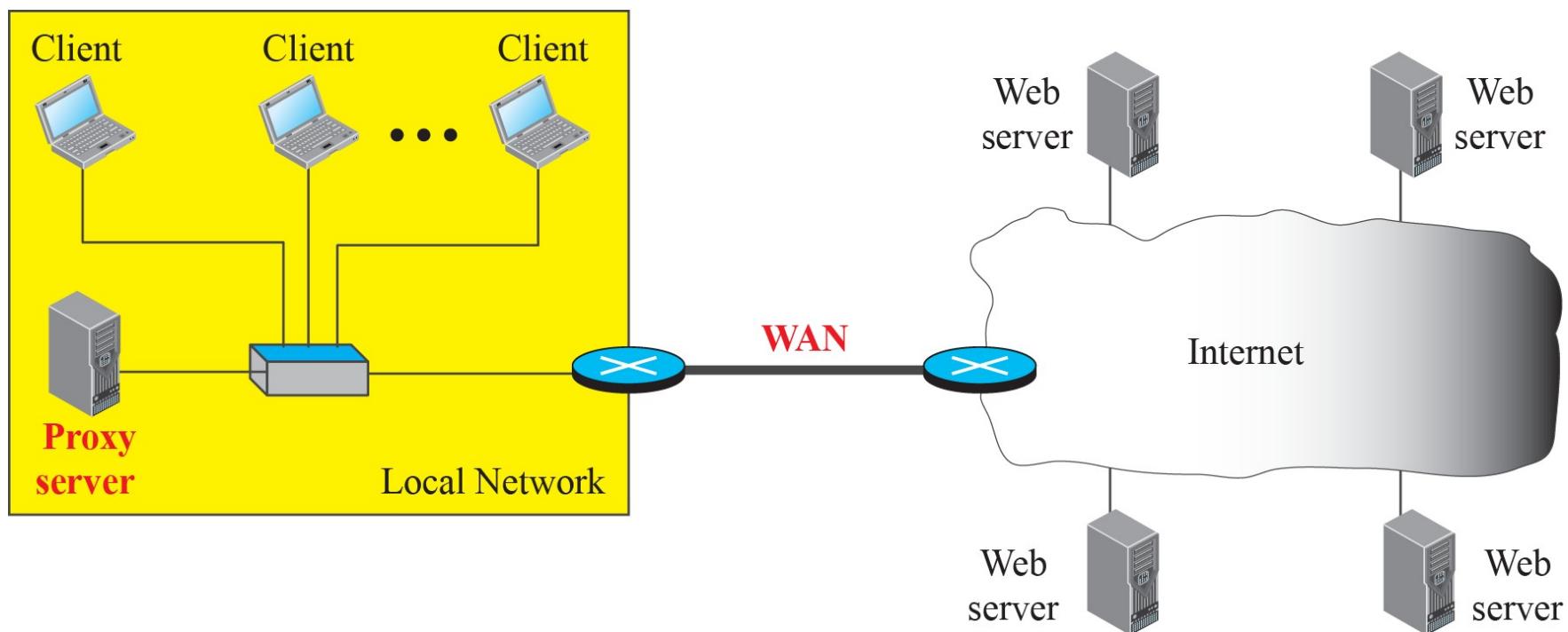
Figure 26.8: Example 26.8



Example 26.9

Figure 26.9 shows an example of a use of a proxy server in a local network, such as the network on a campus or in a company. The proxy server is installed in the local network. When an HTTP request is created by any of the clients (browsers), the request is first directed to the proxy server. If the proxy server already has the corresponding web page, it sends the response to the client. Otherwise, the proxy server acts as a client and sends the request to the web server in the Internet. When the response is returned, the proxy server makes a copy and stores it in its cache before sending it to the requesting client.

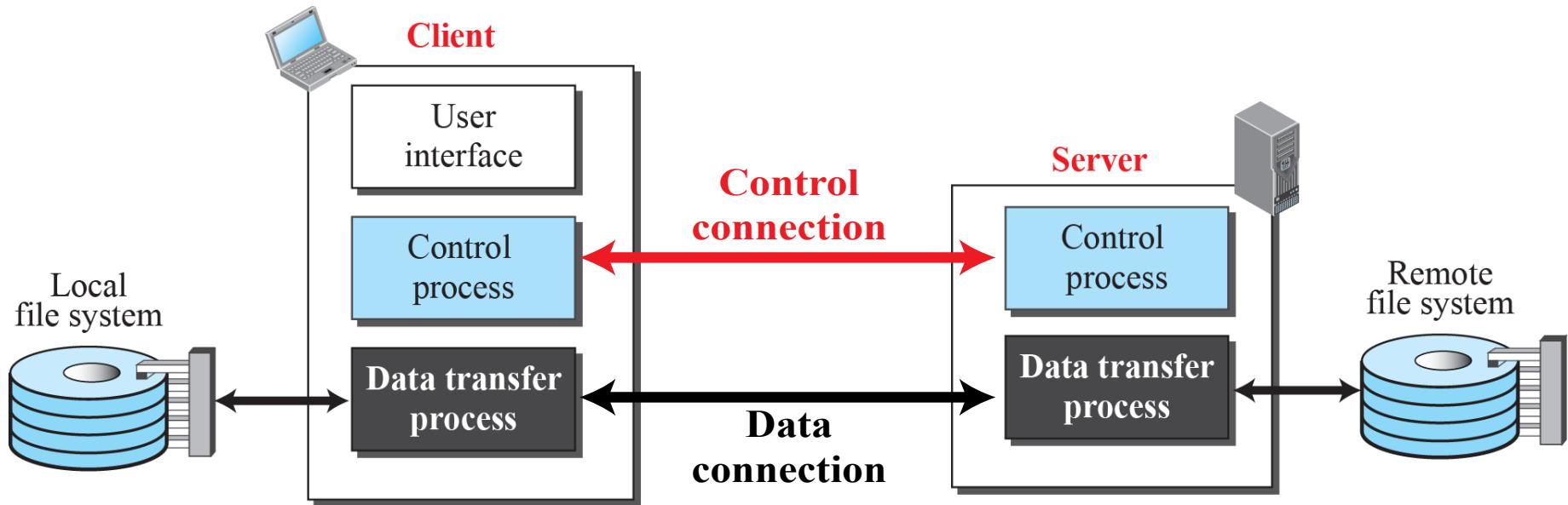
Figure 26.9: Example of a proxy server

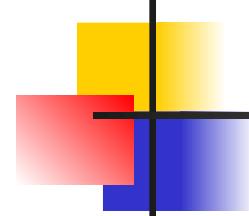


FTP

File Transfer Protocol (FTP) is the standard protocol provided by TCP/IP for copying a file from one host to another. Although transferring files from one system to another seems simple and straightforward, some problems must be dealt with first.

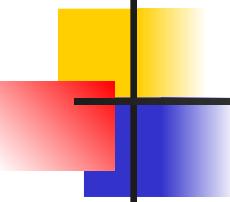
Figure 26.10: FTP





26.2.1 Two Connections

The two connections in FTP have different lifetimes. The control connection remains connected during the entire interactive FTP session. The data connection is opened and then closed for each file transfer activity. It opens each time commands that involve transferring files are used, and it closes when the file is transferred.



26.2.2 Control Connection

For control communication, FTP uses the same approach as TELNET (discussed later). It uses the NVT ASCII character set as used by TELNET. Communication is achieved through commands and responses. This simple method is adequate for the control connection because we send one command (or response) at a time. Each line is terminated with a two-character (carriage return and line feed) end-of-line token.



Table 26.4: Some FTP commands

<i>Command</i>	<i>Argument(s)</i>	<i>Description</i>
ABOR		Abort the previous command
CDUP		Change to parent directory
CWD	Directory name	Change to another directory
DELE	File name	Delete a file
LIST	Directory name	List subdirectories or files
MKD	Directory name	Create a new directory
PASS	User password	Password
PASV		Server chooses a port
PORT	port identifier	Client chooses a port
PWD		Display name of current directory
QUIT		Log out of the system
RETR	File name(s)	Retrieve files; files are transferred from server to client
RMD	Directory name	Delete a directory
RNFR	File name (old)	Identify a file to be renamed

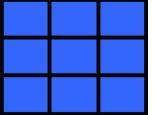


Table 26.4 : Some FTP commands (continued)

<i>Command</i>	<i>Argument(s)</i>	<i>Description</i>
RNTO	File name (new)	Rename the file
STOR	File name(s)	Store files; file(s) are transferred from client to server
STRU	F , R , or P	Define data organization (F : file, R : record, or P : page)
TYPE	A , E , I	Default file type (A : ASCII, E : EBCDIC, I : image)
USER	User ID	User information
MODE	S , B , or C	Define transmission mode (S : stream, B : block, or C : compressed)

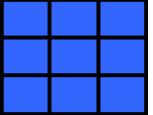
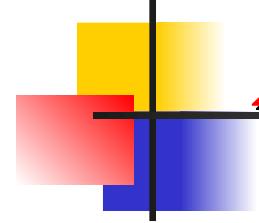


Table 26.5: Some responses in FTP

<i>Code</i>	<i>Description</i>	<i>Code</i>	<i>Description</i>
125	Data connection open	250	Request file action OK
150	File status OK	331	User name OK; password is needed
200	Command OK	425	Cannot open data connection
220	Service ready	450	File action not taken; file not available
221	Service closing	452	Action aborted; insufficient storage
225	Data connection open	500	Syntax error; unrecognized command
226	Closing data connection	501	Syntax error in parameters or arguments
230	User login OK	530	User not logged in



26.2.3 Data Connection

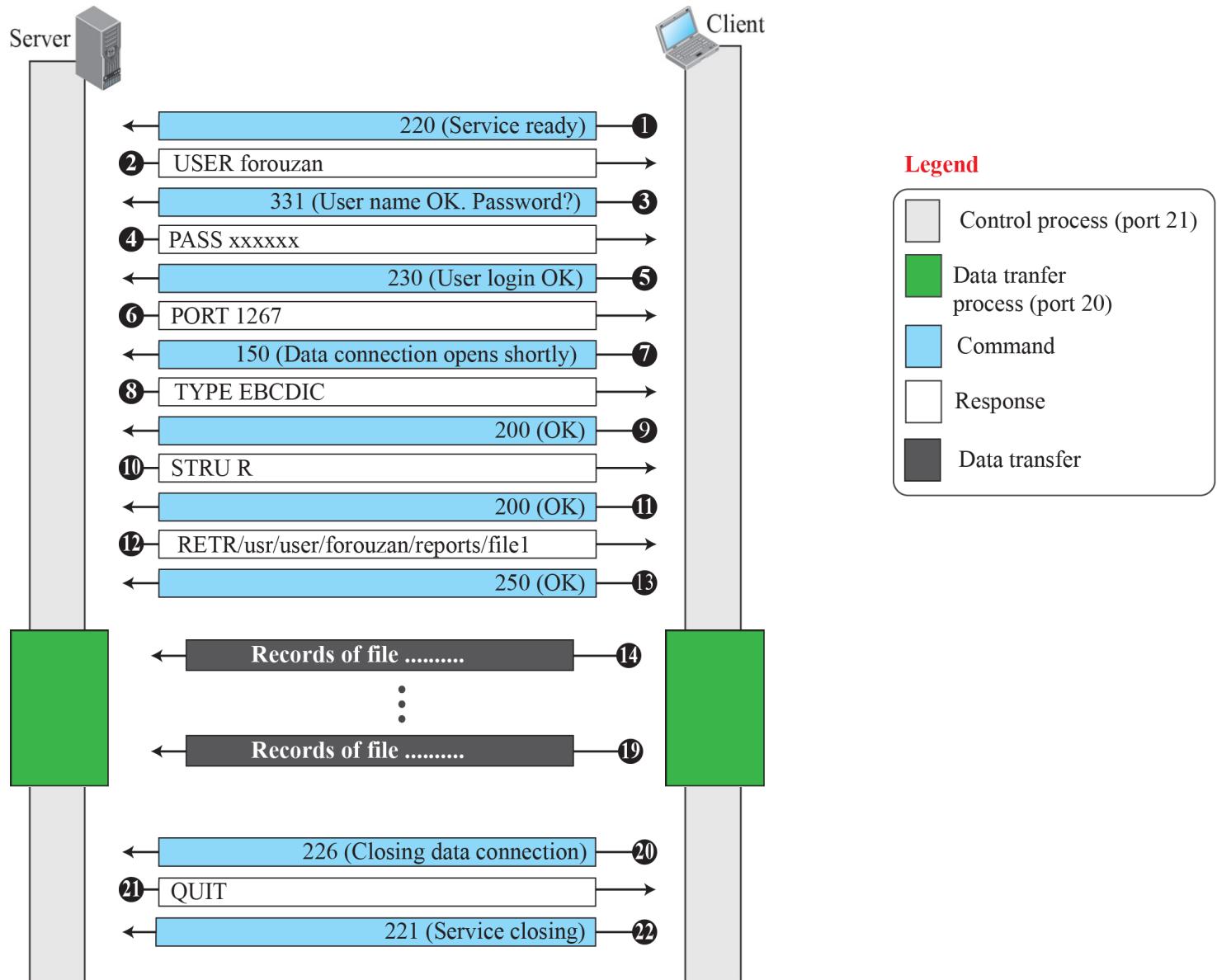
The data connection uses the well-known port 20 at the server site. However, the creation of a data connection is different from the control connection. The following shows the steps:

- 1. The client, not the server, issues a passive open using an ephemeral port.*
- 2. Using the PORT command the client sends this port number to the server.*
- 3. The server receives the port number and issues an active open using the well-known port 20 and the received ephemeral port number.*

Example 26.10

Figure 26.11 shows an example of using FTP for retrieving a file. The figure shows only one file to be transferred. The control connection remains open all the time, but the data connection is opened and closed repeatedly. We assume the file is transferred in six sections. After all records have been transferred, the server control process announces that the file transfer is done. Since the client control process has no file to retrieve, it issues the QUIT command, which causes the service connection to be closed.

