Chapter 5

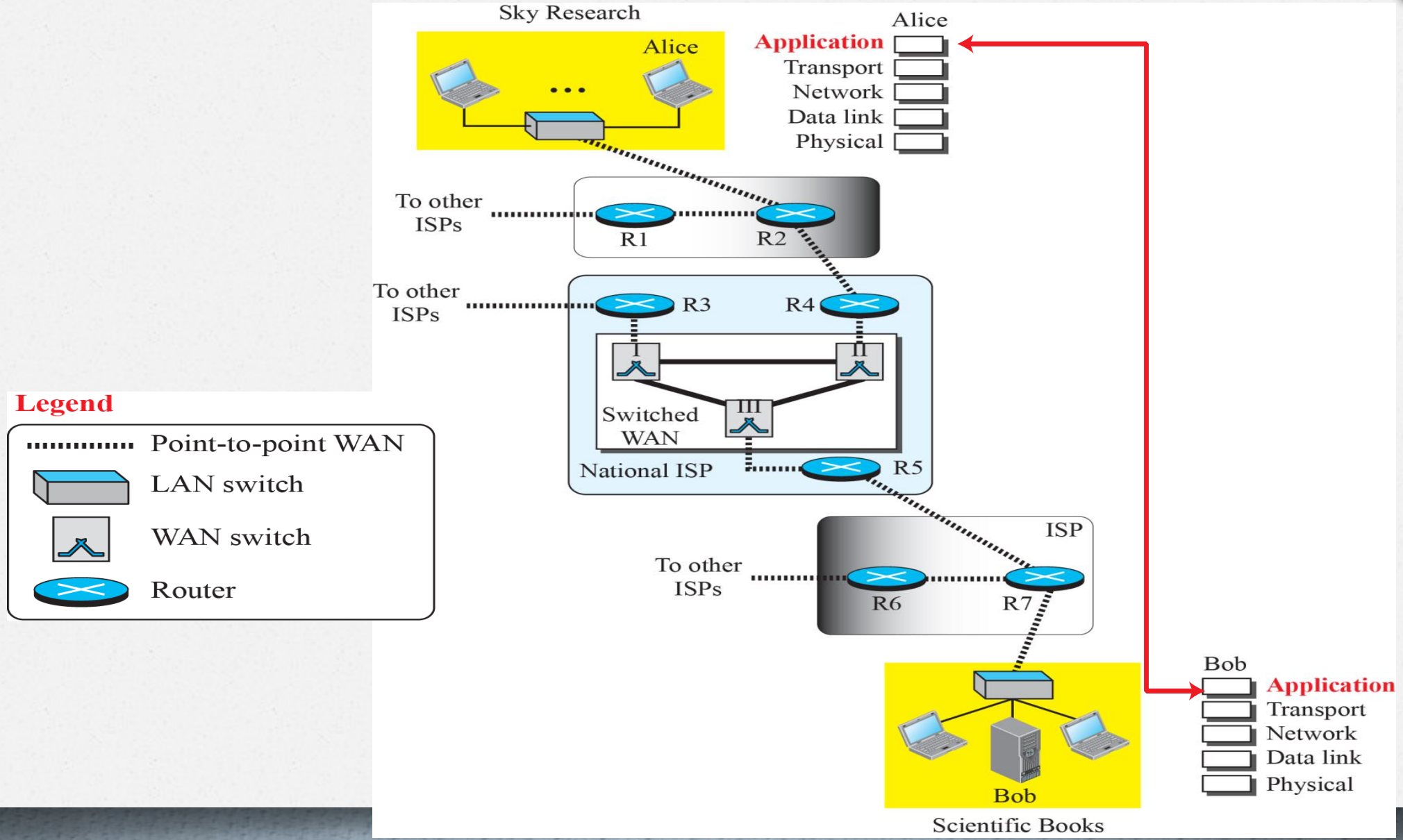
Application Layer

- INTRODUCTION
- CLIENT-SERVER PARADIGM
- STANDARD APPLICATIONS
  - HTTP
  - FTP
  - SMTP
  - Telnet
  - SSH
  - DNS

INTRODUCTION

- The application layer provides services to the user.
- Communication is provided using a logical connection, which means that the two application layers assume that there is an imaginary direct connection through which they can send and receive messages.

Logical connection at the application layer



Providing Services

The Internet was originally designed to provide service to users around the world.

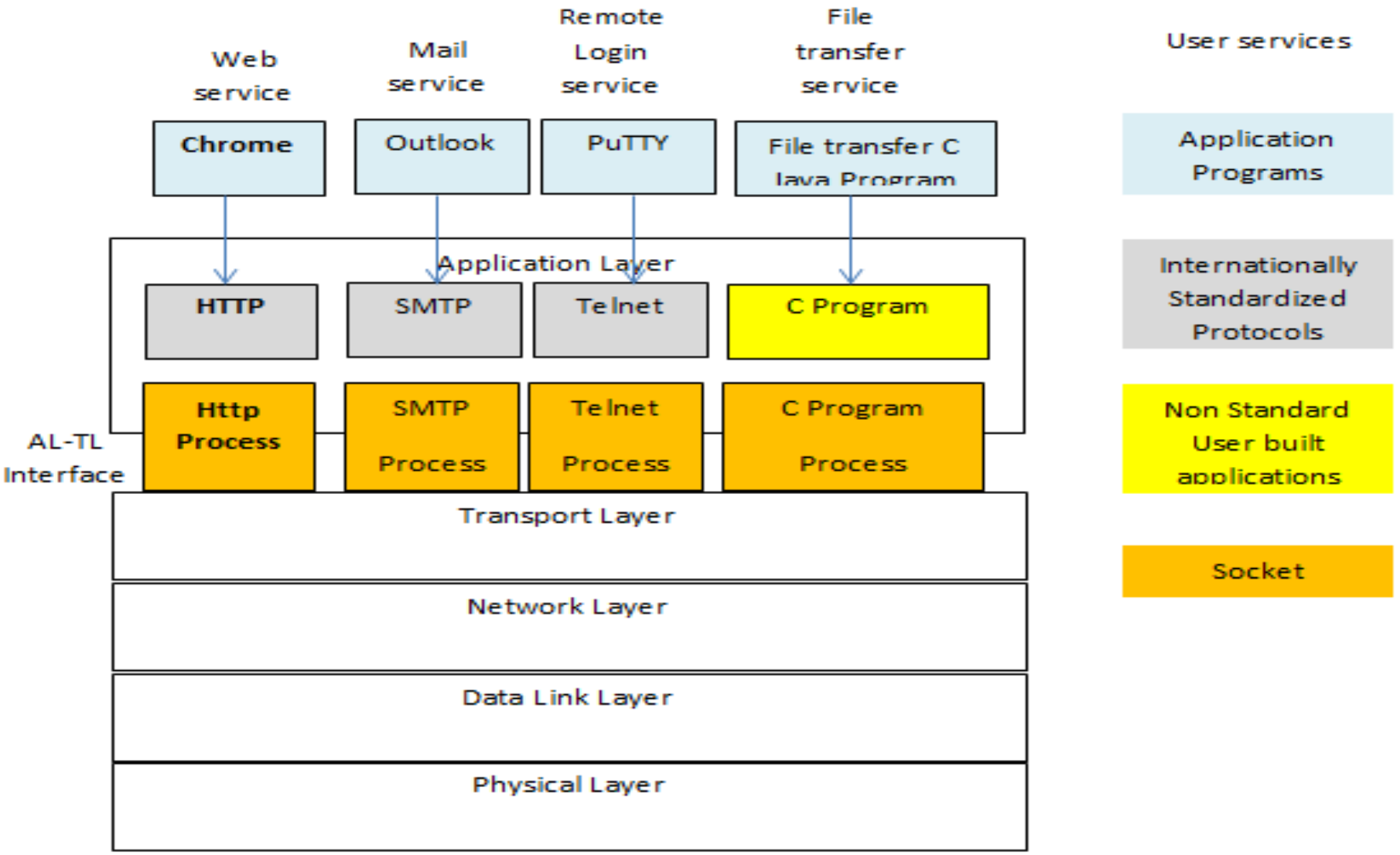
Since the application layer is the only layer that provides services to the Internet user, it allows new application protocols to be easily added to the Internet, which has been occurring during the lifetime of the Internet.

When the Internet was created, only a few application protocols were available to the users; today we cannot give a number for these protocols because new ones are being added constantly.

- It should be clear that to use the Internet we need two application programs to interact with each other: one running on a computer somewhere in the world, the other running on another computer somewhere else in the world. The two programs need to send messages to each other through the Internet infrastructure.

- The protocols in this layer do not provide services to any other protocol in the suite; they only receive services from the protocols in the transport layer.

- This means that protocols can be removed from this layer easily. New protocols can be also added to this layer as long as the new protocol can use the service provided by one of the transport-layer protocols.


Web service
Mail service
Remote Login service
File transfer service
User services
Chrome
Outlook
PuTTY
File transfer C Java Program
Application Programs
Application Layer
HTTP
SMTP
Telnet
C Program
Internationally Standardized Protocols
Http Process
SMTP Process
Telnet Process
C Program Process
Non Standard User built applications
AL-TL Interface
Transport Layer
Socket
Network Layer
Data Link Layer
Physical Layer

Standard Application-Layer Protocols

- Each standard protocol is a pair of computer programs that interact with the user and the transport layer to provide a specific service to the user.
- The study of these protocols enables a network manager to easily solve the problems that may occur when using these protocols

Nonstandard Application-Layer Protocols

- A programmer can create a nonstandard application-layer program if she can write two programs that provide service to the user by interacting with the transport layer.
- User can create a new customized application protocol to communicate using the service provided by the first four layers of the TCP/IP protocol suite without using any of the standard application programs
- User does not even need the approval of the Internet authorities if privately used, has made the Internet so popular in the world.

Application-Layer Paradigm

Traditional Paradigm: Client-Server

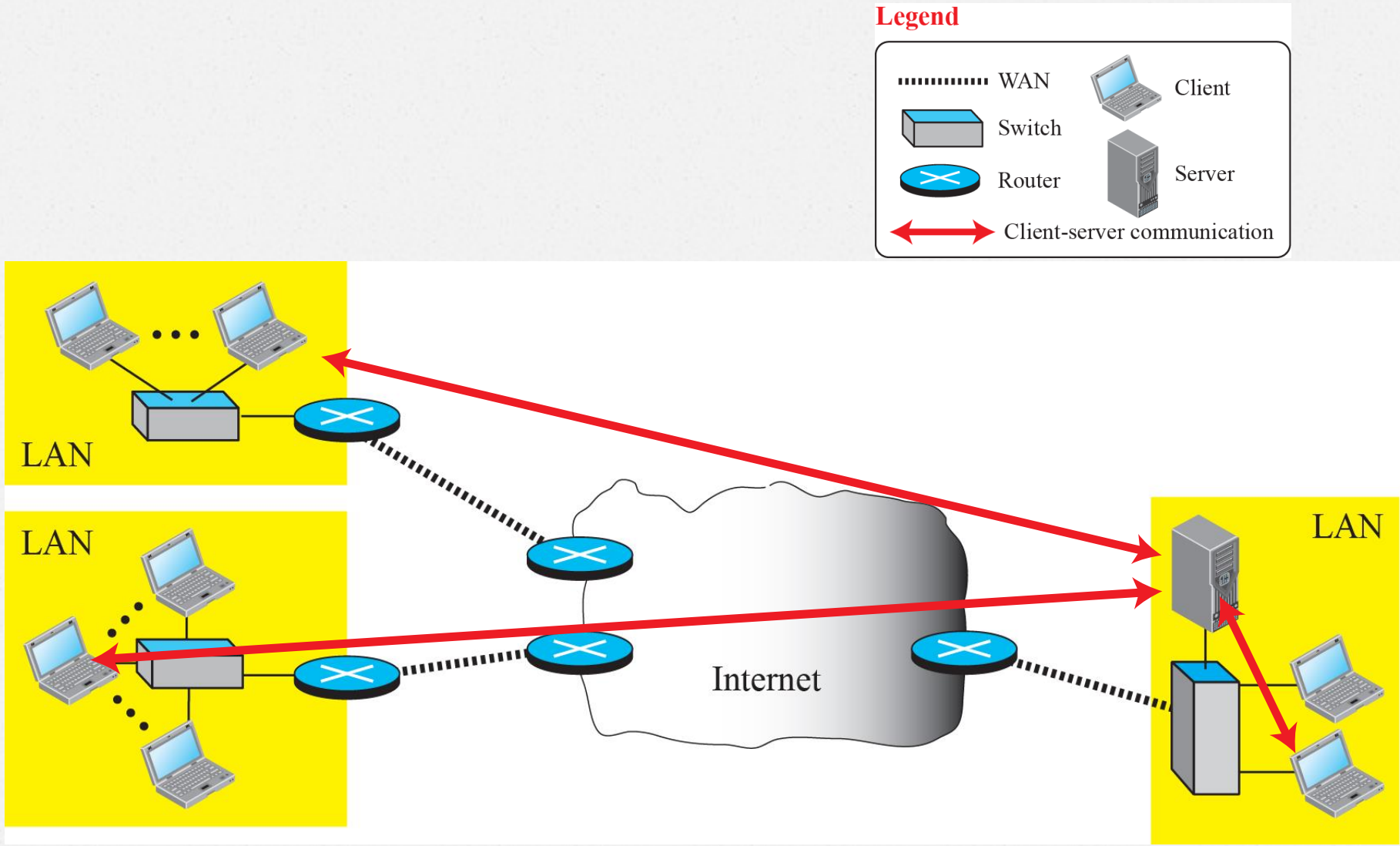
New Paradigm: Peer-to-Peer

Mixed Paradigm

CLIENT-SERVER PARADIGM

- In this paradigm, communication at the application layer is between two running application programs called processes: a client and a server.
  - A client is a running program that initializes the communication by sending a request;
  - A server is another application program that waits for a request from a client.
- The concentration of the communication load is on the shoulder of the server, which means the server should be a powerful computer.