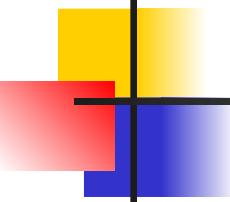
Figure 26.11: Example 26.12



Example 26.11

The following shows an actual FTP session that lists the directories.

```
$ ftp voyager.deanza.fhda.edu
Connected to voyager.deanza.fhda.edu.
220 (vsFTPd 1.2.1)
530 Please login with USER and PASS.
Name (voyager.deanza.fhda.edu:forouzan): forouzan
331 Please specify the password.
Password:*****
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
227 Entering Passive Mode (153,18,17,11,238,169)
150 Here comes the directory listing.
drwxr-xr-x    2      3027     411     4096   Sep 24  2002  business
drwxr-xr-x    2      3027     411     4096   Sep 24  2002  personal
drwxr-xr-x    2      3027     411     4096   Sep 24  2002  school
226 Directory send OK.
ftp> quit
221 Goodbye.
```

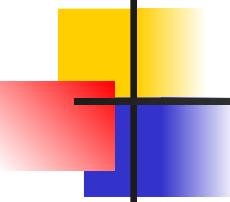


26.2.4 Security for FTP

The FTP protocol was designed when security was not a big issue. Although FTP requires a password, the password is sent in plaintext (unencrypted), which means it can be intercepted and used by an attacker. The data transfer connection also transfers data in plaintext, which is insecure. To be secure, one can add a Secure Socket Layer between the FTP application layer and the TCP layer. In this case FTP is called SSL-FTP. We also explore some secure file transfer applications when we discuss SSH later in the chapter.

ELECTRONIC MAIL

Electronic mail (or e-mail) allows users to exchange messages. The nature of this application is different from other applications discussed so far. This means that the idea of client/server programming should be implemented in another way: using some intermediate computers (servers).



26.3.1 Architecture

To explain the architecture of e-mail, we give a common scenario, as shown in Figure 26.12. Another possibility is the case in which Alice or Bob is directly connected to the corresponding mail server, in which LAN or WAN connection is not required, but this variation in the scenario does not affect our discussion.

Figure 26.12: Common scenario

UA: user agent
MTA: message transfer agent
MAA: message access agent

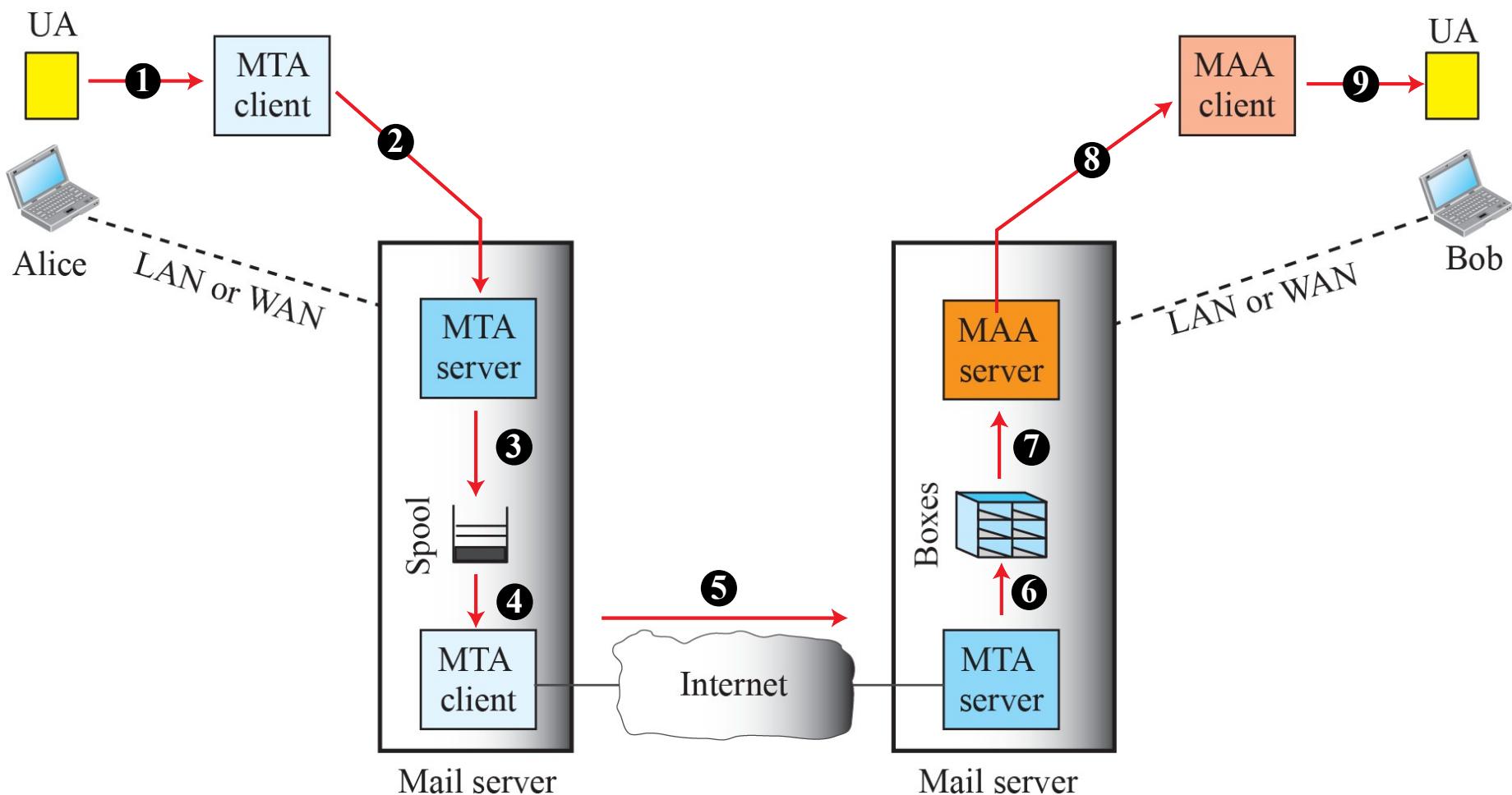
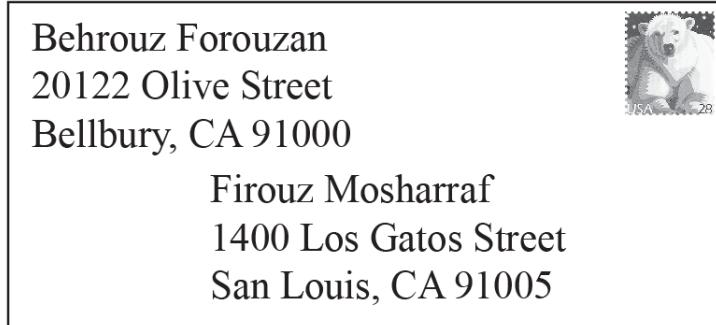


Figure 26.13: Format of an e-mail



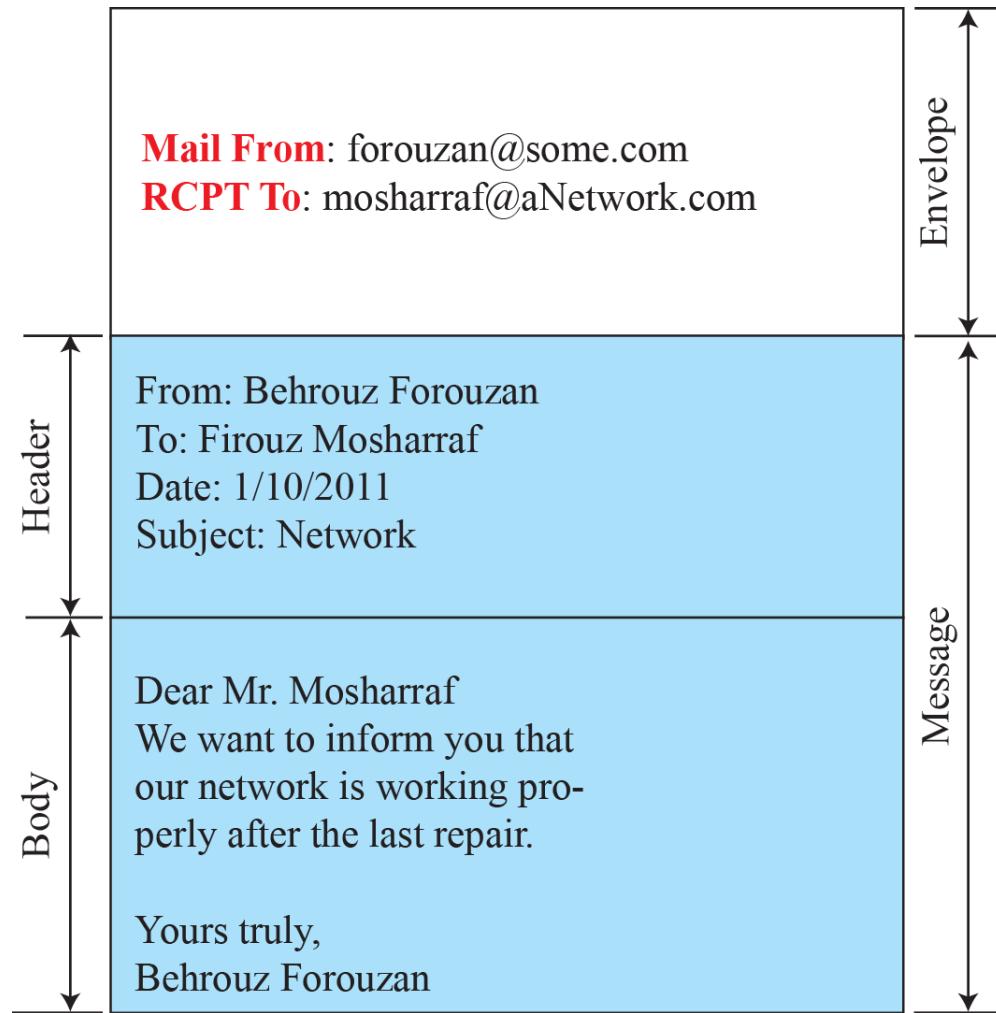
Behrouz Forouzan
20122 Olive Street
Bellbury, CA 91000
Jan. 10, 2011

Subject: Network

Dear Mr. Mosharraf
We want to inform you that
our network is working pro-
perly after the last repair.

Yours truly,
Behrouz Forouzan

Postal mail



Electronic mail

Figure 26.14: E-mail address

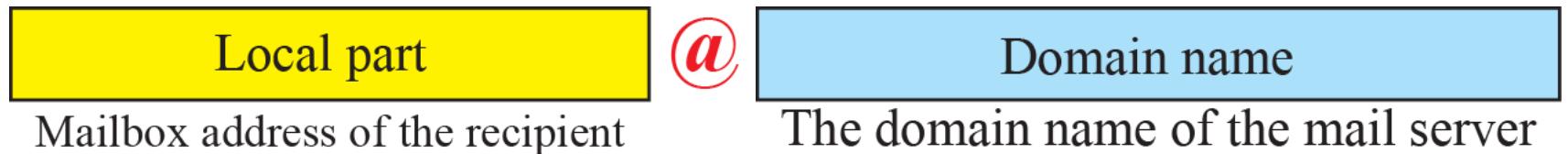
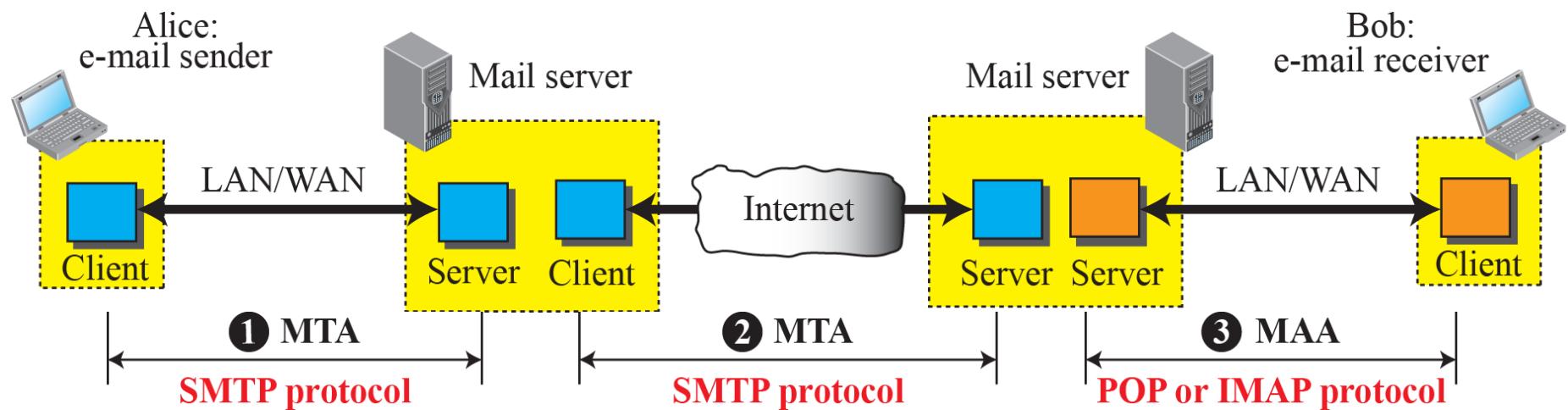


Figure 26.15: Protocols used in electronic mail



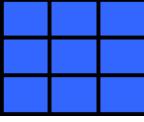


Table 26.6: SMTP Commands

Keyword	Argument(s)	Description
HELO	Sender's host name	Identifies itself
MAIL FROM	Sender of the message	Identifies the sender of the message
RCPT TO	Intended recipient	Identifies the recipient of the message
DATA	Body of the mail	Sends the actual message
QUIT		Terminates the message
RSET		Aborts the current mail transaction
VRFY	Name of recipient	Verifies the address of the recipient
NOOP		Checks the status of the recipient
TURN		Switches the sender and the recipient
EXPN	Mailing list	Asks the recipient to expand the mailing list.
HELP	Command name	Asks the recipient to send information about the command sent as the argument
SEND FROM	Intended recipient	Specifies that the mail be delivered only to the terminal of the recipient, and not to the mailbox
SMOL FROM	Intended recipient	Specifies that the mail be delivered to the terminal <i>or</i> the mailbox of the recipient
SMAL FROM	Intended recipient	Specifies that the mail be delivered to the terminal <i>and</i> the mailbox of the recipient



Table 26.7: SMTP responses (Continued)

<i>Code</i>	<i>Description</i>
Positive Completion Reply	
211	System status or help reply
214	Help message
220	Service ready
221	Service closing transmission channel
250	Request command completed
251	User not local; the message will be forwarded
Positive Intermediate Reply	
354	Start mail input
Transient Negative Completion Reply	
421	Service not available
450	Mailbox not available
451	Command aborted: local error
452	Command aborted; insufficient storage

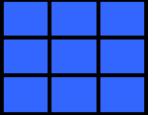


Table 26.7: SMTP responses (continued)

Permanent Negative Completion Reply	
500	Syntax error; unrecognized command
501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command temporarily not implemented
550	Command is not executed; mailbox unavailable
551	User not local
552	Requested action aborted; exceeded storage location
553	Requested action not taken; mailbox name not allowed
554	Transaction failed

Example 26.12

To show the three mail transfer phases, we show all of the steps described above using the information depicted in Figure 26.16. In the figure, we have separated the messages related to the envelope, header, and body in the data transfer section. Note that the steps in this figure are repeated two times in each e-mail transfer: once from the e-mail sender to the local mail server and once from the local mail server to the remote mail server. The local mail server, after receiving the whole e-mail message, may spool it and send it to the remote mail server at another time.