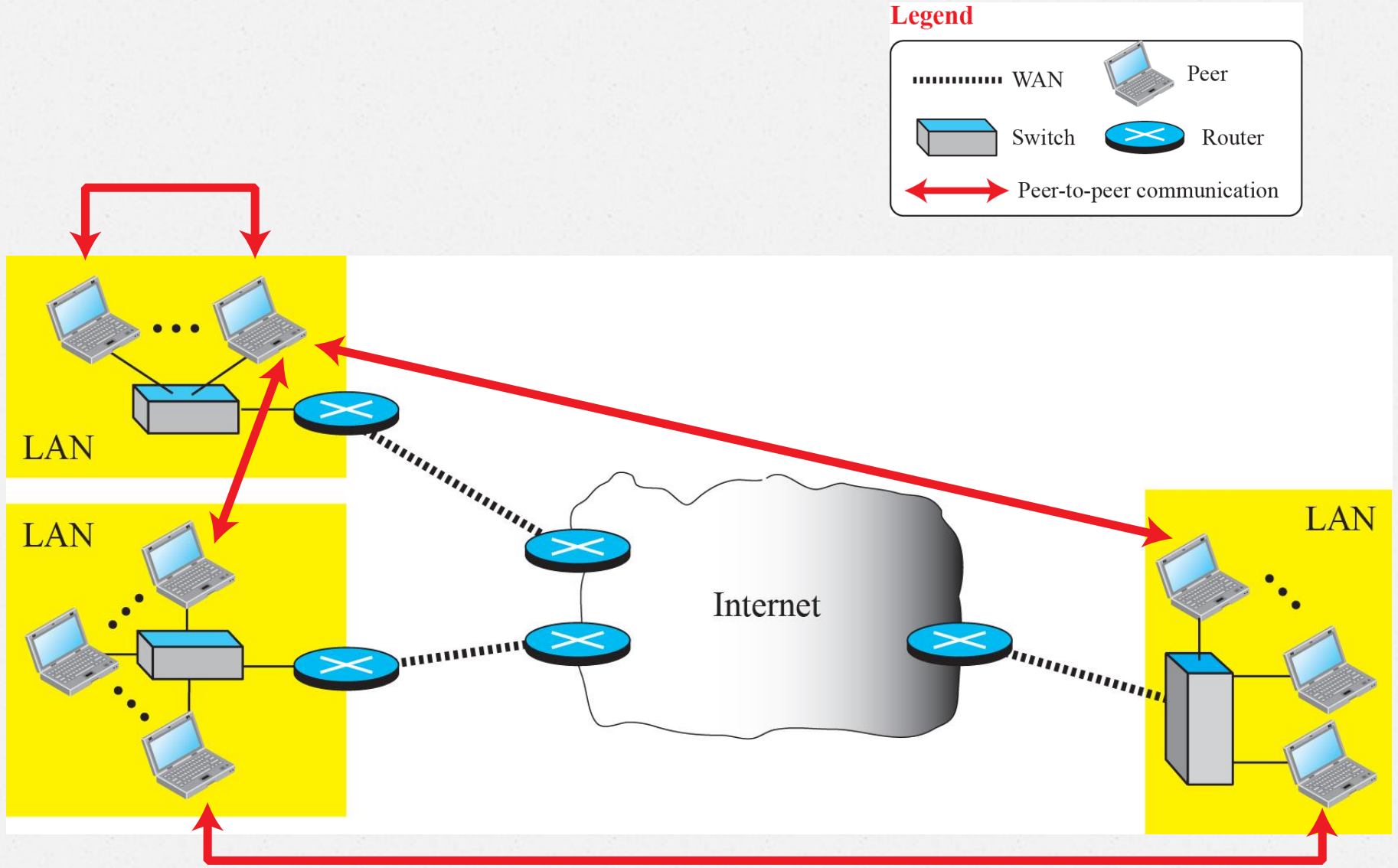
Example of a client-server paradigm



PEER to PEER PARADIGM

- A computer connected to the Internet can provide service at one time and receive service at another time. A computer can even provide and receive services at the same time.
- There is no need for a server process to be running all the time and waiting for the client processes to connect.
- easily scalable and cost-effective in eliminating the need for expensive servers to be running and maintained all the time
- It is more difficult to create secure communication between distributed services than between those controlled by some dedicated servers
- New applications, such as BitTorrent, Skype, IPTV, and Internet telephony, use this paradigm

Example of a peer-to-peer paradigm

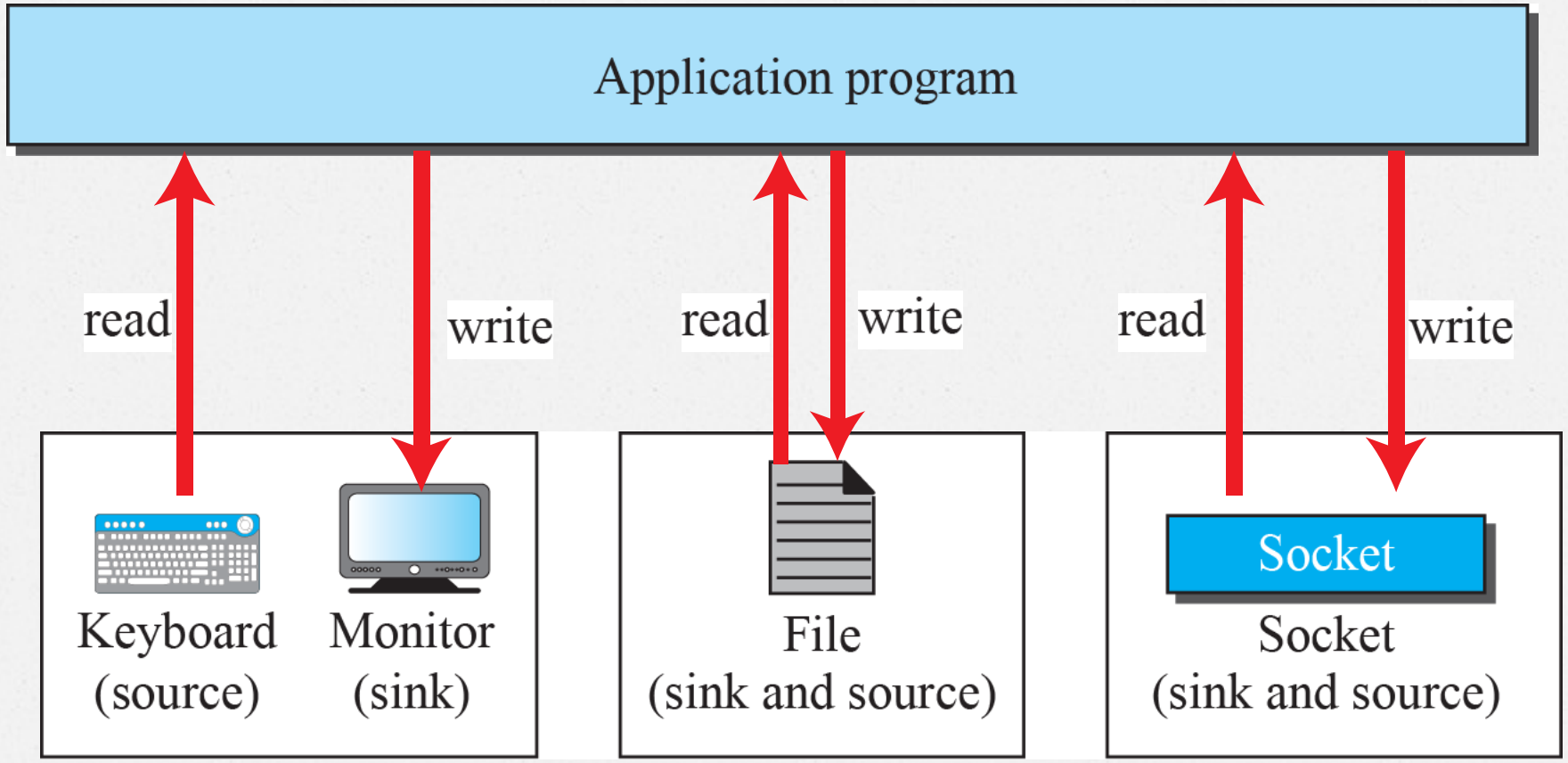


Application Programming Interface

A computer language has a set of instructions for mathematical operations, a set of instructions for string manipulation, a set of instructions for input/ output access, and so on.

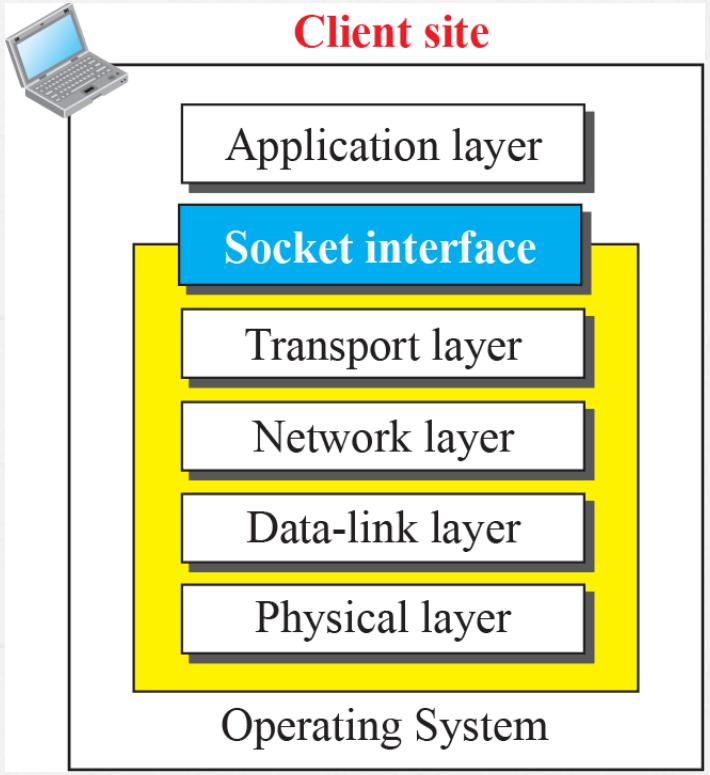
If we need a process(at AL) to be able to communicate with another process(i.e. OS), we need a new set of instructions to tell the lowest four layers of the TCP/IP suite to open the connection, send and receive data from the other end, and close the connection. A set of instructions of this kind is normally referred to as **Application Programming Interface (API).**

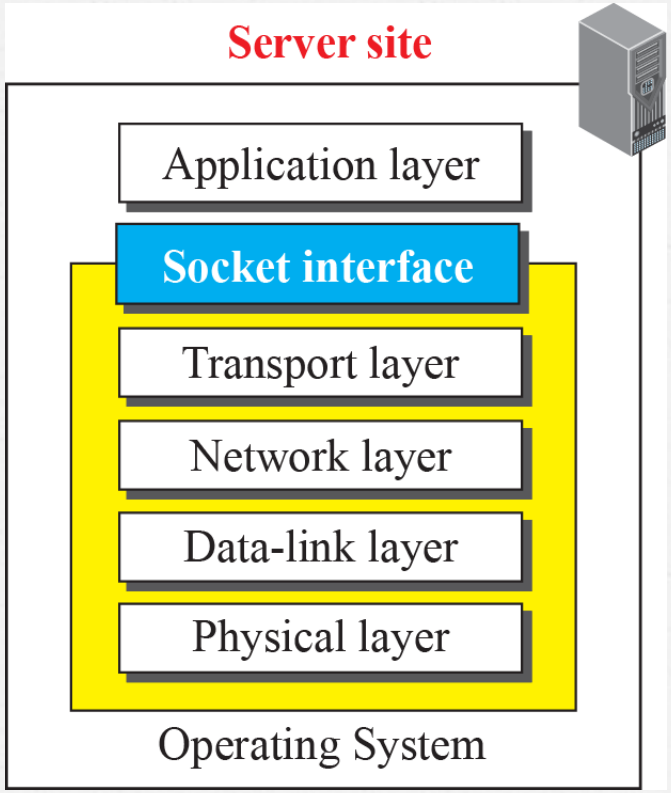
A Sockets used like other sources and sinks



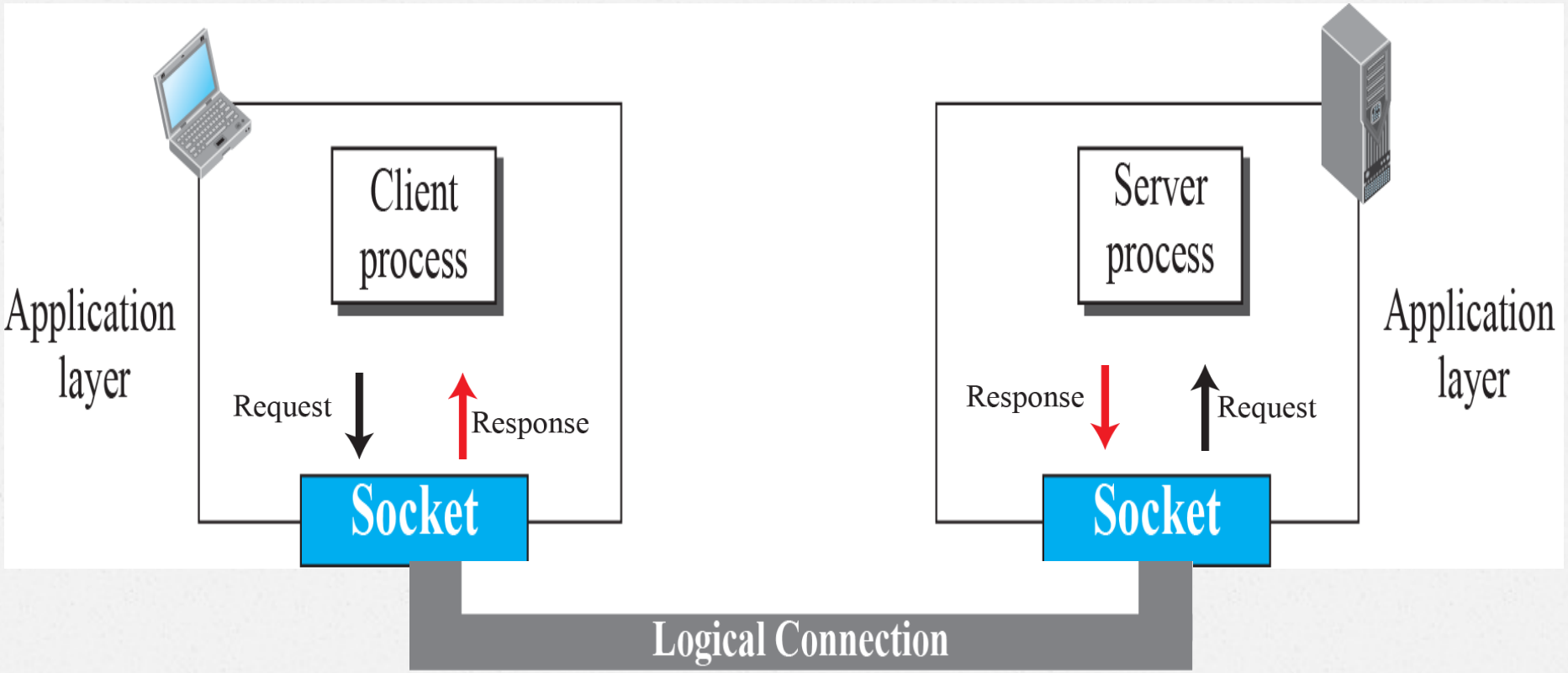
- Socket
- Socket Addresses
- Finding Socket Addresses
  - Server Site
  - Client Site

Position of the socket interface





Use of sockets in process-to-process communication



A socket address

We can find a two-level address in telephone communication. A telephone number can define an organization, and an extension can define a specific connection in that organization. The telephone number in this case is similar to the IP address, which defines the whole organization; the extension is similar to the port number, which defines the particular connection.