Figure 26.16: Example 26.12

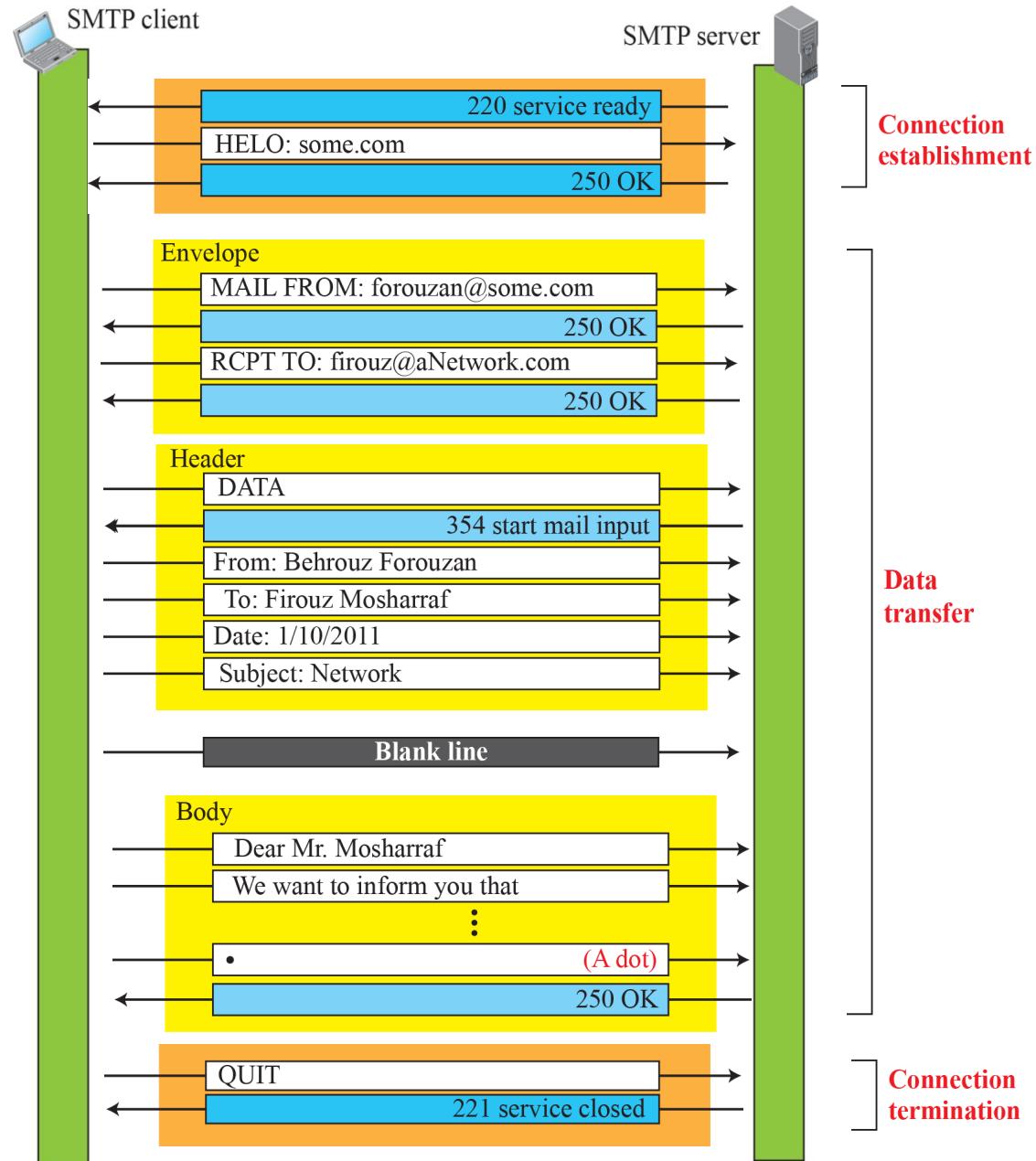


Figure 26.17: POP3

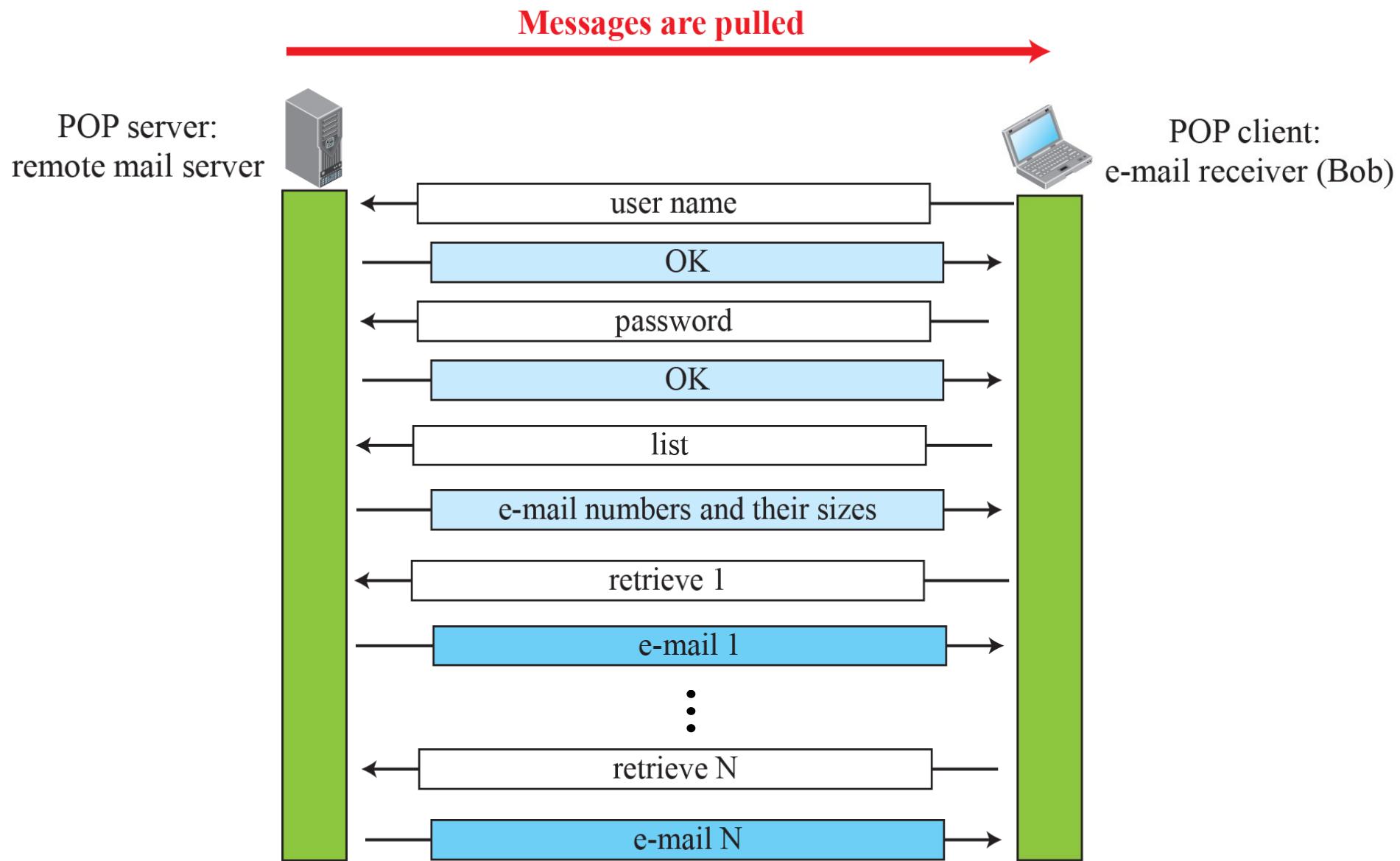


Figure 26.18: MIME

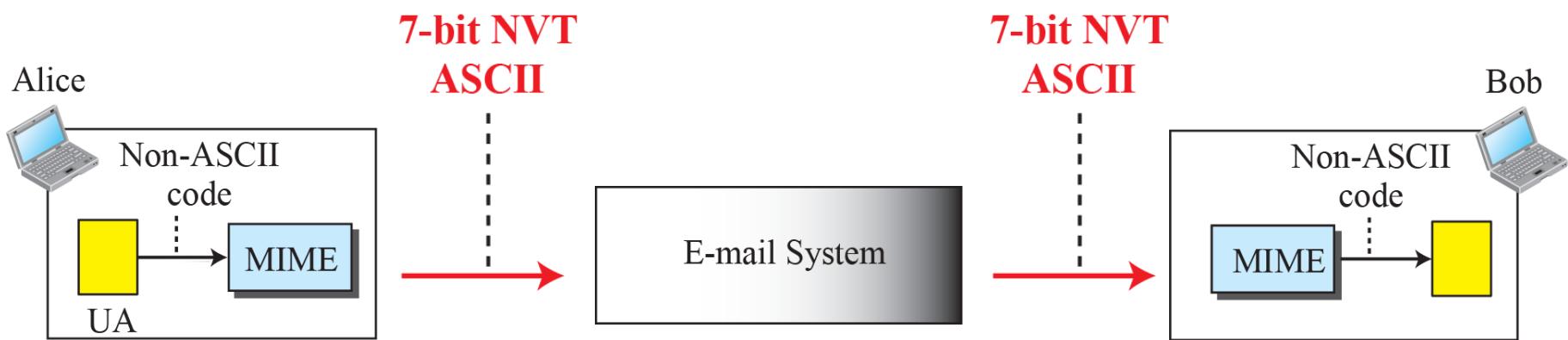


Figure 26.19: MIME header

MIME headers

E-mail header
MIME-Version: 1.1 Content-Type: type/subtype Content-Transfer-Encoding: encoding type Content-ID: message ID Content-Description: textual explanation of nontextual contents
E-mail body



Table 26.8: Data Types and Subtypes in MIME

Type	Subtype	Description
Text	Plain	Unformatted
	HTML	HTML format (see Appendix C)
Multipart	Mixed	Body contains ordered parts of different data types
	Parallel	Same as above, but no order
	Digest	Similar to Mixed, but the default is message/RFC822
	Alternative	Parts are different versions of the same message
Message	RFC822	Body is an encapsulated message
	Partial	Body is a fragment of a bigger message
	External-Body	Body is a reference to another message
Image	JPEG	Image is in JPEG format
	GIF	Image is in GIF format
Video	MPEG	Video is in MPEG format
Audio	Basic	Single channel encoding of voice at 8 KHz
Application	PostScript	Adobe PostScript
	Octet-stream	General binary data (eight-bit bytes)

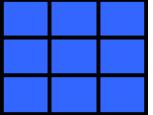


Table 2.9: Methods for Content-Transfer-Encoding

Type	Description
7-bit	NVT ASCII characters with each line less than 1000 characters
8-bit	Non-ASCII characters with each line less than 1000 characters
Binary	Non-ASCII characters with unlimited-length lines
Base64	6-bit blocks of data encoded into 8-bit ASCII characters
Quoted-printable	Non-ASCII characters encoded as an equal sign plus an ASCII code