Using Services of Transport Layer

A pair of processes provide services to the users of the Internet, human or programs. A pair of processes, however, need to use the services provided by the transport layer for communication because there is no physical communication at the application layer. There are three common transport layer protocols in the TCP/IP suite: UDP, TCP, and SCTP.

- UDP Protocol
- TCP Protocol
- SCTP Protocol

STANDARD CLIENT-SERVER APPLICATIONS

During the lifetime of the Internet, several application programs have been developed. We do not have to redefine them, but we need to understand what they do. For each application, we also need to know the options available to us. The study of these applications can help us to create customized applications in the future.

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.

Founded 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

2.3.1 (continued)

- World Wide Web
  - Architecture –Distributed, linked
  - Web page
  - Hypertext / Hypermedia
  - Web Client(Browser)
  - Web Server
  - Uniform Resource Locator (URL)
  - Web Documents
    - Static
    - Dynamic
    - Active

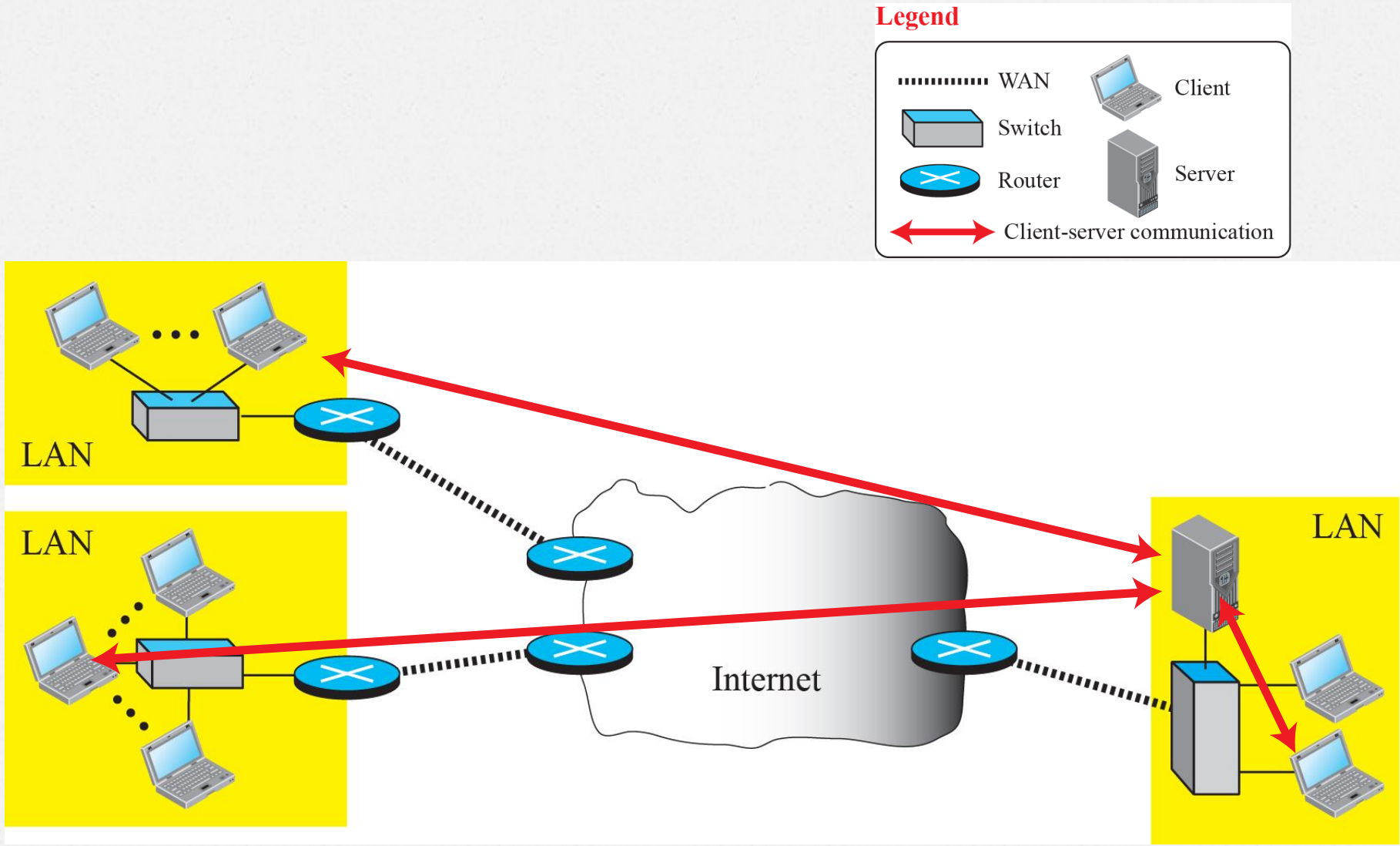
2.3.1 (continued)

- HyperText Transfer Protocol (HTTP)
  - Nonpersistent versus Persistent Connections
  - Message Formats
  - Conditional Request
  - Cookies

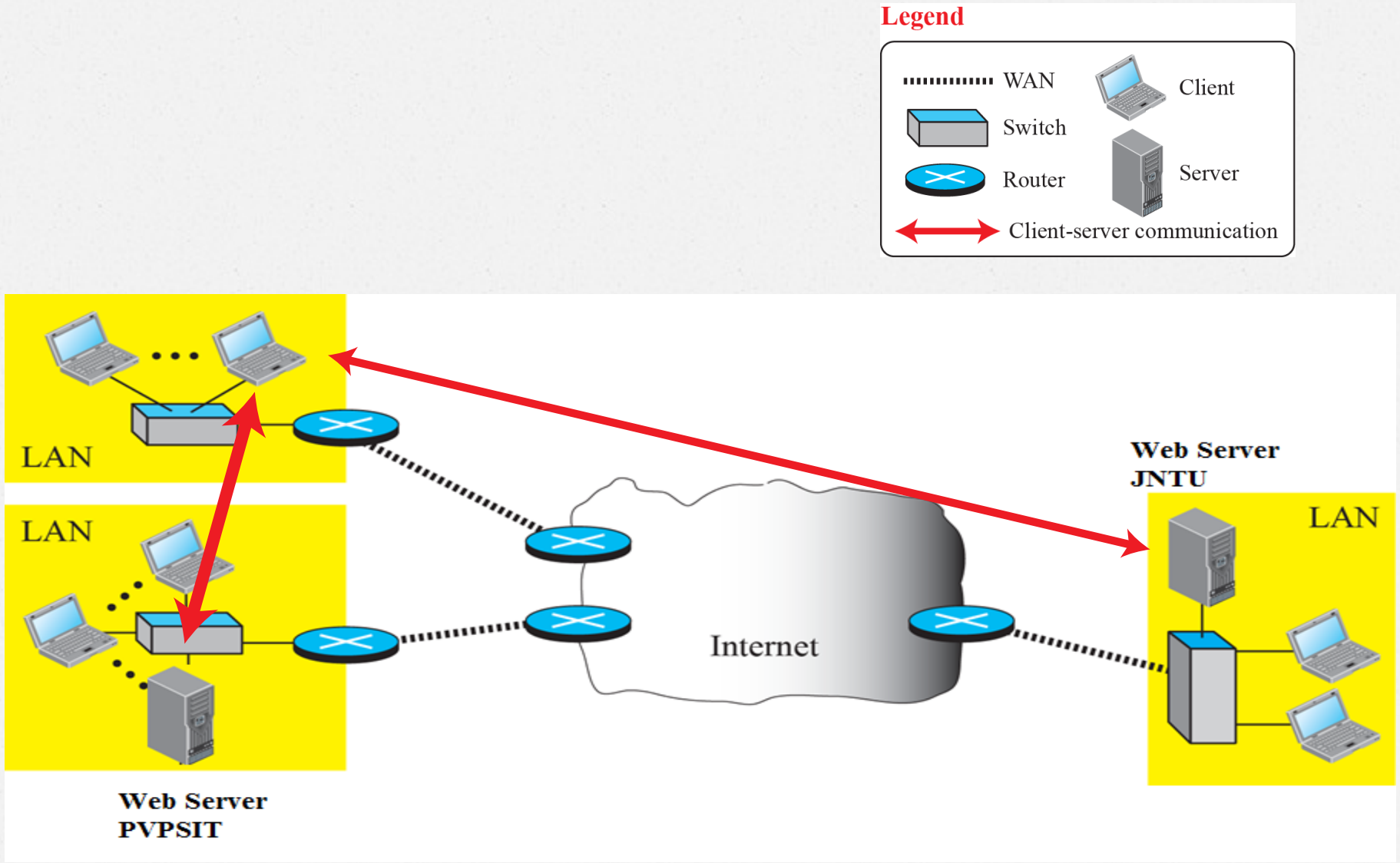
2.3.1 (continued)

- Web Caching: Proxy Server
  - Proxy Server Location
  - Cache Update
- HTTP Security

Example of a client-server paradigm Single server



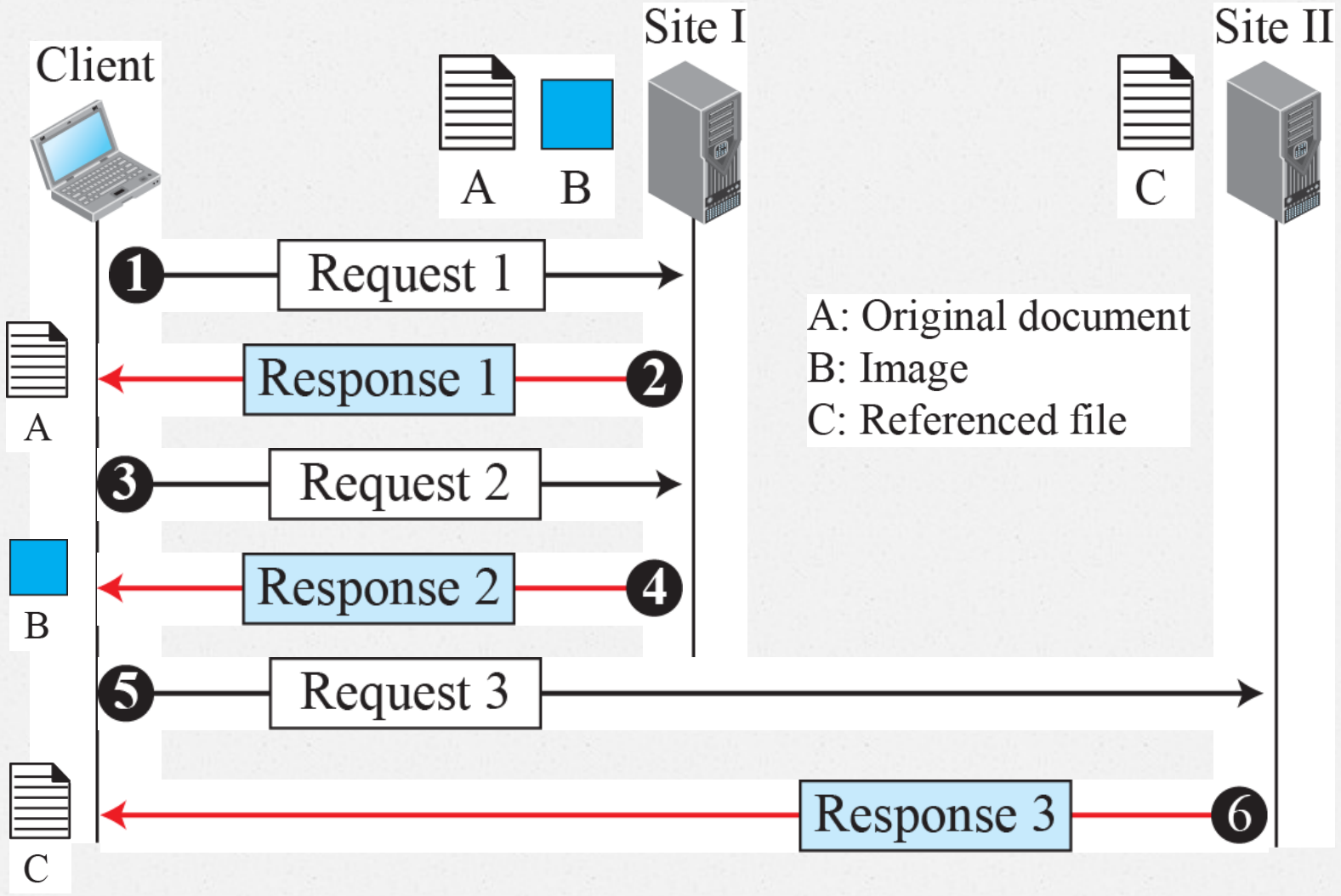
Example of a client-server paradigm multiple web servers



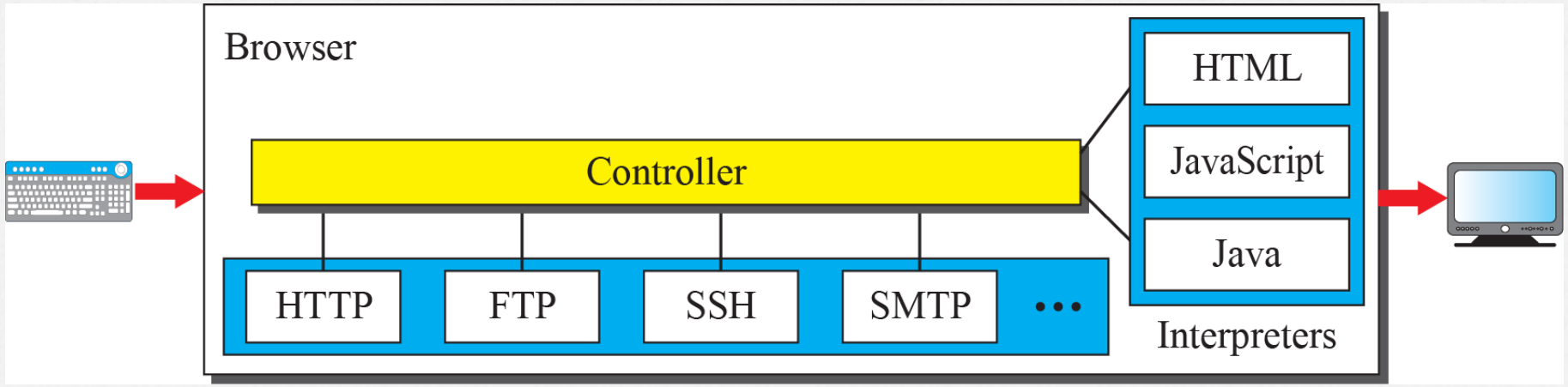
Assume we need to retrieve a scientific document that contains one reference to another text file and one reference to a large image. Figure 2.8 shows the situation.

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.

Retrieving two files and one image



Browser



- controller
- client protocols
- interpreters

Uniform Resource Locator (URL)

protocol://host/path	Used most of the time
protocol://host:port/path	Used when port number is needed

- Protocol
- Host
- Port
- Path

The URL **http://www.mhhe.com/compsci/forouzan/** defines the web page related to one of the 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).

Web Documents

Static documents

- fixed-content documents that are created and stored in a server. The client can get a copy of the document only.
- Hypertext Markup Language (HTML), Extensible Markup Language (XML), Extensible Style Language (XSL), and Extensible Hypertext Markup Language (XHTML)

Web Documents

Dynamic Documents

- created by a web server whenever a browser requests the document
- When a request arrives, the web server runs an application program or a script that creates the dynamic document
- the contents of a dynamic document may vary from one request to another
- Java Server Pages (JSP), which uses the Java language for scripting, or Active Server Pages (ASP), a Microsoft product that uses Visual Basic language for scripting, or ColdFusion

Web Documents

Active Documents

- a program or a script to be run at the client site.
- Java applets, a program written in Java on the server. It is compiled and ready to be run. The document is in bytecode (binary) format.
- Another way is to use JavaScripts but download and run the script at the client site.

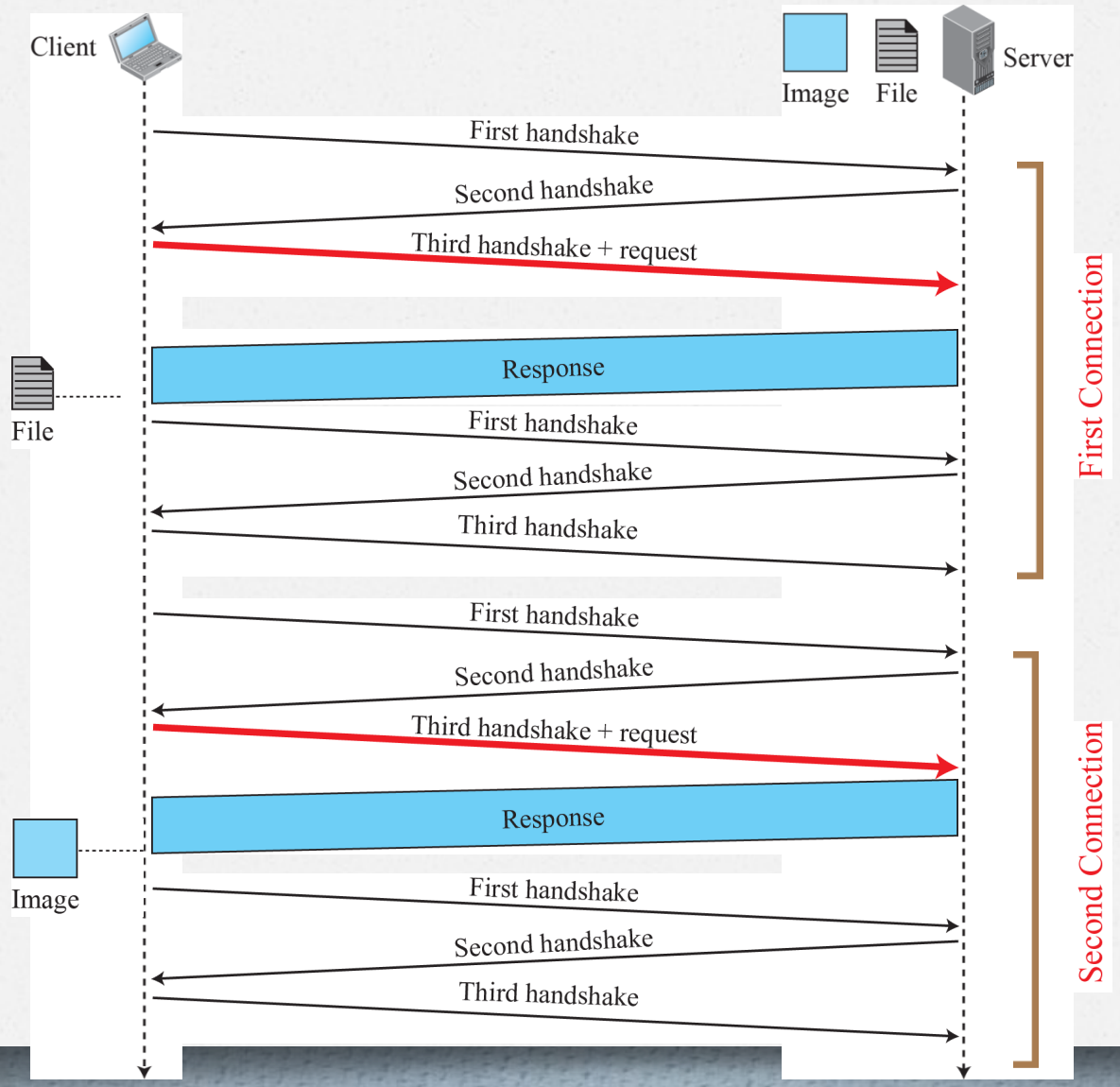
Hyper Text Transfer Protocol (HTTP)

- It is a protocol that is used to define how the client-server programs can be written to retrieve web pages from the Web.
- HTTP Client sends request and HTTP Server returns a response
- Server Port Number:80
- Client Port Number: temporary port Number
- HTTP is connection oriented i.e. uses TCP protocol which is connection oriented and reliable
- The type of connections to retrieve a web pages are:
  - Persistent Connection
  - Non Persistent Connection

Non-persistent **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.

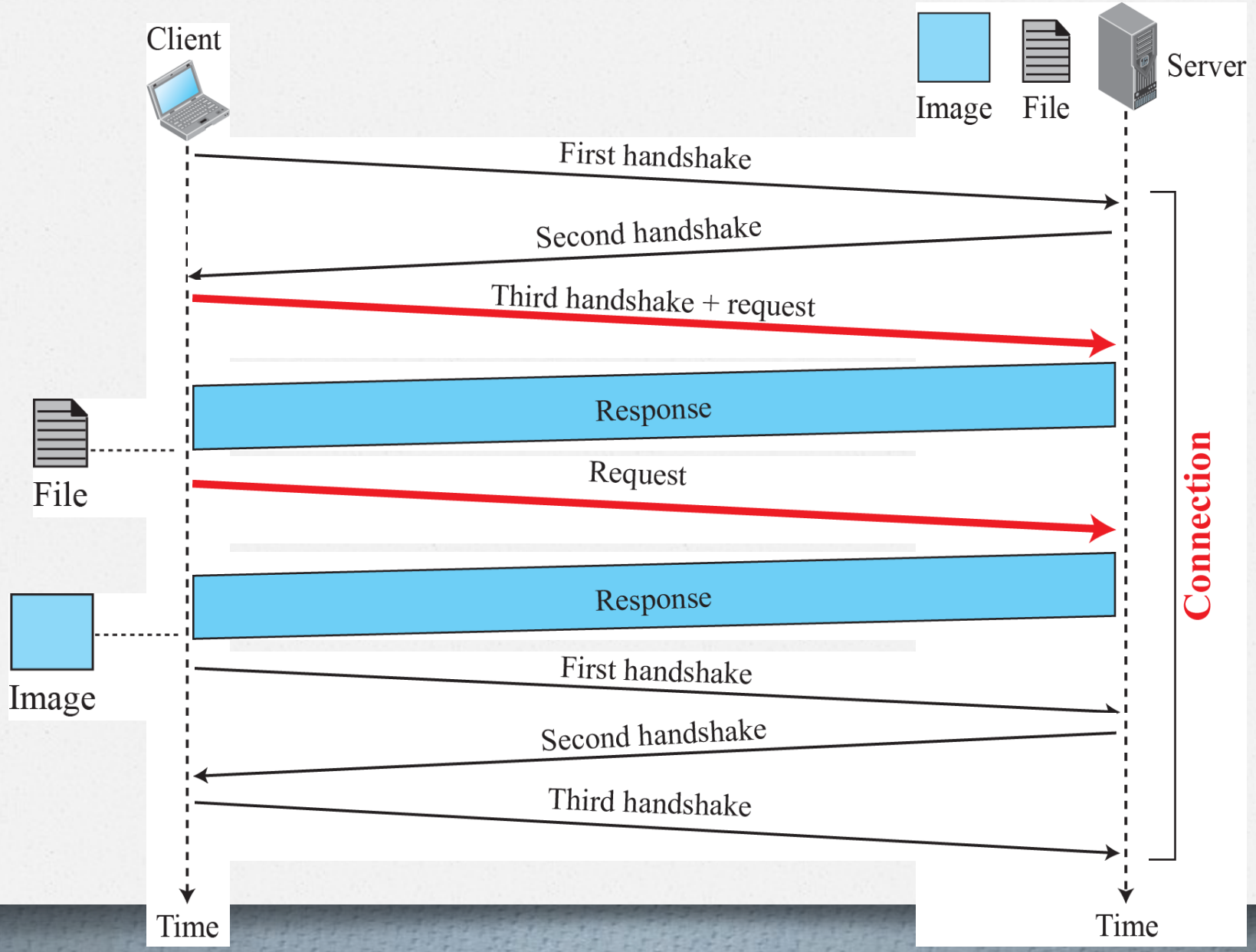
Example


Client
Image
File
Server
First handshake
Second handshake
Third handshake + request
Response
File
First handshake
Second handshake
Third handshake
First handshake
Second handshake
Third handshake + request
Response
Image
First handshake
Second handshake
Third handshake
First Connection
Second Connection

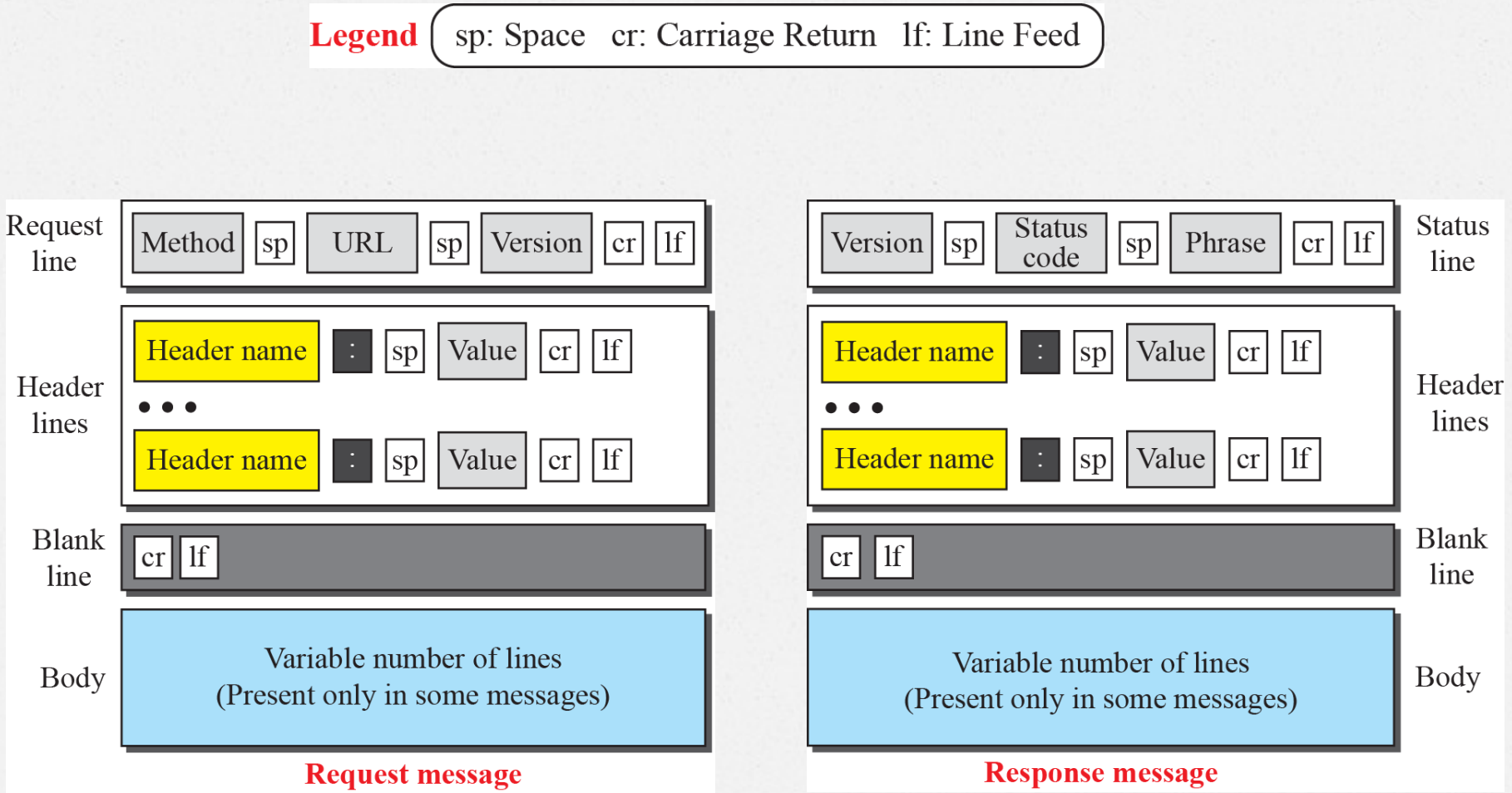
Persistent connection.

Only one connection establishment and connection termination is used, but the request for the image is sent separately.

Example


Client
Server
Image
File
First handshake
Second handshake
Third handshake + request
Response
File
Request
Response
Image
First handshake
Second handshake
Third handshake
Connection
Time
Time

Message Formats of the request and response messages



Methods

Method	*Action*
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

Request Header Names

Header	*Description*
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

Status codes

100 range are only informational,
200 range indicate a successful request.
300 range redirect the client to another URL,
400 range indicate an error at the client site.
500 range indicate an error at the server site.

The status phrase explains the status code in text form.