Figure 26.20: Base64 conversion

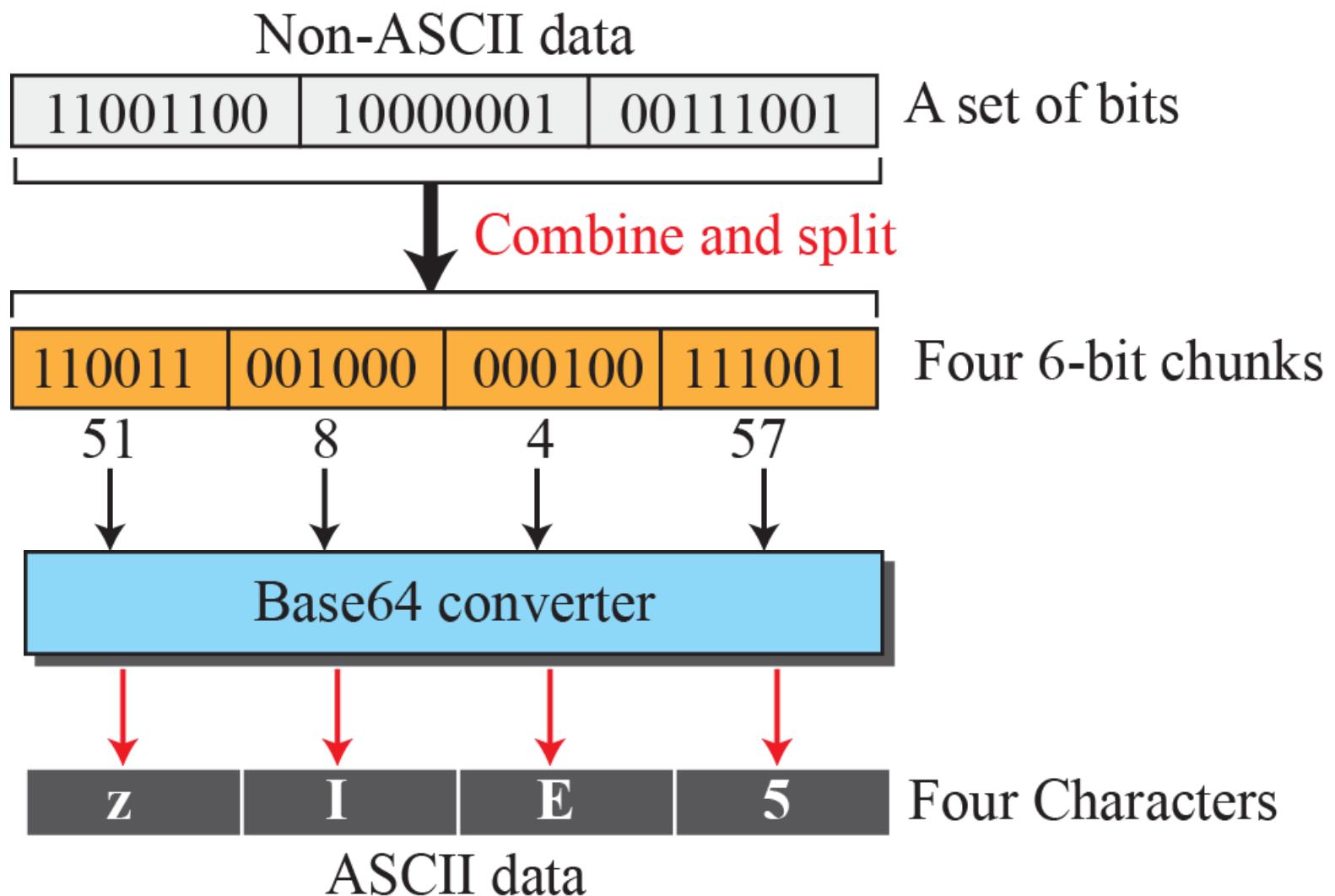
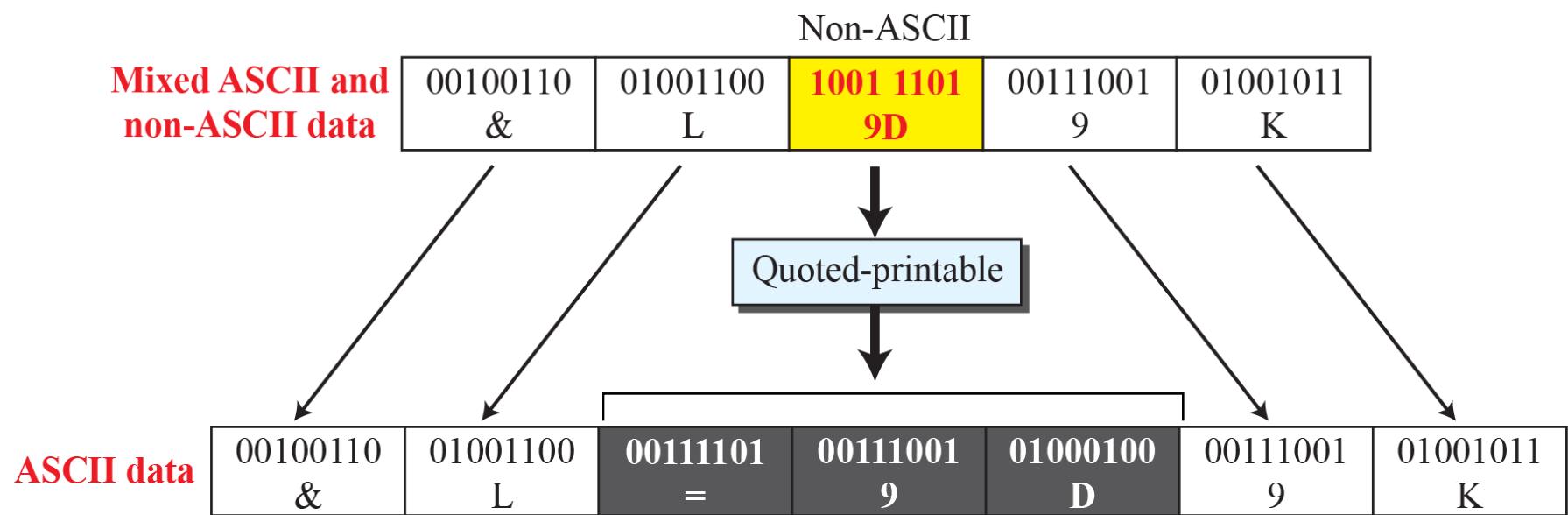


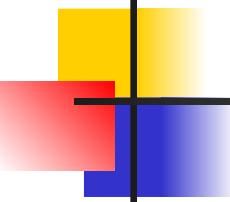


Table 26.10: Base64 Converting Table

Value	Code												
0	A	11	L	22	W	33	h	44	s	55	3		
1	B	12	M	23	X	34	i	45	t	56	4		
2	C	13	N	24	Y	35	j	46	u	57	5		
3	D	14	O	25	Z	36	k	47	v	58	6		
4	E	15	P	26	a	37	l	48	w	59	7		
5	F	16	Q	27	b	38	m	49	x	60	8		
6	G	17	R	28	c	39	n	50	y	61	9		
7	H	18	S	29	d	40	o	51	z	62	+		
8	I	19	T	30	e	41	p	52	0	63	/		
9	J	20	U	31	f	42	q	53	1				
10	K	21	V	32	g	43	r	54	2				

Figure 26.21: Quoted-printable

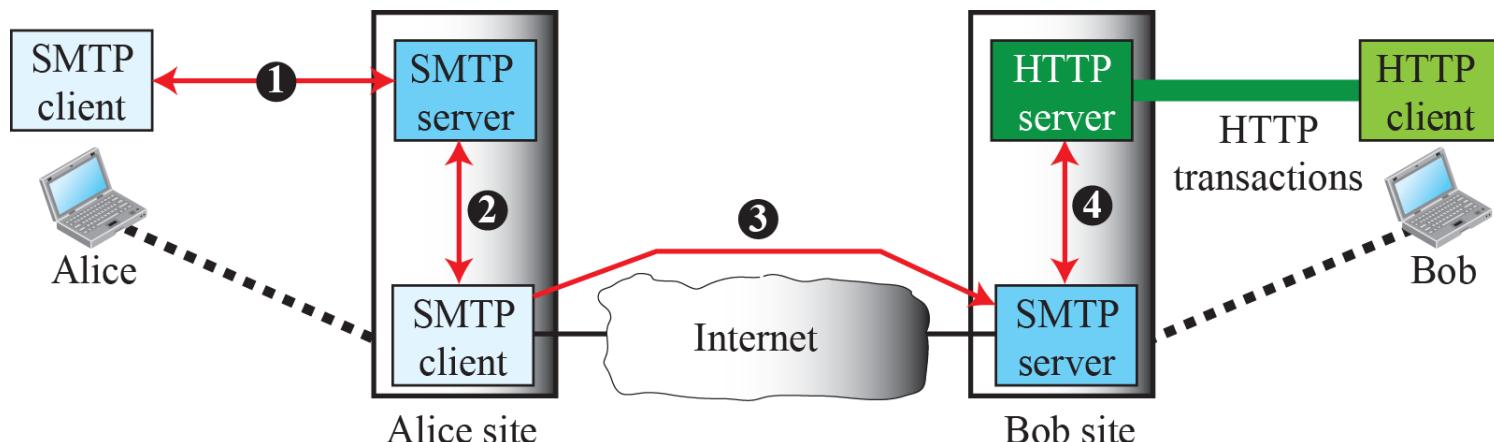




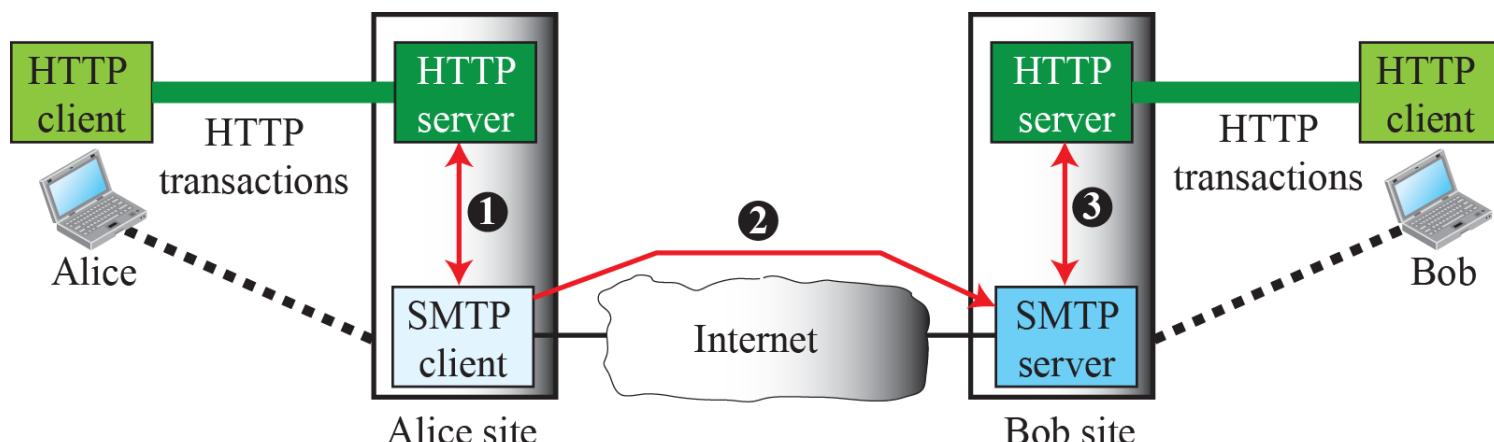
26.3.2 E-Mail Security

The protocol discussed in this chapter does not provide any security provisions per se. However, e-mail exchanges can be secured using two application-layer securities designed in particular for e-mail systems. Two of these protocols, Pretty Good Privacy (PGP) and Secure/Multipurpose Internet Mail Extensions (S/MIME), are discussed in Chapter 32 after we have discussed basic network security.

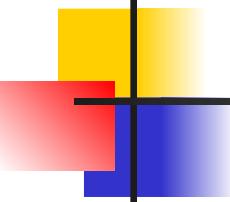
Figure 26.22: Web-based e-mail, cases I and II



Case 1: Only receiver uses HTTP



Case 2: Both sender and receiver use HTTP

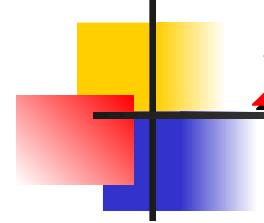


26.3.3 Iter. Programming Using TCP

We are now ready to discuss network programming using the service of TCP, a connection-oriented service.

26-4 TELNET

It is impossible to have a client/server pair for each type of service we need; the number of servers soon becomes intractable. The idea is not scalable. The solution is to have a specific client/server program for a set of common scenarios, but to have some generic client/server programs for the rest.



26.4.1 Local versus Remote Logging

We first discuss the concept of local and remote logging as shown in Figure 26.23.