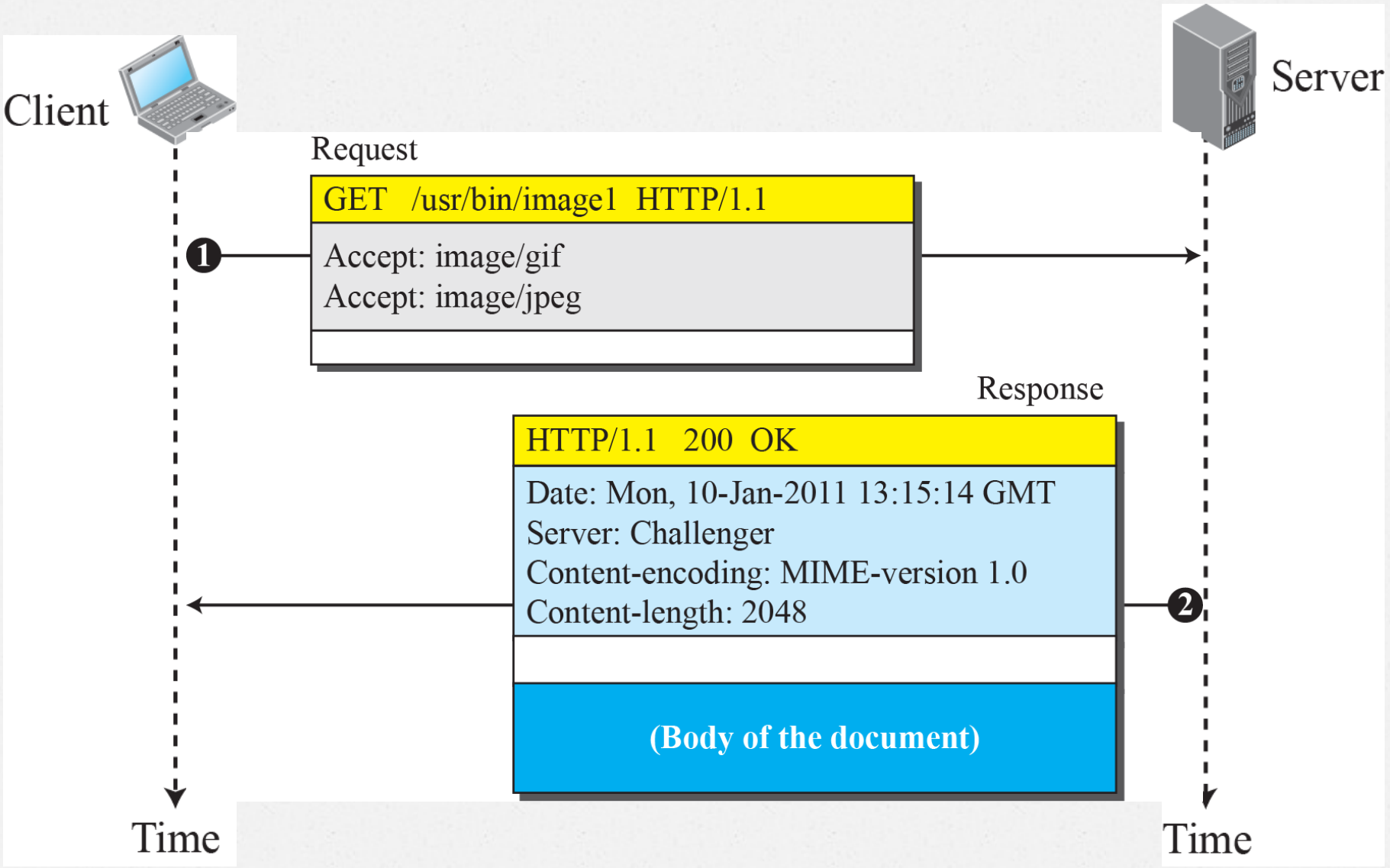
Response Header Names

Header	*Description*
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

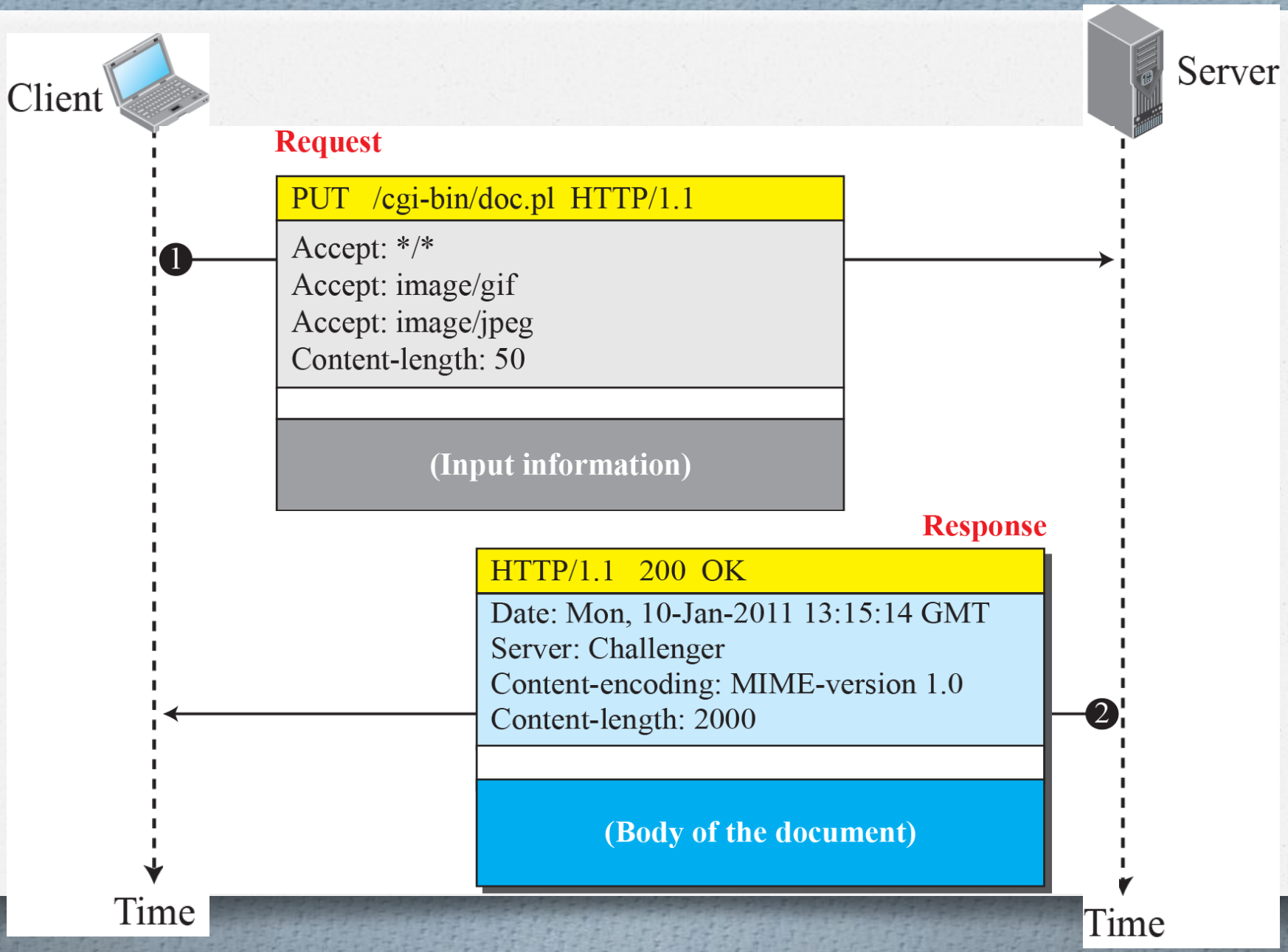
This example retrieves a document. We use the GET method to retrieve an image with the path **/usr/bin/image1**. The request line shows the method (GET), the URL, and the HTTP version (1.1). The header has two lines that show that the client can accept images in the GIF or JPEG format. The request does 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.

Example


Client
Server
Request
GET /usr/bin/image1 HTTP/1.1
Accept: image/gif
Accept: image/jpeg
1
Response
HTTP/1.1 200 OK
Date: Mon, 10-Jan-2011 13:15:14 GMT
Server: Challenger
Content-encoding: MIME-version 1.0
Content-length: 2048
(Body of the document)
2
Time
Time

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 (1.1). 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.

Example 2


Client
Server
Request
PUT /cgi-bin/doc.pl HTTP/1.1
Accept: */*
Accept: image/gif
Accept: image/jpeg
Content-length: 50
(Input information)
1
Response
HTTP/1.1 200 OK
Date: Mon, 10-Jan-2011 13:15:14 GMT
Server: Challenger
Content-encoding: MIME-version 1.0
Content-length: 2000
(Body of the document)
2
Time
Time

Conditional Request

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**
	Blank line
(Empty Body)	**Empty body**

Cookies

- The World Wide Web was originally designed as a stateless entity
- **Cookies** are messages that **web** servers pass to your **web** browser when you visit **Internet** sites.
- **cookie** is a small piece of data stored on the user's computer by the **web** browser
- Cookies are most commonly used to track website activity.
- When you visit some sites, the server gives you a cookie that acts as your identification card. Upon each return visit to that site, your browser passes that cookie back to the server.

Creating & Storing Cookies

The creation and storing of cookies depend on the implementation; however, the principle is the same

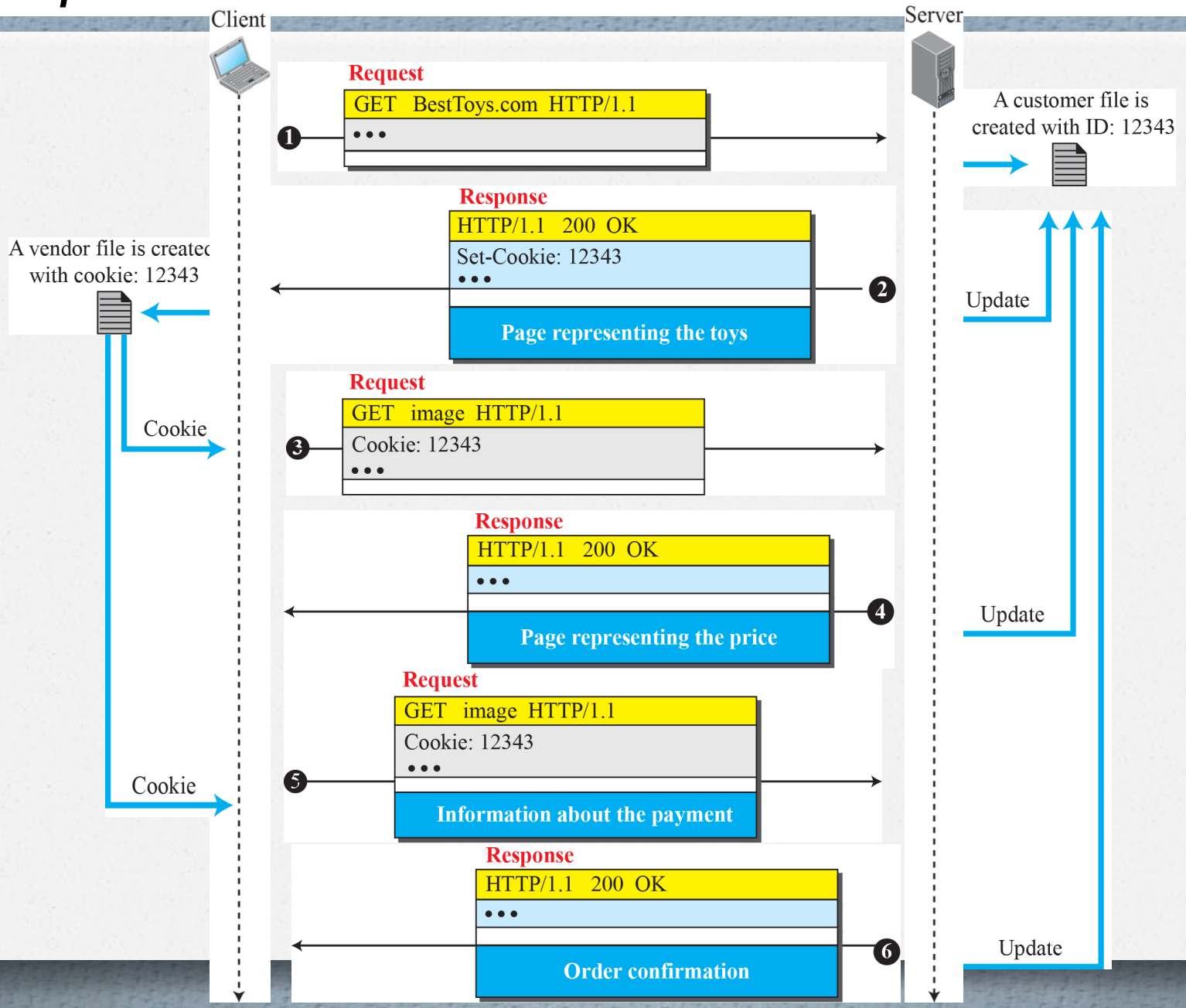
1. When a server receives a request from a client, it stores information about the client in a file or a string. The information may include the domain name of the client, the contents of the cookie (information the server has gathered about the client such as name, registration number, and so on), a timestamp, and other information depending on the implementation.
2. The server includes the cookie in the response that it sends to the client.
3. When the client receives the response, the browser stores the cookie in the cookie directory, which is sorted by the server domain name.

Using Cookies

- When a client sends a request to a server, the browser looks in the cookie directory to see if it can find a cookie sent by that server.
- If found, the cookie is included in the request.
- When the server receives the request, it knows that this is an old client, not a new one.
- Note that the contents of the cookie are never read by the browser or disclosed to the user. It is a cookie made by the server and eaten by the server

Figure 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.

Example

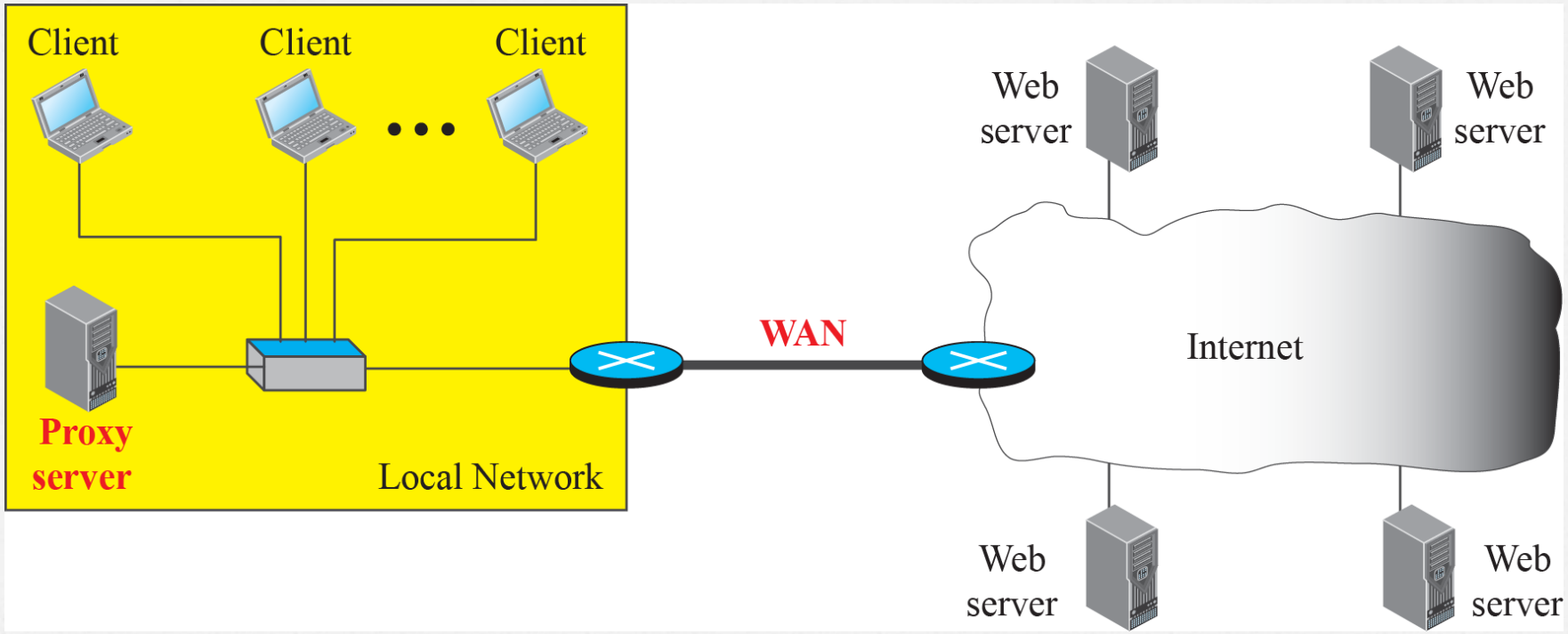

Client
Server
Request
GET BestToys.com HTTP/1.1
1
A customer file is created with ID: 12343
Response
HTTP/1.1 200 OK
Set-Cookie: 12343
2
Page representing the toys
A vendor file is created with cookie: 12343
Update
Request
GET image HTTP/1.1
Cookie: 12343
3
Cookie
Response
HTTP/1.1 200 OK
4
Page representing the price
Update
Request
GET image HTTP/1.1
Cookie: 12343
5
Cookie
Information about the payment
Response
HTTP/1.1 200 OK
6
Order confirmation
Update

Web Caching & proxy server

- HTTP supports proxy servers.
- A proxy server is a computer that keeps copies of responses to recent requests. The HTTP client sends a request to the proxy server. The proxy server checks its cache. If the response is not stored in the cache, the proxy server sends the request to the corresponding server.
- Incoming responses are sent to the proxy server and stored for future requests from other clients.
- **The proxy server reduces the load on the original server, decreases traffic, and improves latency**.
- The proxy server acts as both server and client.
- The proxy servers are normally located at the client site such as client system, LAN proxy server or at ISP.