Figure 2.23: Local versus remote logging

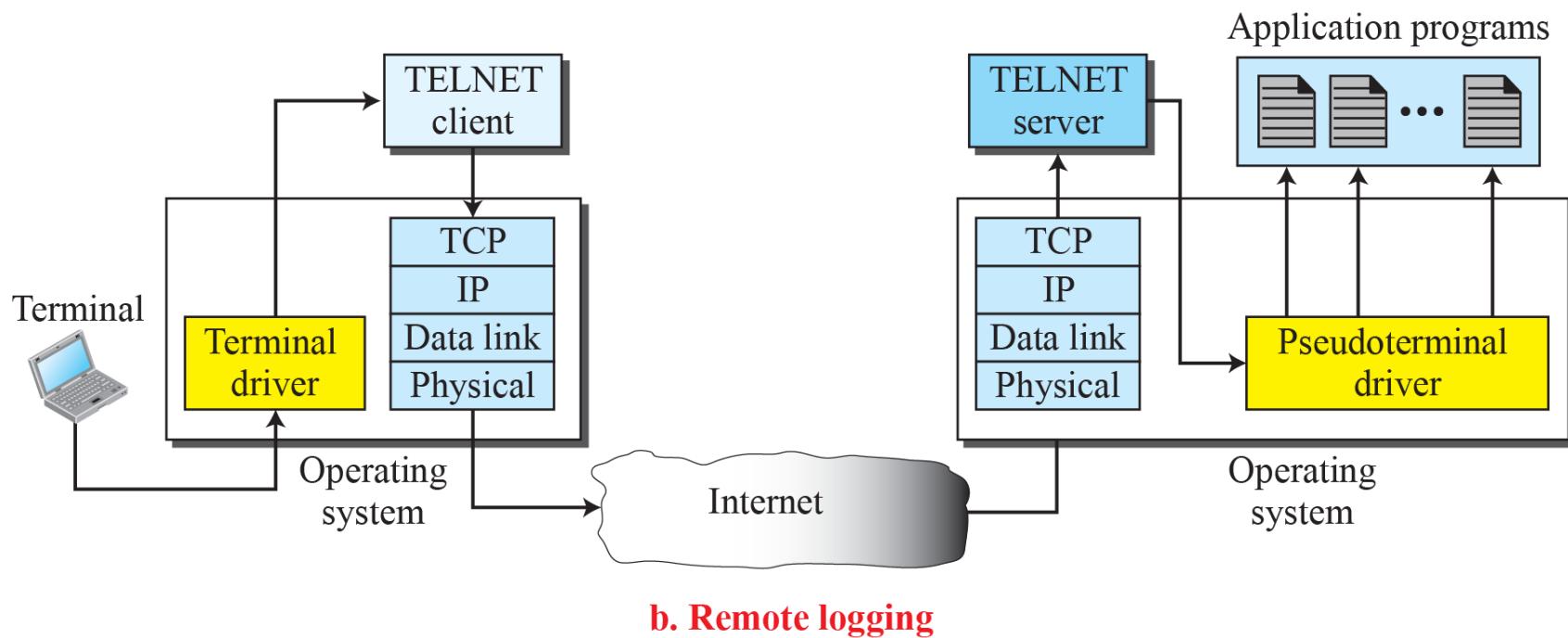
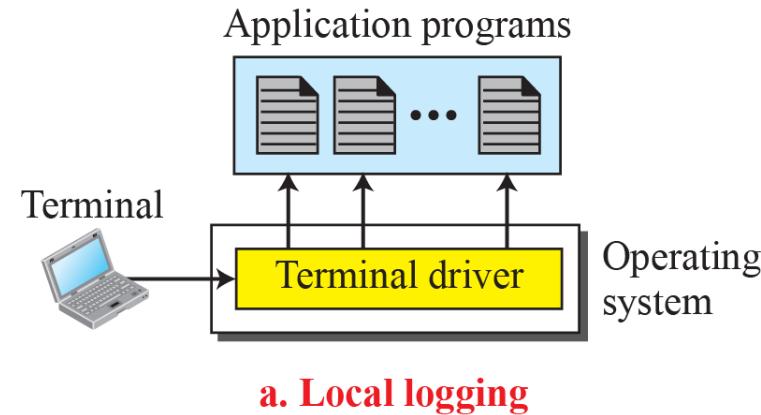
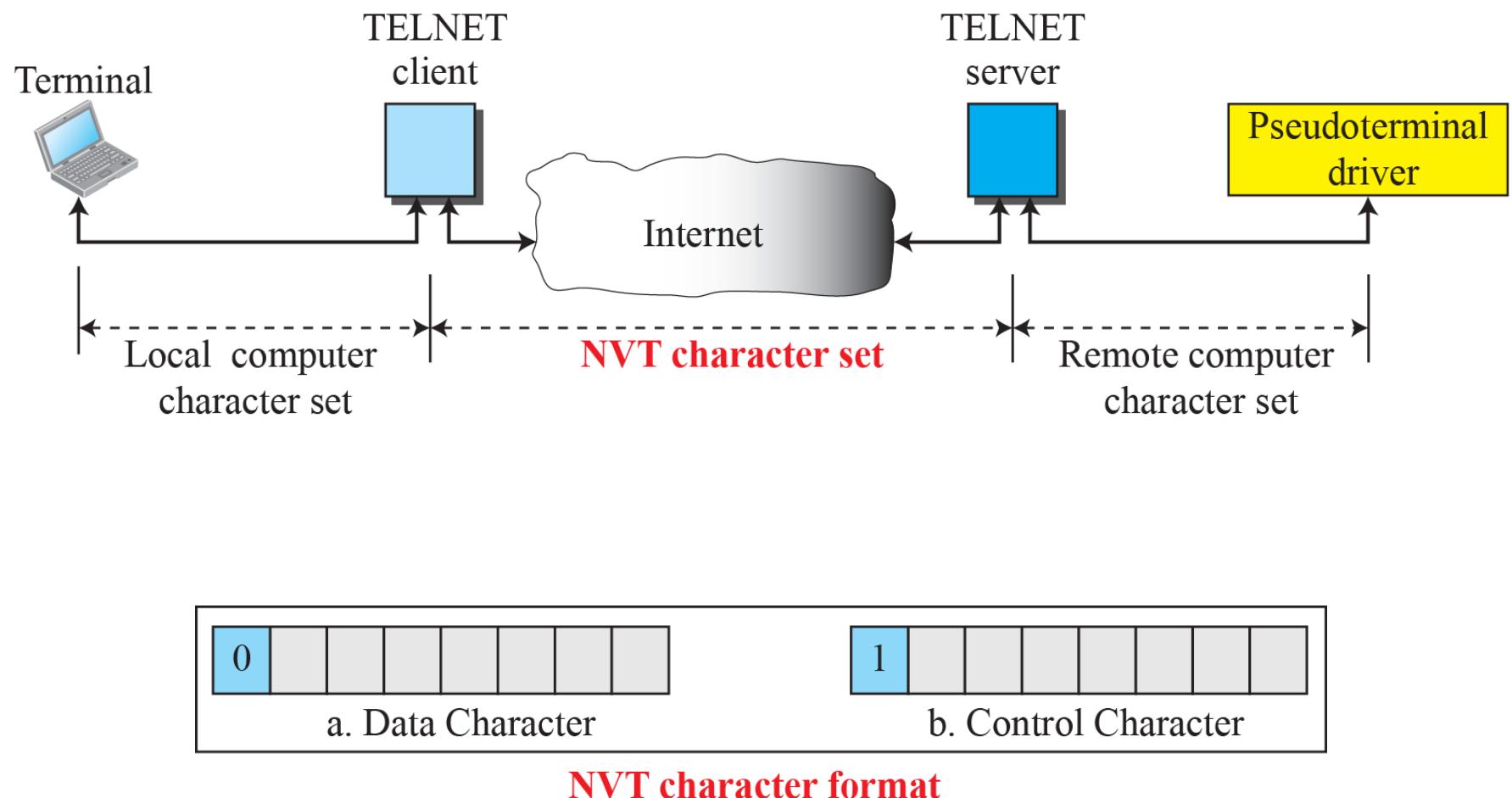


Figure 26.24: Concept of NVT



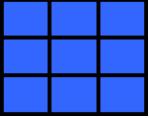
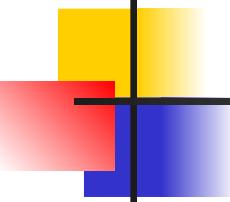


Table 26.11: Examples of interface commands

<i>Command</i>	<i>Meaning</i>	<i>Command</i>	<i>Meaning</i>
open	Connect to a remote computer	set	Set the operating parameters
close	Close the connection	status	Display the status information
display	Show the operating parameters	send	Send special characters
mode	Change to line or character mode	quit	Exit TELNET

SECURE SHELL (SSH)

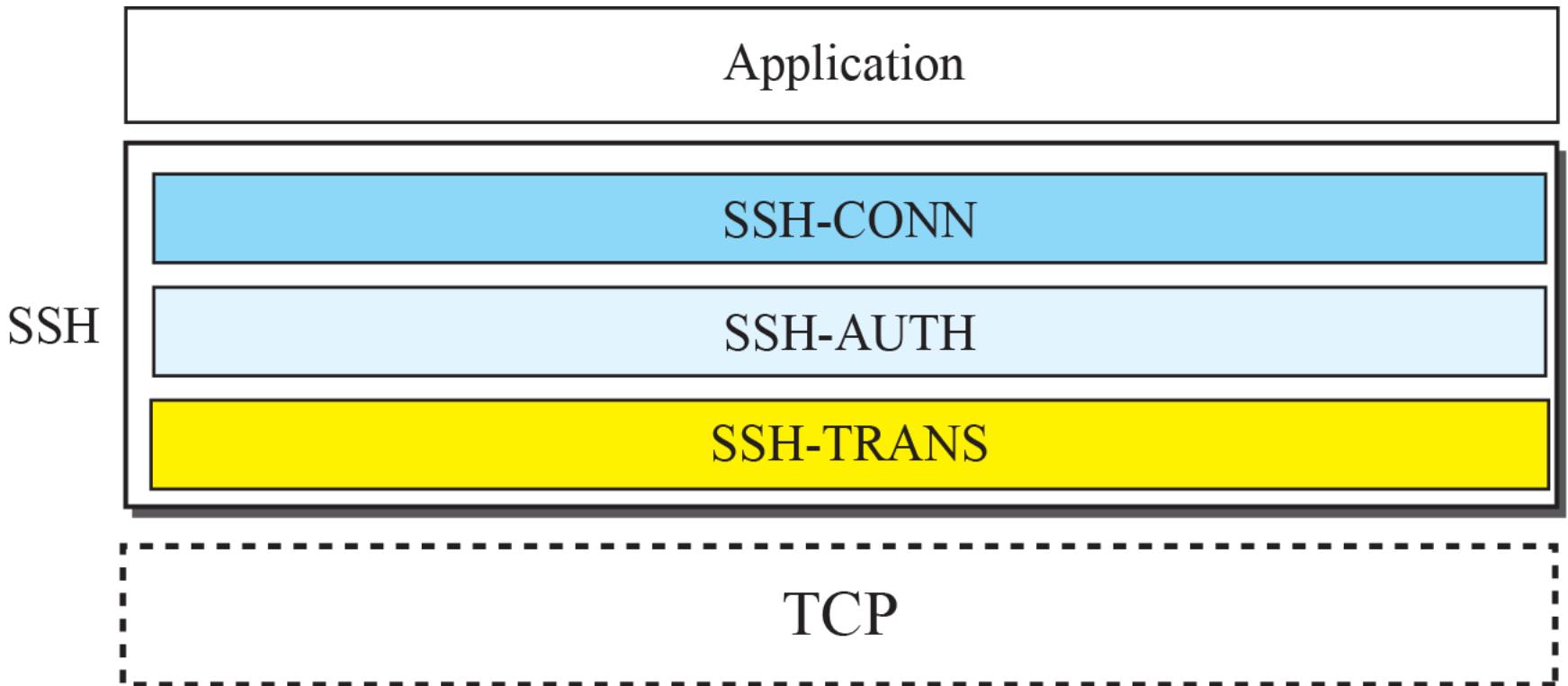
Although Secure Shell (SSH) is a secure application program that can be used today for several purposes such as remote logging and file transfer, it was originally designed to replace TELNET. There are two versions of SSH. The first version, SSH-1, is now deprecated because of security flaws in it. In this section, we discuss only SSH-2.

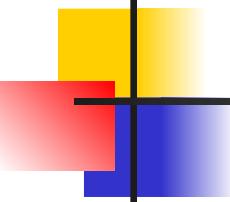


26.5.1 Components

SSH is an application-layer protocol with three components, as shown in Figure 26.25.

Figure 26.25: Components of SSH





26.5.2 Applications

Although SSH is often thought of as a replacement for TELNET, SSH is, in fact, a general-purpose protocol that provides a secure connection between a client and server.

Figure 26.26: Port Forwarding

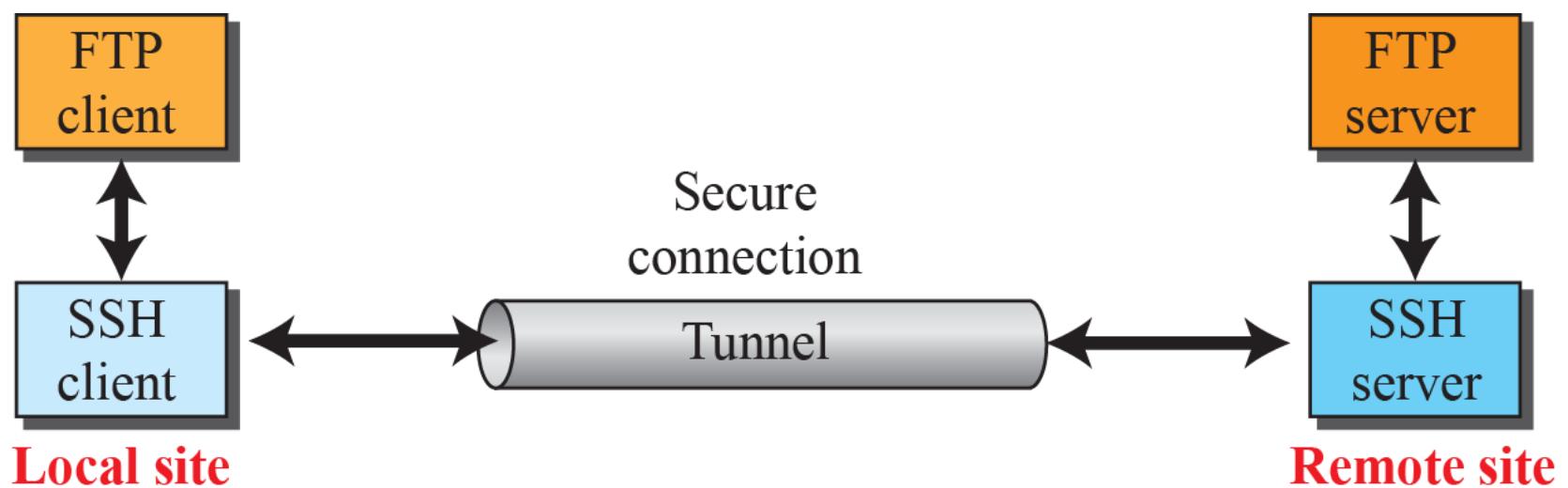
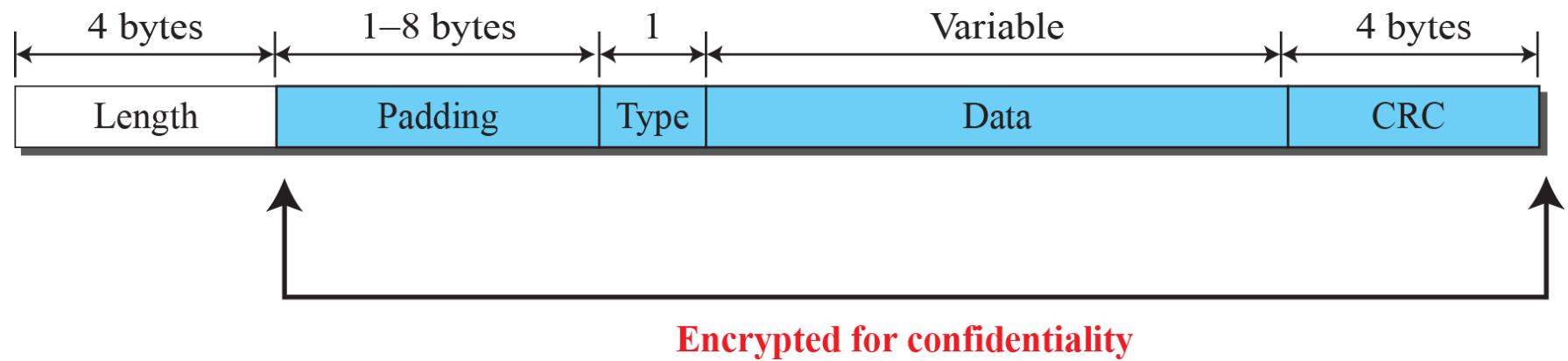


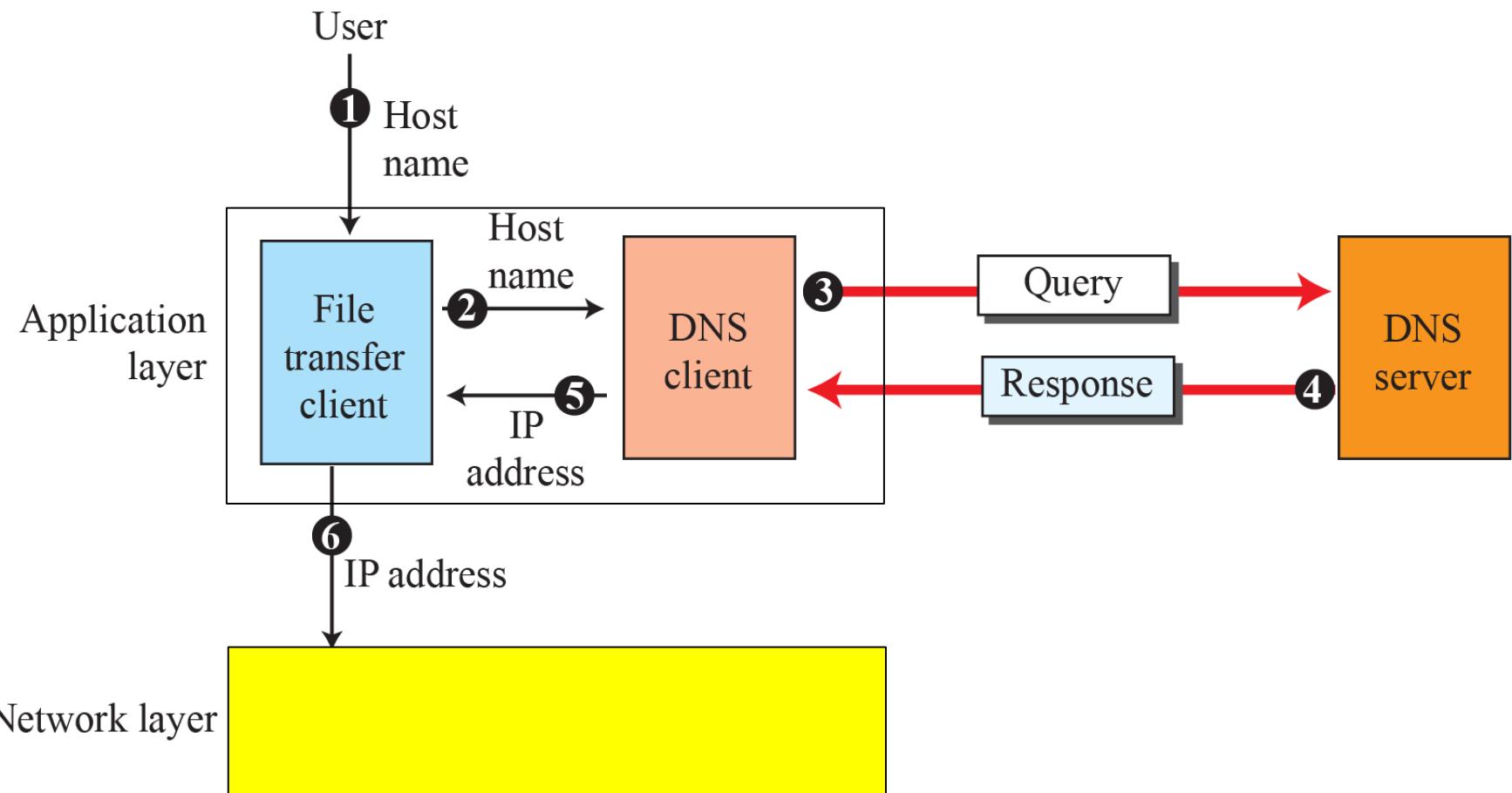
Figure 26.27: SSH Packet Format

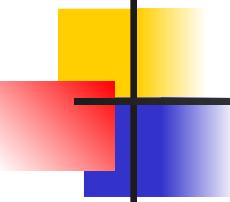


DOMAIN NAME SYSTEM (DNS)

The last client-server application program we discuss has been designed to help other application programs. The Internet needs to have a directory system that can map a name to an address. This is analogous to the telephone network. Figure 26.28 shows how TCP/IP uses a DNS client and a DNS server to map a name to an address.

Figure 26.28: Purpose of DNS





26.6.1 Name Space

To be unambiguous, the names assigned to machines must be carefully selected from a name space with complete control over the binding between the names and IP addresses. In other words, the names must be unique because the addresses are unique. A name space that maps each address to a unique name can be organized in two ways: flat or hierarchical.