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.

Example of a proxy server



Cache Update

- How long a response should remain in the proxy server before being deleted and replaced
  - store the list of sites whose information remains the same for a while e.g. news page could be loaded every morning
  - to add some headers to show the last modification time of the information.

HTTP Security

- HTTP per se does not provide security
- However, HTTP can be run over the Secure Socket Layer (SSL). In this case, HTTP is referred to as HTTPS.
- HTTPS provides confidentiality, client and server authentication, and data integrity.

Chapter 5

Application Layer

- INTRODUCTION
- CLIENT-SERVER PARADIGM
- STANDARD APPLICATIONS
  - HTTP
  - FTP
  - SMTP
  - Telnet
  - SSH
  - DNS

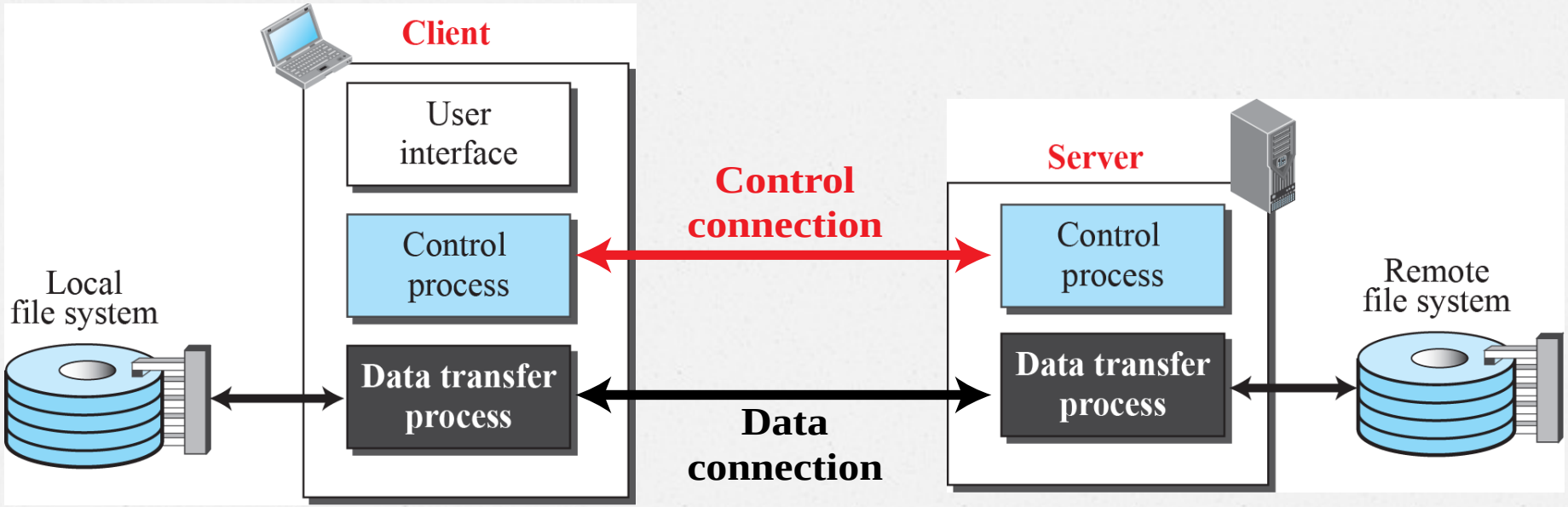
File Transfer Protocol (FTP)

- Components of FTP Client & Server
- Connections
  - Control connection
    - Commands
      - Data structure
      - File type
      - Transmission mode
      - File transfer
    - Responses
  - Data connection
- FTP Security

File Transfer Protocol(FTP)

- Standard protocol provided by TCP/IP for copying a file from one host to another.
- Connection Oriented Protocol
- Though HTTP can also transfer files FTP is ideal protocol and better choice for large files.
- Several issues have to be handled by FTP, for example, two systems may use different file name conventions, different ways to represent data, different directory structures.
- Components of FTP Client: user interface, client control process, and the client data transfer process.
- Components of FTP Server: the server control process and the server data transfer process

FTP



FTP uses two well-known TCP ports:

- port 21 is used for the control connection, and
- port 20 is used for the data connection.

Lifetimes of Two Connections

- FTP Uses Two Connections
  - The control connection is made between the control processes.
  - The data connection is made between the data transfer processes.
- Separation of commands and data transfer makes FTP more efficient
- 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.
- When a user starts an FTP session, the control connection opens. While the control connection is open, the data connection can be opened and closed multiple times if several files are transferred.

Control Connection

- Port-21
- It uses the NVT ASCII character set
- **commands** are sent from the client to the server.(Only one command)
- **responses** are sent from the server to the client.(Only one command)
- Commands, which are sent from the FTP client control process, are in the form of ASCII uppercase, which may or may not be followed by an argument
- Each line is terminated with a two-character (carriage return and line feed) end-of-line token
- Every FTP command generates at least one response
- We want to transfer files through the data connection. The **client must define the type of file to be transferred**, the structure of the data, and the transmission mode using commands in control connection.

Some FTP commands

Command	*Argument(s)*	*Description*
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

Some FTP commands (continued)

Command	*Argument(s)*	*Description*
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

Responses in FTP

- A response has two parts: **a three-digit number** followed by **text.**
  - The numeric part defines the code
    - The first digit defines the status of the command. (i.e. good, bad, incomplete)
    - The second digit defines the area in which the status applies. (i.e. syntax, file system, authentication, connection..)
    - The third digit provides additional information.
  - The text part defines needed parameters or further explanations

Some responses in FTP

Code	Description	Code	Description
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

Data Structure

Three kinds of interpretations

- The **file structure** format (used by default) has no structure. It is a continuous stream of bytes.
- In the **record structure**, the file is divided into records. This can be used only with text files.
- In the **page structure**, the file is divided into pages, with each page having a page number and a page header. The pages can be stored and accessed randomly or sequentially

File Type

FTP can transfer one of the following file types across the data connection: ASCII file, EBCDIC file, or image file.

Transmission Mode

Three kinds of modes

- The **stream mode** is the default mode; data are delivered from FTP to TCP as a continuous stream of bytes..
- The **block mode**, data can be delivered from FTP to TCP in blocks. In this case, each block is preceded by a 3-byte header. The first byte is called the block descriptor; the next two bytes define the size of the block in bytes.
- In the **compressed mode**, the file is compressed and the file is delivered from FTP to TCP as a continuous stream of bytes

File Transfer

File transfer in FTP means one of three things:

- retrieving a file (server to client),
- storing a file (client to server), and
- directory listing (server to client).

Data Connection

- Port 20
- Steps in data connection
  1. The client, not the server, issues a passive open using an ephemeral port. This must be done by the client because it is the client that issues the commands for transferring files.
  2. The client sends this port number to the server using the PORT command.
  3. The server receives the port number and issues an active open using the wellknown port 20 and the received ephemeral port number.

FTP Connection Modes

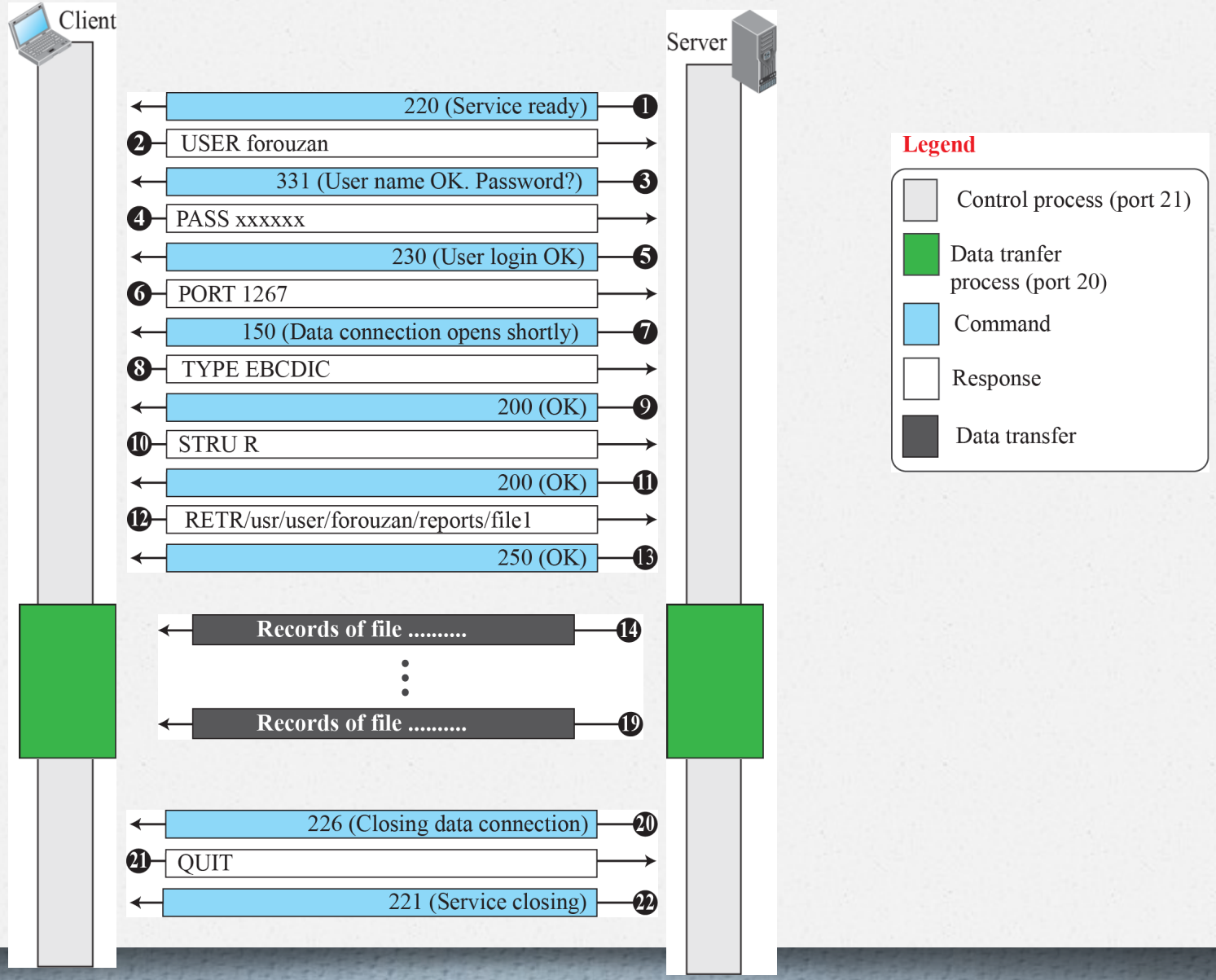
FTP may operate in an active or a passive mode, which determines how a data connection is established. In both cases, a client creates a TCP control connection to an FTP server command port 21.

- In the ***active* mode**, the client starts listening on a random port for incoming data connections from the server (the client sends the FTP command PORT to inform the server on which port it is listening).
- In the ***passive* mode**, the client uses the control connection to send a PASV command to the server and then receives a server IP address and server port number from the server, which the client then uses to open a data connection to the server IP address and server port number received.

Example

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.

Example


Client
Server
220 (Service ready)
USER forouzan
331 (User name OK. Password?)
PASS xxxxxx
230 (User login OK)
PORT 1267
150 (Data connection opens shortly)
TYPE EBCDIC
200 (OK)
STRU R
200 (OK)
RETR/usr/user/forouzan/reports/file1
250 (OK)
Records of file .....
Records of file .....
226 (Closing data connection)
QUIT
221 (Service closing)
Legend
Control process (port 21)
Data tranfer process (port 20)
Command
Response
Data transfer

Example

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.
```

Security for FTP

- 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.
- one can add a Secure Socket Layer between the FTP application layer and the TCP layer. In this case FTP is called SSL-FTP.

Chapter 5

Application Layer

- INTRODUCTION
- CLIENT-SERVER PARADIGM
- STANDARD APPLICATIONS
  - HTTP
  - FTP
  - **SMTP**
  - Telnet
  - SSH
  - DNS

Electronic Mail

- Architecture
- User Agent
  - Sending Mail
  - Receiving Mail
  - Addresses
  - Mailing List or Group List
- Message Transfer Agent: SMTP
  - Commands and Responses
  - Mail Transfer Phases
- Message Access Agent: POP and IMAP
- MIME
  - MIME Headers
- Web-Based Mail
- E-Mail Security

Electronic Mail

Electronic mail (or e-mail) allows users to exchange messages. The nature of this application, however, is different from other applications discussed so far. In an application such as HTTP or FTP, the server program is running all the time, waiting for a request from a client. When the request arrives, the server provides the service. In the case of electronic mail, the situation is different.

First, e-mail is considered a one-way transaction. When Alice sends an e-mail to Bob, she may expect a response, but this is not a mandate. Bob may or may not respond. If he does respond, it is another one-way transaction. Second, it is neither feasible nor logical for Bob to run a server program and wait until someone sends an e-mail to him. Bob may turn off his computer when he is not using it. This means that the idea of client/ server programming should be implemented in another way: using some intermediate computers (servers).

Architecture

Mail Box:

- The administrator has created one mailbox for each user where the received messages are stored. A mailbox is part of a server hard drive, a special file with permission restrictions.
- The administrator has also created a queue (spool) to store messages waiting to be sent.

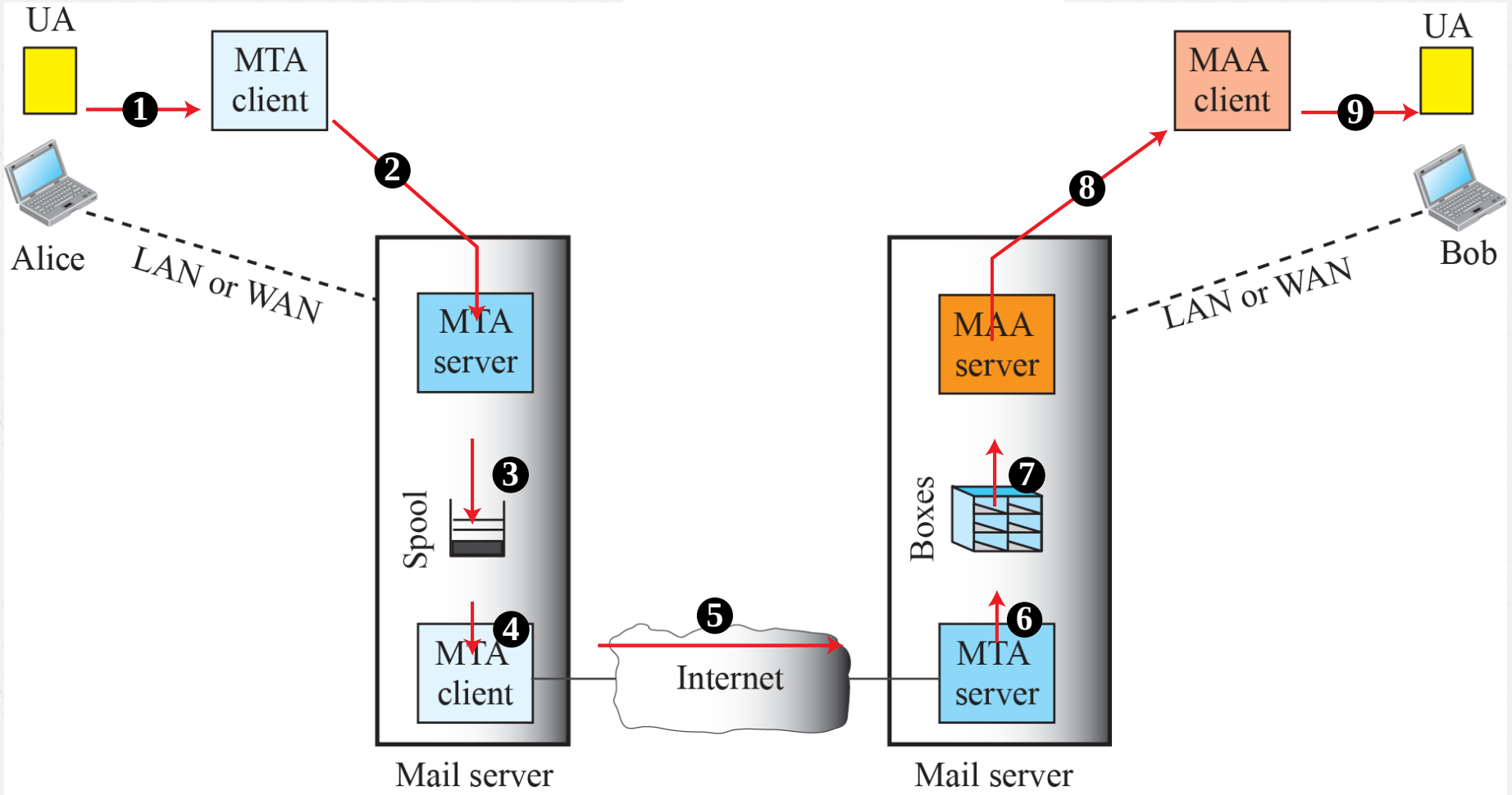
Uses three different agents:

- **a User Agent (UA),**
- **a Mail Transfer Agent (MTA), and**
- **a Message Access Agent (MAA).**

The electronic mail system needs two UAs, two pairs of MTAs (client and server), and a pair of MAAs (client and server).

Common scenario

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



User Agent

- It provides service to the user to make the process of sending and receiving a message easier. A user agent is a software package (program) that composes, reads, replies to, and forwards messages. It also handles local mailboxes on the user computers.
- Two types of user agents:

Command-driven:

- belong to the early days of electronic mail. They are still present as the underlying user agents.
- A command-driven user agent normally accepts a one character
- command from the keyboard to perform its task.
- e.g. mail, pine, and elm

GUI-based:

- They contain graphical user interface (GUI) components that allow the user to interact with the software by using both the keyboard and the mouse.
- e.g. Eudora and Outlook

The main functionalities of User agent are:

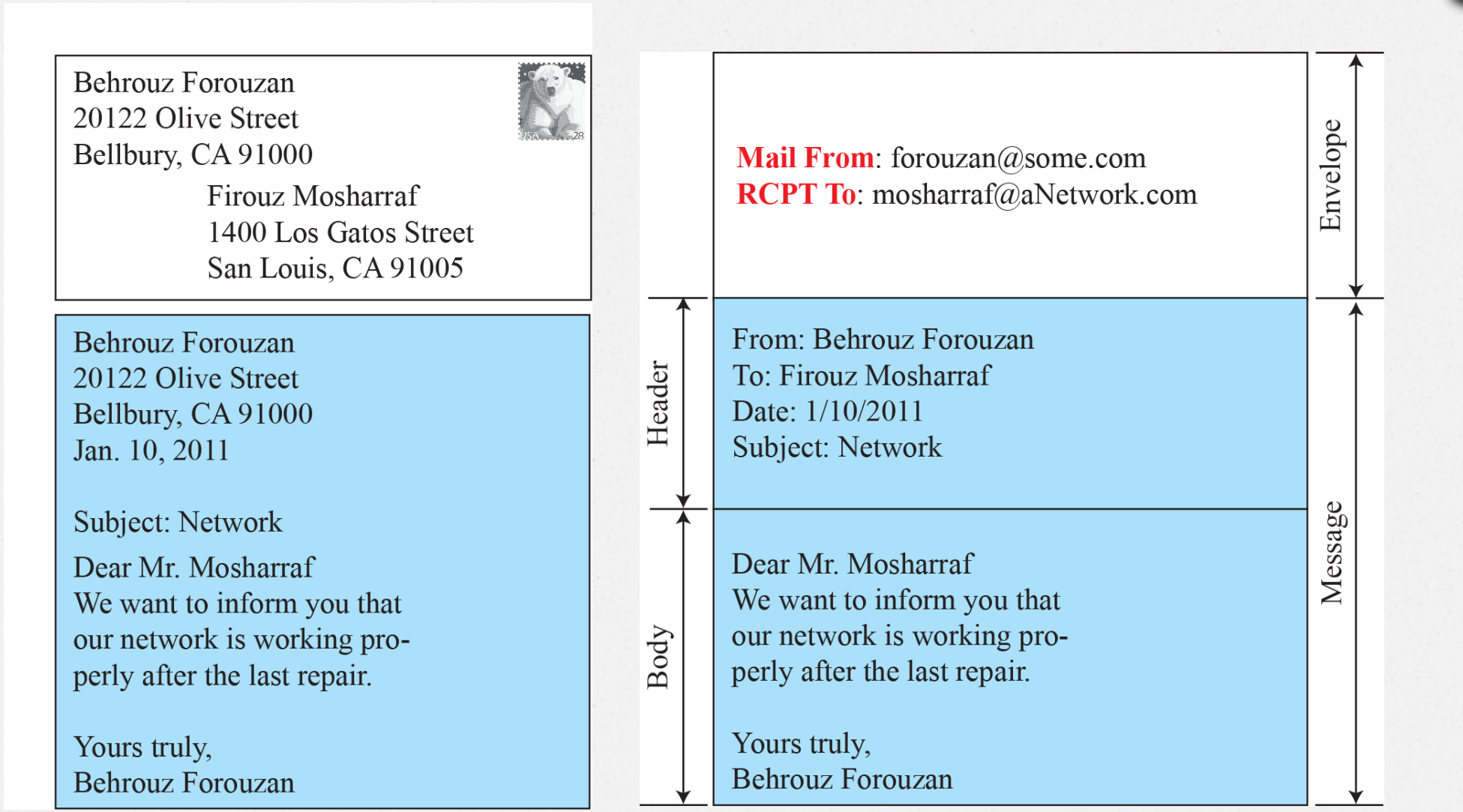
Sending Mail

- To send mail, the user, through the UA, creates mail that looks very similar to postal mail. It has an envelope and a message.
  - The envelope usually contains the sender address, the receiver address, and other information.
  - The message contains the header and the body.

Receiving Mail

- The user agent is triggered by the user (or a timer). If a user has mail, the UA informs the user with a notice.
- If the user is ready to read the mail, a list is displayed in which each line contains a summary of the information about a particular message in the mailbox.

Format of an e-mail



Postal mail **Electronic mail**

E-mail address

To deliver mail, a mail handling system must use an addressing system with unique addresses.

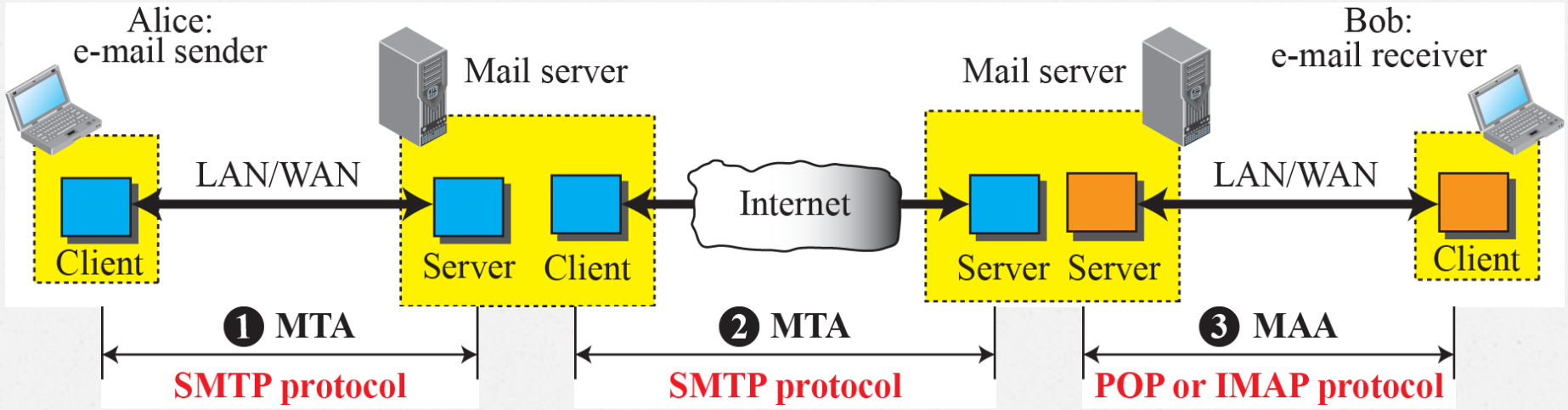
The **local part** defines the name of a special file, called the user mailbox, where all the mail received for a user is stored for retrieval by the message access agent.

The second part of the address is the **domain name**. An organization usually selects one or more hosts to receive and send e-mail; they are sometimes called mail servers or exchangers. The domain name assigned to each mail exchanger either comes from the DNS database or is a logical name

Mailing List or Group List

Electronic mail allows one name, an alias, to represent several different e-mail addresses; this is called a mailing list.
Every time a message is to be sent, the system checks the recipient's name against the alias database; if there is a mailing list for the defined alias, separate messages, one for each entry in the list, must be prepared and handed to the MTA.

Protocols used in electronic mail



Simple Mail Transfer Protocol (SMTP) → MTA

Post Office Protocol(POP3) and Internet Mail Access Protocol(IMAP4) → MAA

SMTP: Well known port 25 POP/IMAP: Well known port 110

Message Transfer Agent: SMTP

- The formal protocol that defines the MTA client and server in the Internet
- SMTP is used two times, between **the sender and the sender's mail server** and **between the two mail servers**.
- SMTP uses commands and responses to transfer messages between an MTA client and an MTA server.
  - The command is from an MTA client to an MTA server;
  - the response is from an MTA server to the MTA client.

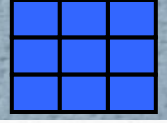
Each command or reply is terminated by a two character (carriage return and line feed) end-of-line token.

SMTP Commands:

- Commands are sent from the client to the server
- It consists of a keyword followed by zero or more arguments

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 *or* the mailbox of the recipient
SMAL FROM	Intended recipient	Specifies that the mail be delivered to the terminal *and* the mailbox of the recipient



SMTP responses

SMTP Responses:

- Responses are sent from the server to the client.
- A response is a three digit code that may be followed by additional textual information.

Code	*Description*
	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

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