Figure 26.29: Domain name space

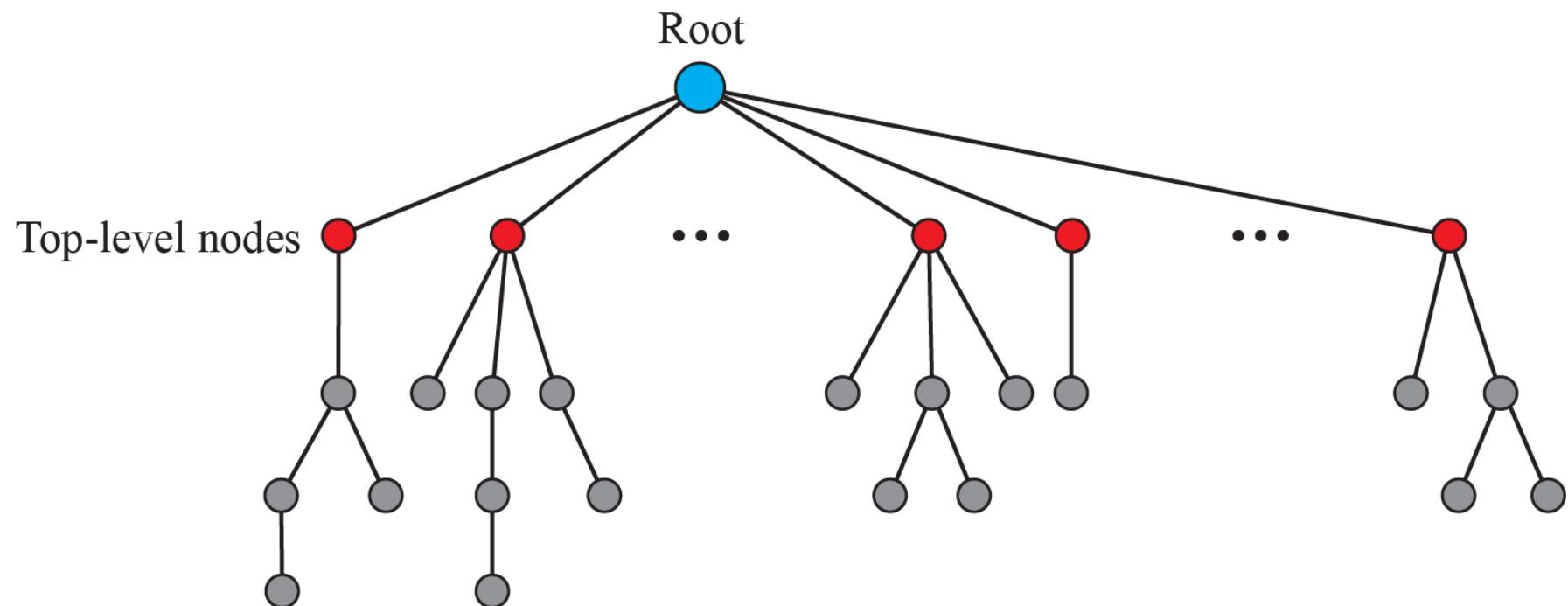


Figure 26.30: Domain names and labels

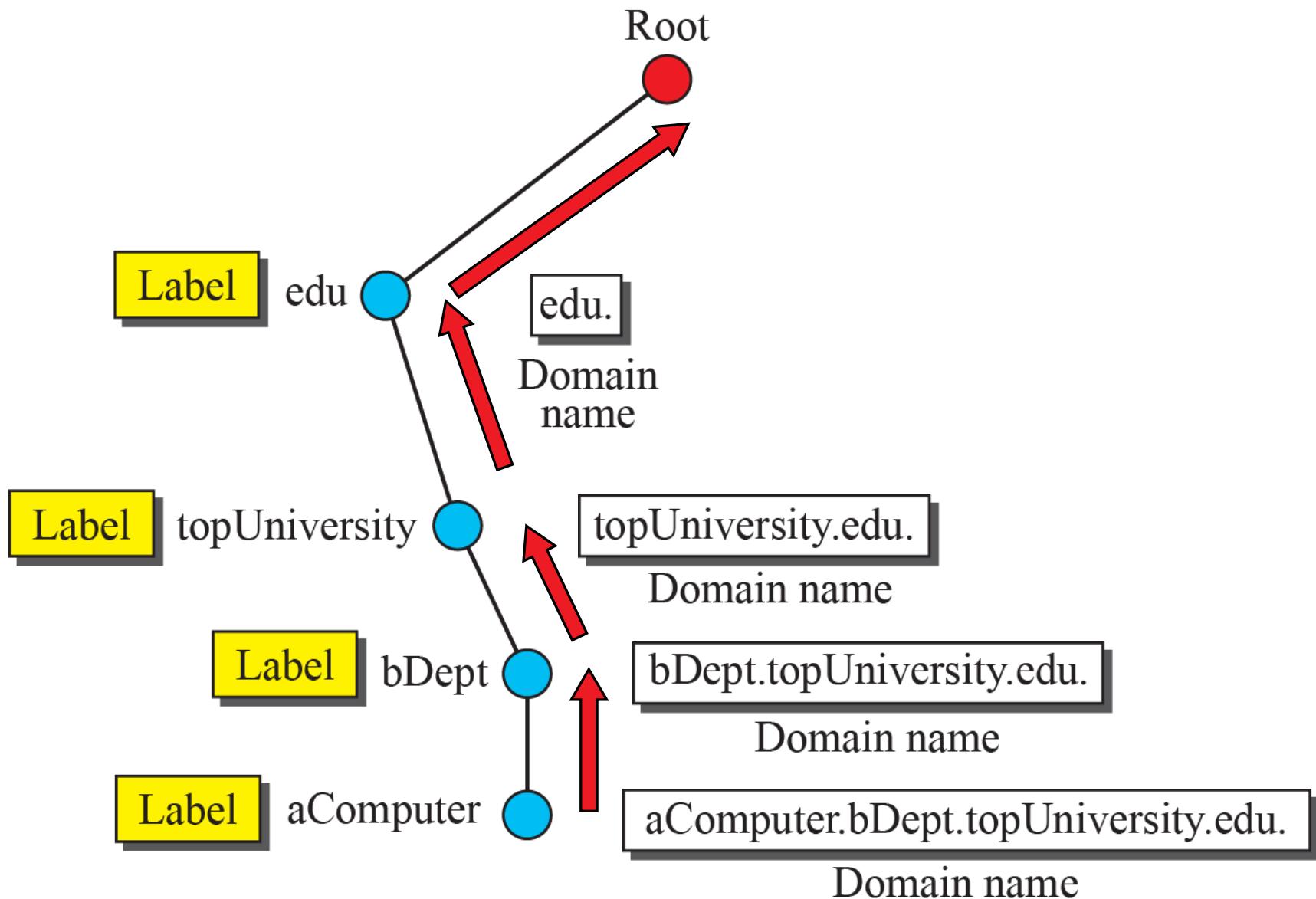


Figure 26.31: Domains

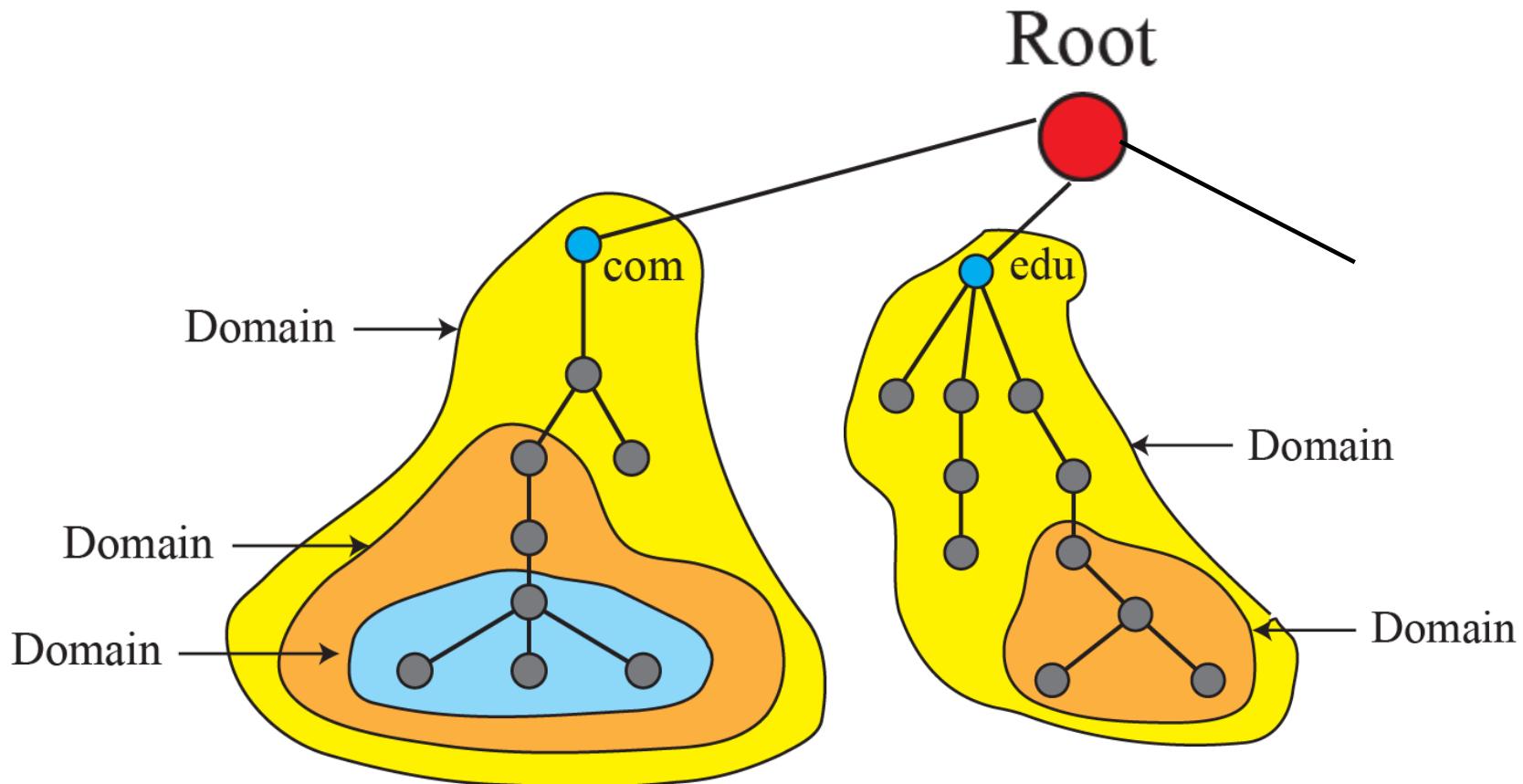


Figure 26.32: Hierarchy of name servers

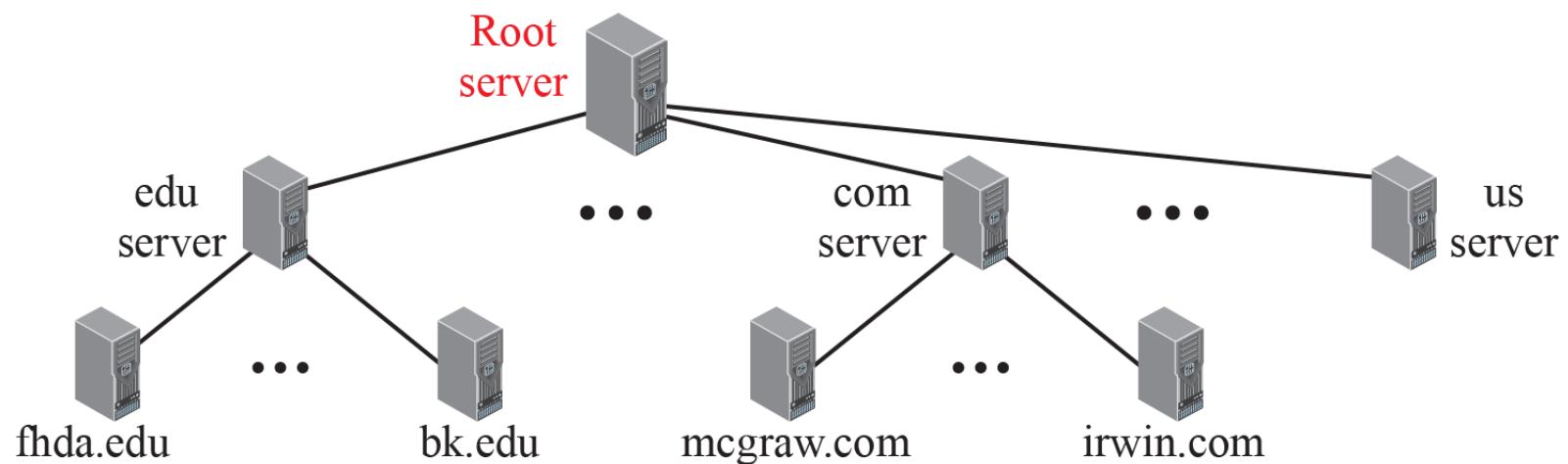
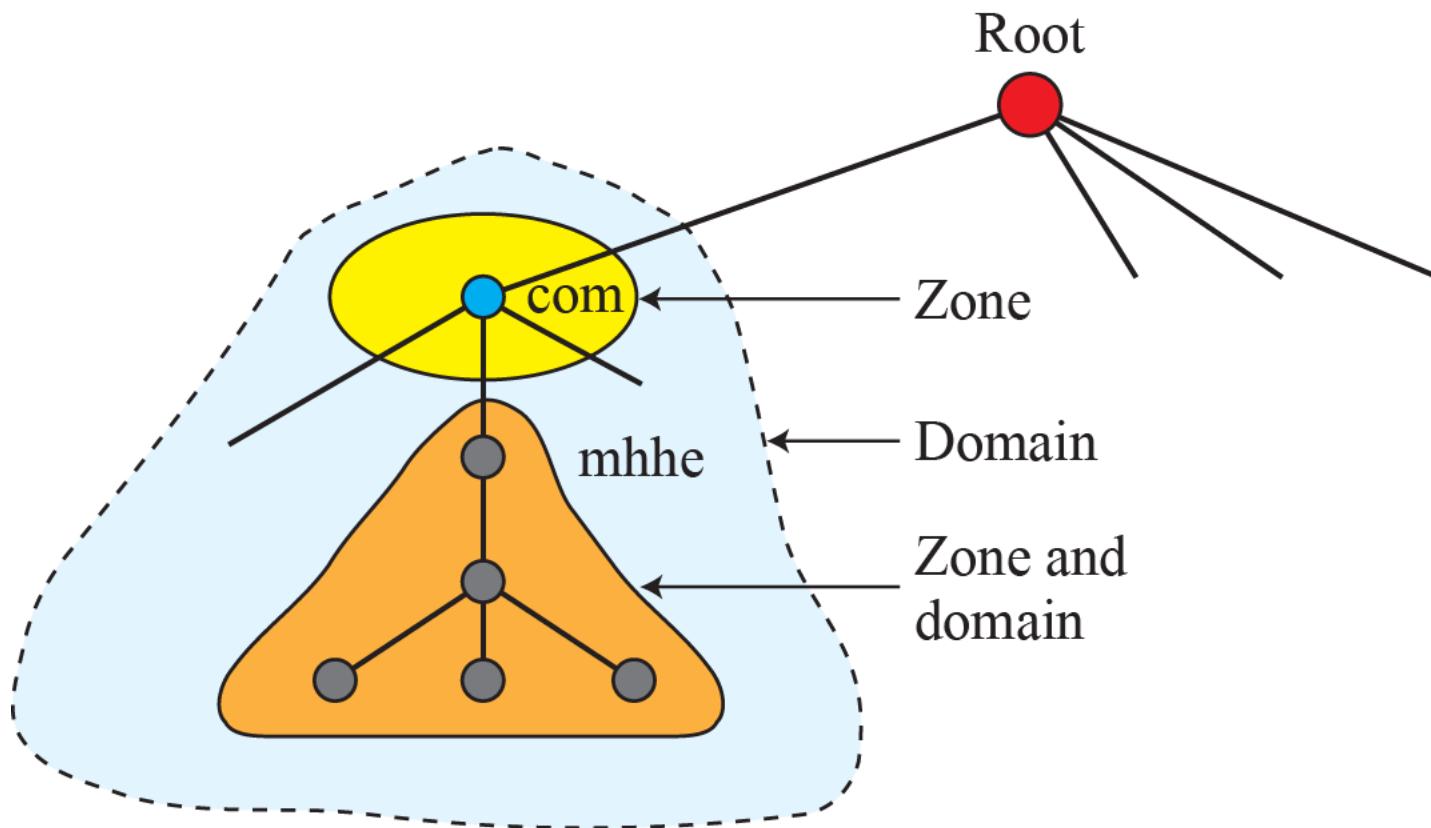
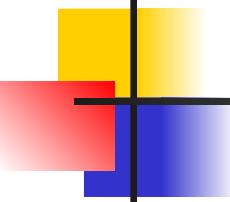


Figure 26.33: Zone





26.6.2 DNS in the Internet

DNS is a protocol that can be used in different platforms. In the Internet, the domain name space (tree) was originally divided into three different sections: generic domains, country domains, and the inverse domains. However, due to the rapid growth of the Internet, it became extremely difficult to keep track of the inverse domains, which could be used to find the name of a host when given the IP address. The inverse domains are now deprecated (see RFC 3425). We, therefore, concentrate on the first two.

Figure 26.34: Generic domains

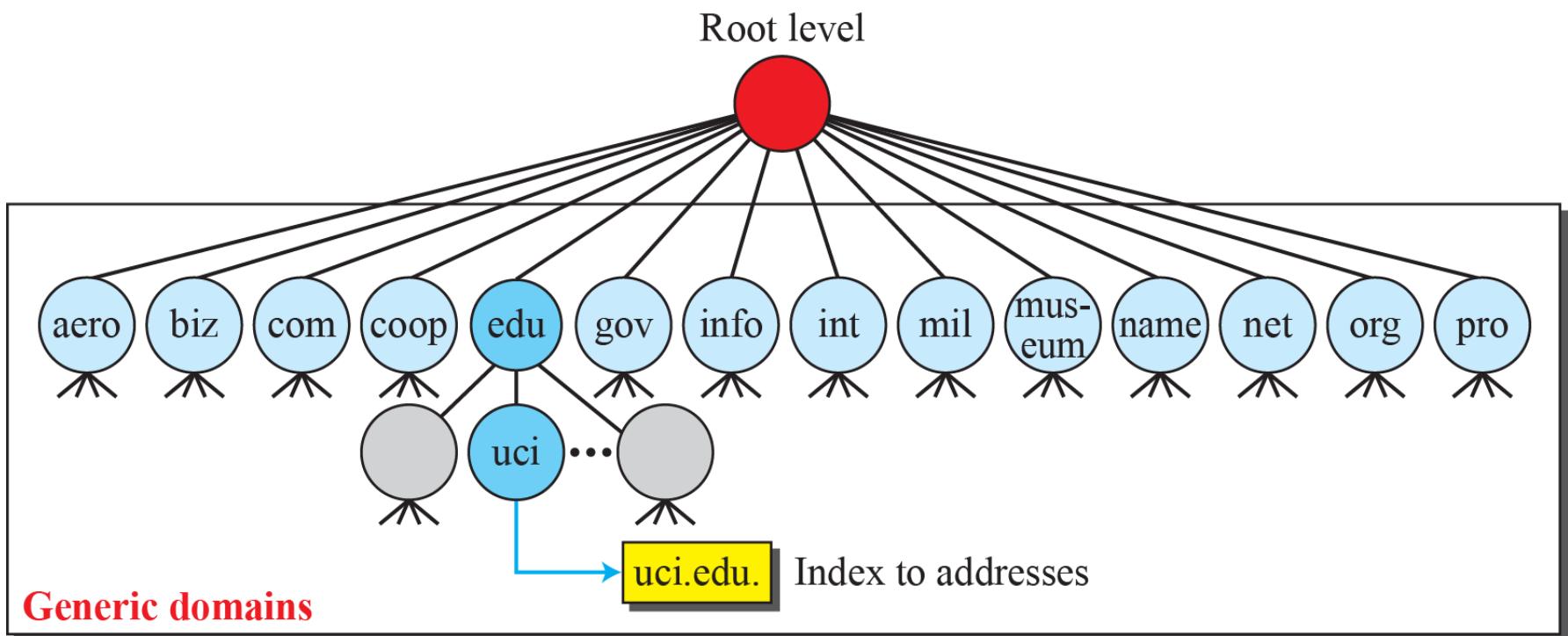
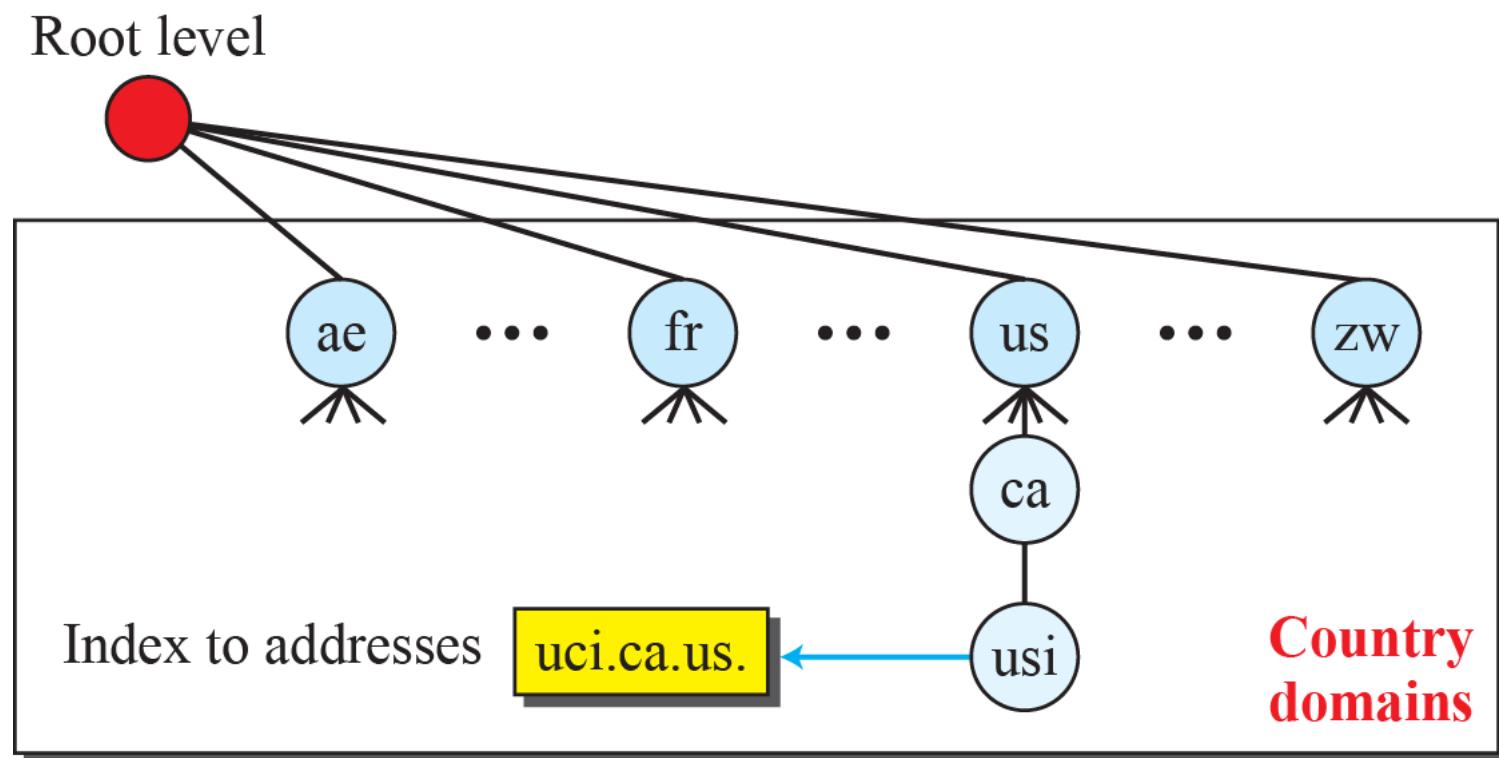


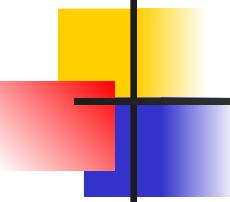


Table 26.12: Generic domain labels

<i>Label</i>	<i>Description</i>	<i>Label</i>	<i>Description</i>
aero	Airlines and aerospace	int	International organizations
biz	Businesses or firms	mil	Military groups
com	Commercial organizations	museum	Museums
coop	Cooperative organizations	name	Personal names (individuals)
edu	Educational institutions	net	Network support centers
gov	Government institutions	org	Nonprofit organizations
info	Information service providers	pro	Professional organizations

Figure 26.35: Country domains

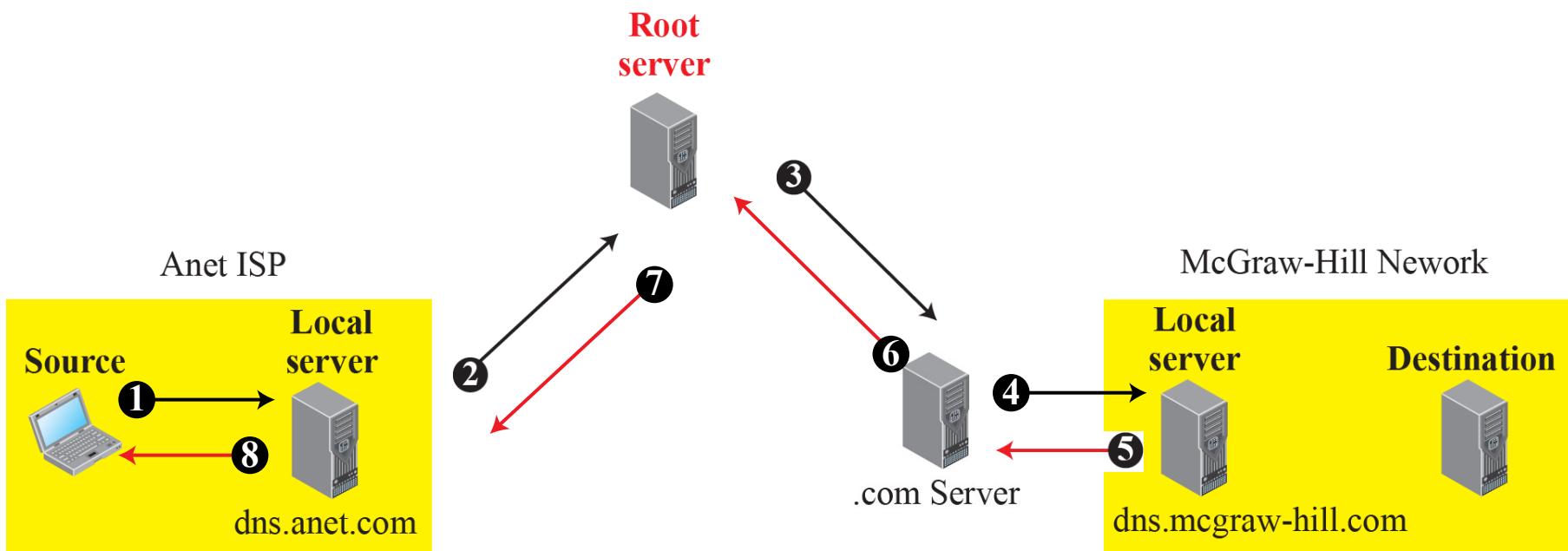




26.6.3 Resolution

Mapping a name to an address is called name-address resolution. DNS is designed as a client-server application. A host that needs to map an address to a name or a name to an address calls a DNS client called a resolver. The resolver accesses the closest DNS server with a mapping request. If the server has the information, it satisfies the resolver; otherwise, it either refers the resolver to other servers or asks other servers to provide the information.

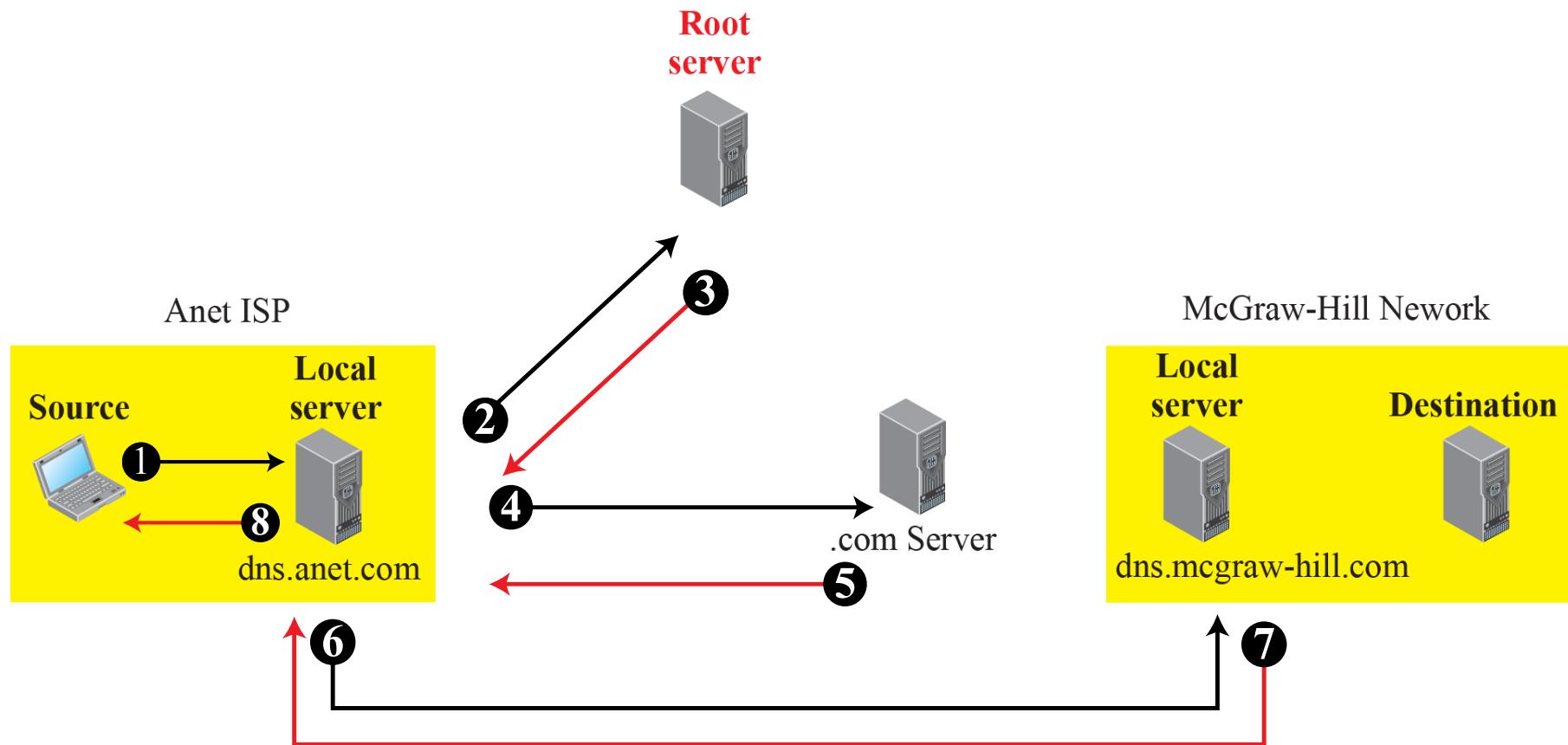
Figure 26.36: Recursive resolution



Source: some.anet.com

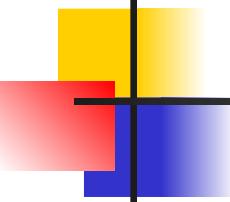
Destination: engineering.mcgraw-hill.com

Figure 26.37: Iterative resolution



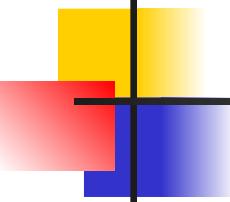
Source: some.anet.com

Destination: engineering.mcgraw-hill.com



26.6.4 Caching

Each time a server receives a query for a name that is not in its domain, it needs to search its database for a server IP address. Reduction of this search time would increase efficiency. DNS handles this with a mechanism called caching.



26.6.5 Resource Records

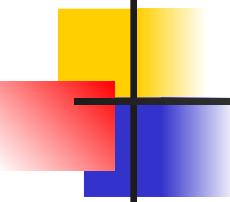
The zone information associated with a server is implemented as a set of resource records. In other words, a name server stores a database of resource records. A resource record is a 5-tuple structure, as shown below:

(Domain Name, Type, Class, TTL, Value)



Table 26.13: DNS types

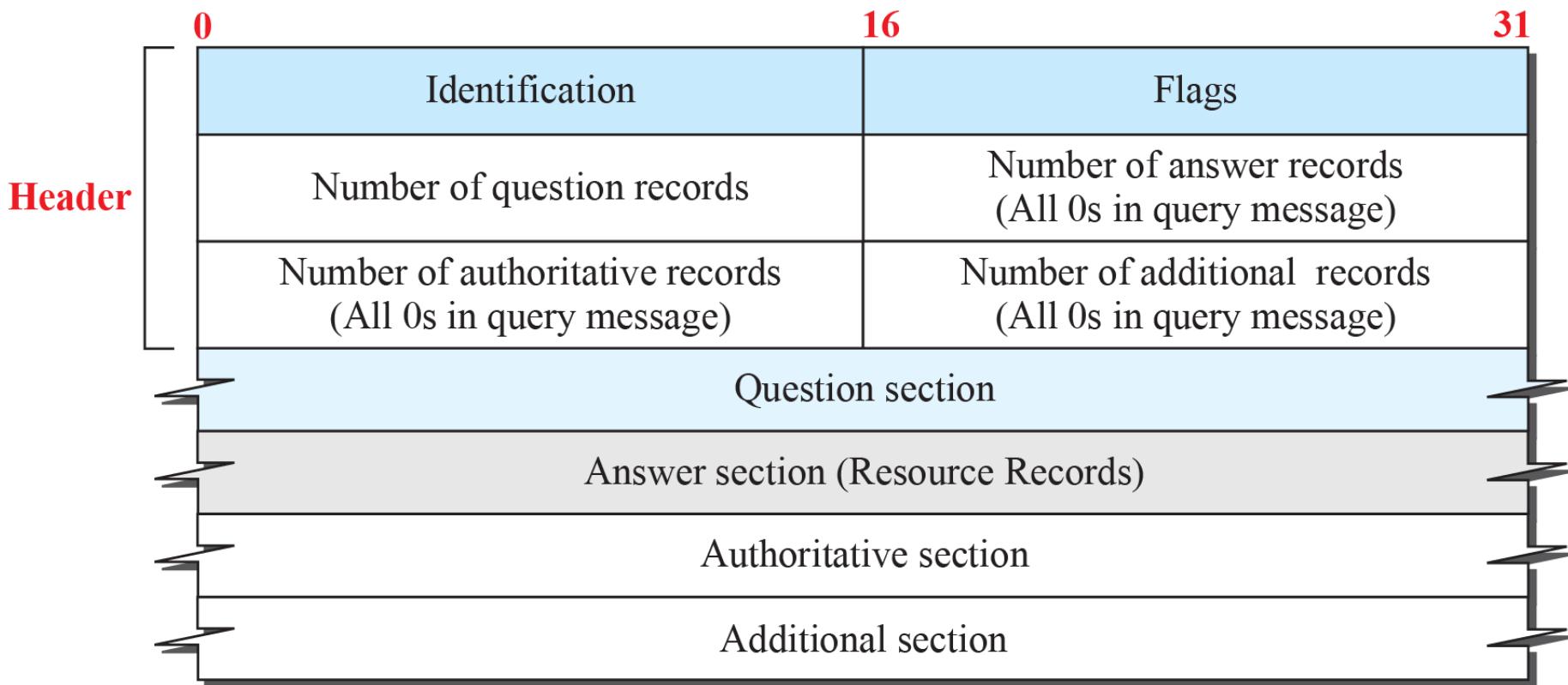
Type	<i>Interpretation of value</i>
A	A 32-bit IPv4 address (see Chapter 4)
NS	Identifies the authoritative servers for a zone
CNAME	Defines an alias for the official name of a host
SOA	Marks the beginning of a zone
MX	Redirects mail to a mail server
AAAA	An IPv6 address (see Chapter 4)



26.6.6 DNS Messages

To retrieve information about hosts, DNS uses two types of messages: query and response. Both types have the same format as shown in Figure 26.38.

Figure 26.38: DNS message



Note:

The query message contains only the question section.
The response message includes the question section,
the answer section, and possibly two other sections.

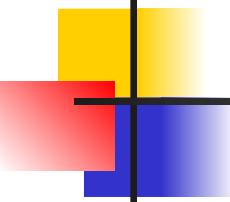
Example 26.13

In UNIX and Windows, the nslookup utility can be used to retrieve address/name mapping. The following shows how we can retrieve an address when the domain name is given.

```
$nslookup www.forouzan.biz
```

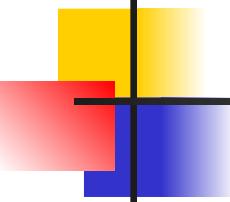
```
Name: www.forouzan.biz
```

```
Address: 198.170.240.179
```



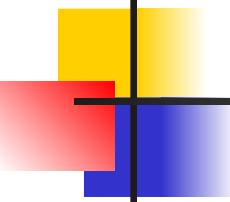
26.6.7 Registrars

How are new domains added to DNS? This is done through a registrar, a commercial entity accredited by ICANN. A registrar first verifies that the requested domain name is unique and then enters it into the DNS database.



26.6.8 DDNS

When the DNS was designed, no one predicted that there would be so many address changes. In DNS, when there is a change, such as adding a new host, removing a host, or changing an IP address, the change must be made to the DNS master file. These types of changes involve a lot of manual updating. The size of today's Internet does not allow for this kind of manual operation.



26.6.9 Security of DNS

DNS is one of the most important systems in the Internet infrastructure; it provides crucial services to Internet users. Applications such as Web access or e-mail are heavily dependent on the proper operation of DNS.