Mail Transfer Phases

The process of transferring a mail message occurs in three phases: connection establishment, mail transfer, and connection termination.

Connection Establishment:
After a client has made a TCP connection to the well known port 25, the SMTP server starts the connection phase. This phase involves the following three steps:

1. The server sends code 220 (service ready) to tell the client that it is ready to receive mail. If the server is not ready, it sends code 421 (service not available).
2. The client sends the HELO message to identify itself, using its domain name address. This step is necessary to inform the server of the domain name of the client.
3. The server responds with code 250 (request command completed) or some other code depending on the situation.

Message Transfer

After connection has been established between the SMTP client and server, a single message between a sender and one or more recipients can be exchanged. This phase involves eight steps: (3&4 for more recpients)

1. The client sends the MAIL FROM message to introduce the sender of the message. It includes the mail address of the sender (mailbox and the domain name). This step is needed to give the server the return mail address for returning errors and reporting messages.
2. The server responds with code 250 or some other appropriate code.
3. The client sends the RCPT TO (recipient) message, which includes the mail address of the recipient.
4. The server responds with code 250 or some other appropriate code.
5. The client sends the DATA message to initialize the message transfer.
6. The server responds with code 354 (start mail input) or some other appropriate message.
7. The client sends the contents of the message in consecutive lines. Each line is terminated by a two-character end-of-line token (carriage return and line feed). The message is terminated by a line containing just one period.
8. The server responds with code 250 (OK) or some other appropriate code.

Connection Termination

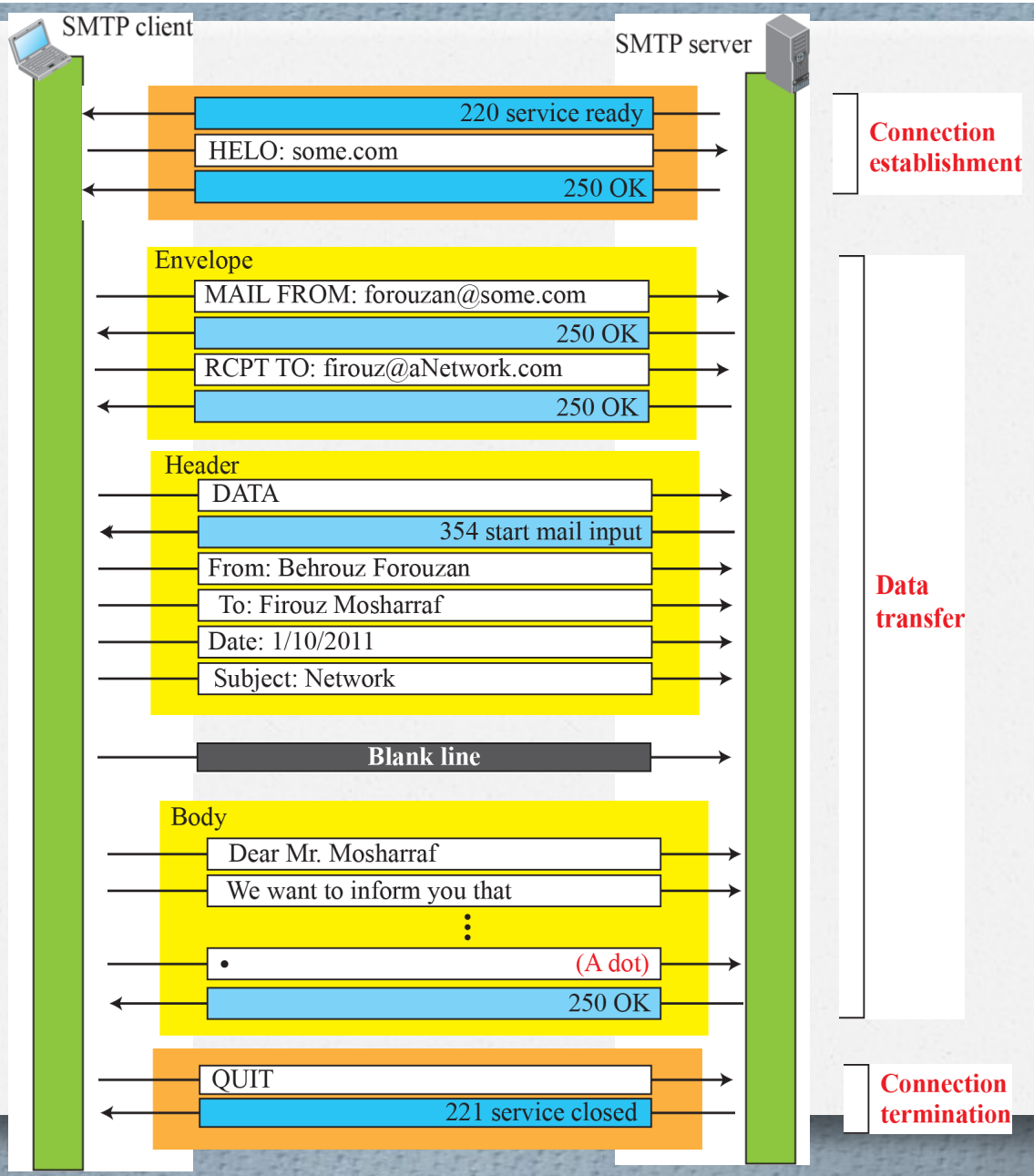
After the message is transferred successfully, the client terminates the connection. This phase involves two steps.

1. The client sends the QUIT command.
2. The server responds with code 221 or some other appropriate code.

Example

To show the three mail transfer phases, we show all of the steps described above using the information depicted in Figure. 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.

Example


SMTP client
SMTP server
220 service ready
HELO: some.com
250 OK
Envelope
MAIL FROM: forouzan@some.com
250 OK
RCPT TO: firouz@aNetwork.com
250 OK
Header
DATA
354 start mail input
From: Behrouz Forouzan
To: Firouz Mosharraf
Date: 1/10/2011
Subject: Network
Blank line
Body
Dear Mr. Mosharraf
We want to inform you that
•
(A dot)
250 OK
QUIT
221 service closed
Connection establishment
Data transfer
Connection termination

Message Access Agent: POP and IMAP

- The first and second stages of mail delivery use SMTP. However, SMTP is not involved in the third stage because SMTP is a push protocol; it pushes the message from the client to the server.
- the third stage needs a pull protocol; the client must pull messages from the server. The direction of the bulk data is from the server to the client. The third stage uses a message access agent.
- Currently two message access protocols are available:
  - Post Office Protocol, version 3 (POP3) and
  - Internet Mail Access Protocol, version 4 (IMAP4).

Post Office Protocol, Version 3 (POP3)

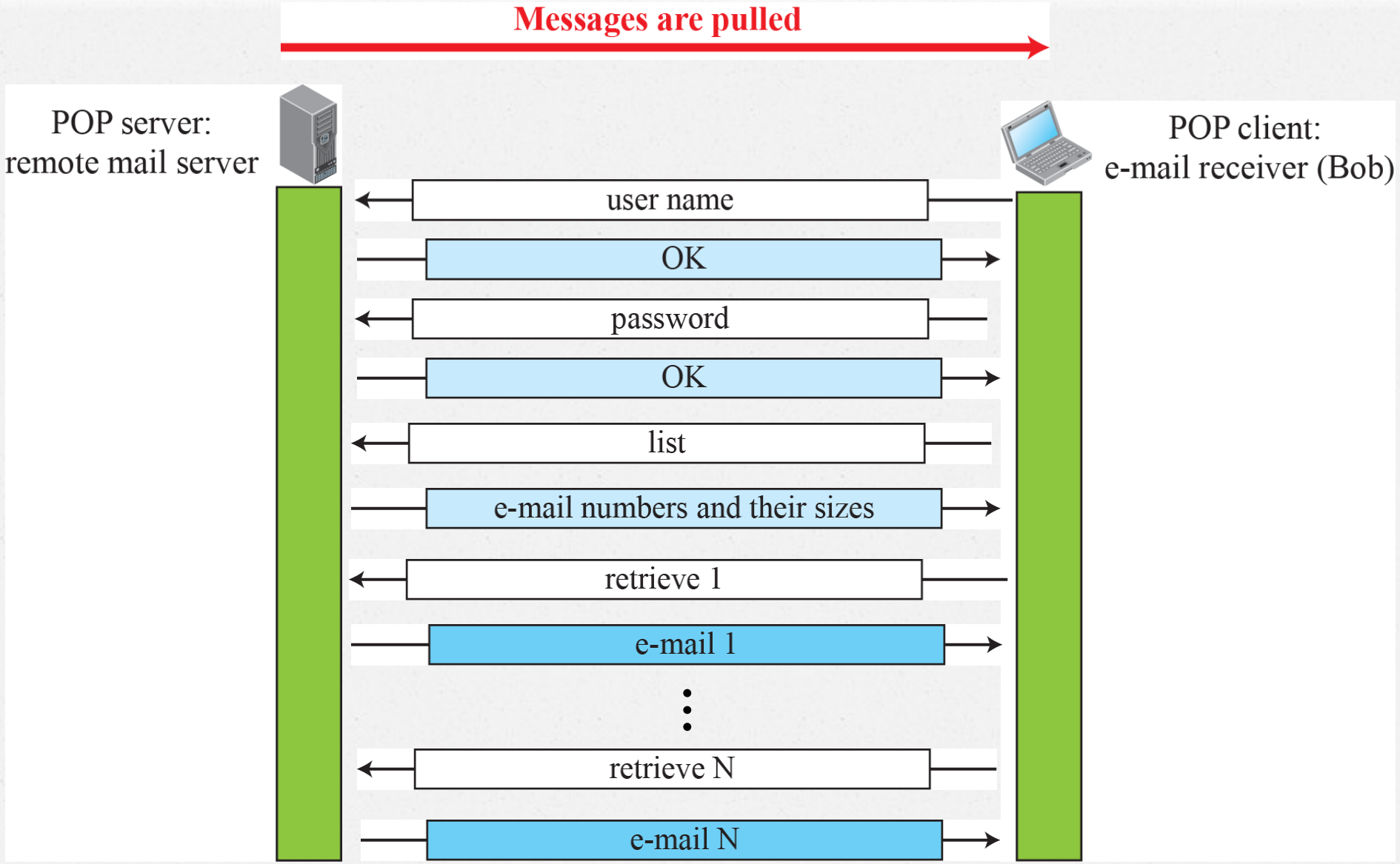
The client POP3 software is installed on the recipient computer; the server POP3 software is installed on the mail server.

- The client opens a connection to the server on TCP port 110.
- It then sends its user name and password to access the mailbox.
- The user can then list and retrieve the mail messages, one by one.

POP3 has two modes: the delete mode and the keep mode.

- In the **delete mode**, the mail is deleted from the mailbox after each retrieval. The delete mode is normally used when the user is working at her permanent computer and can save and organize the received mail after reading or replying.
- In the **keep mode**, the mail remains in the mailbox after retrieval. The keep mode is normally used when the user accesses her mail away from her primary computer (for example, from a laptop). The mail is read but kept in the system for later retrieval and organizing.

POP3


Messages are pulled
POP server:
remote mail server
POP client:
e-mail receiver (Bob)
user name
OK
password
OK
list
e-mail numbers and their sizes
retrieve 1
e-mail 1
retrieve N
e-mail N

Internet Mail Access Protocol, Version 4 (IMAP4)

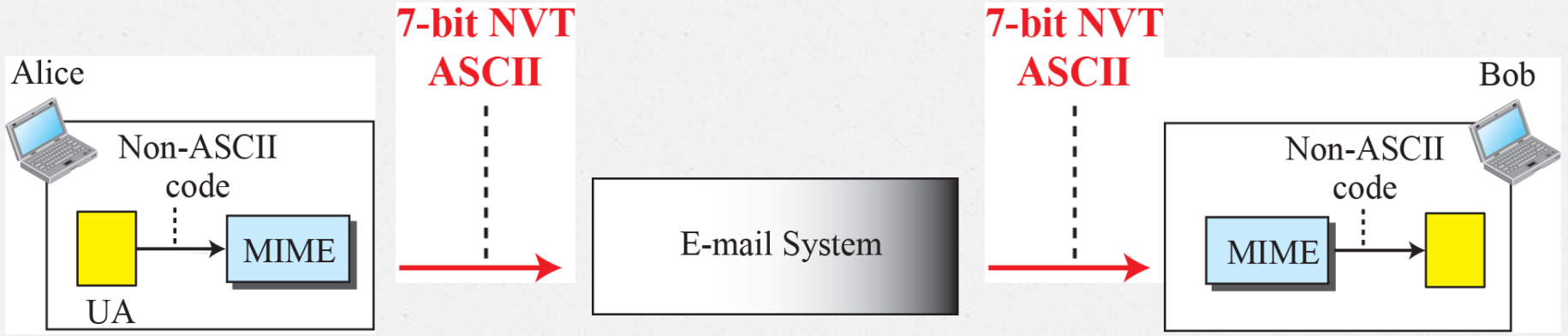
- IMAP4 is more powerful and more complex
- POP3 is deficient in several ways. It does not allow the user to organize her mail on the server; the user cannot have different folders on the server. In addition, POP3 does not allow the user to partially check the contents of the mail before downloading.
- IMAP4 provides the following extra functions:
  - A user can check the e-mail header prior to downloading.
  - A user can search the contents of the e-mail for a specific string of characters prior to downloading.
  - A user can partially download e-mail. This is especially useful if bandwidth is limited and the e-mail contains multimedia with high bandwidth requirements.
  - A user can create, delete, or rename mailboxes on the mail server.
  - A user can create a hierarchy of mailboxes in a folder for e-mail storage.

Multipurpose Internet Mail Extensions (MIME)

- Electronic mail has a simple structure.
  - It can send messages only in NVT 7-bit ASCII format.
  - It cannot be used for languages other than English
  - It cannot be used to send binary files or video or audio data.

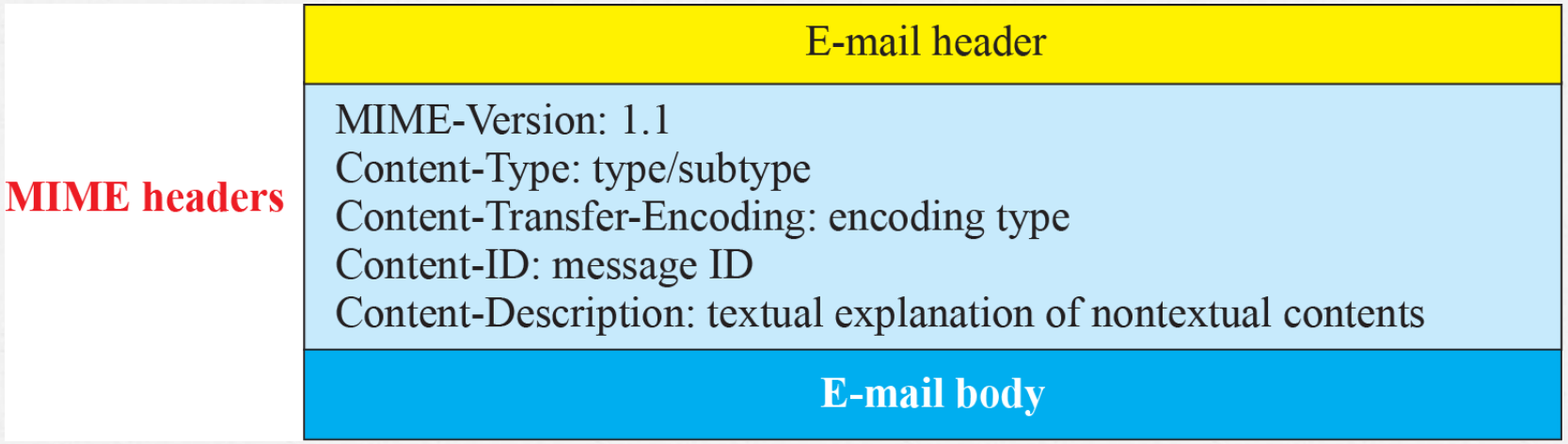
- MIME is a supplementary protocol that allows non-ASCII data to be sent through e-mail.
- MIME transforms non-ASCII data at the sender site to NVT ASCII data and delivers it to the client MTA to be sent through the Internet. The message at the receiving site is transformed back to the original data.

MIME


Alice
Non-ASCII
code
MIME
UA
7-bit NVT
ASCII
E-mail System
7-bit NVT
ASCII
Bob
Non-ASCII
code
MIME

MIME header

MIME defines five headers, which can be added to the original e-mail header section to define the transformation parameters



MIME-Version: This header defines the version of MIME used. The current version is 1.1.

Content-Type: This header defines the type of data used in the body of the message. The content type and the content subtype are separated by a slash.

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)

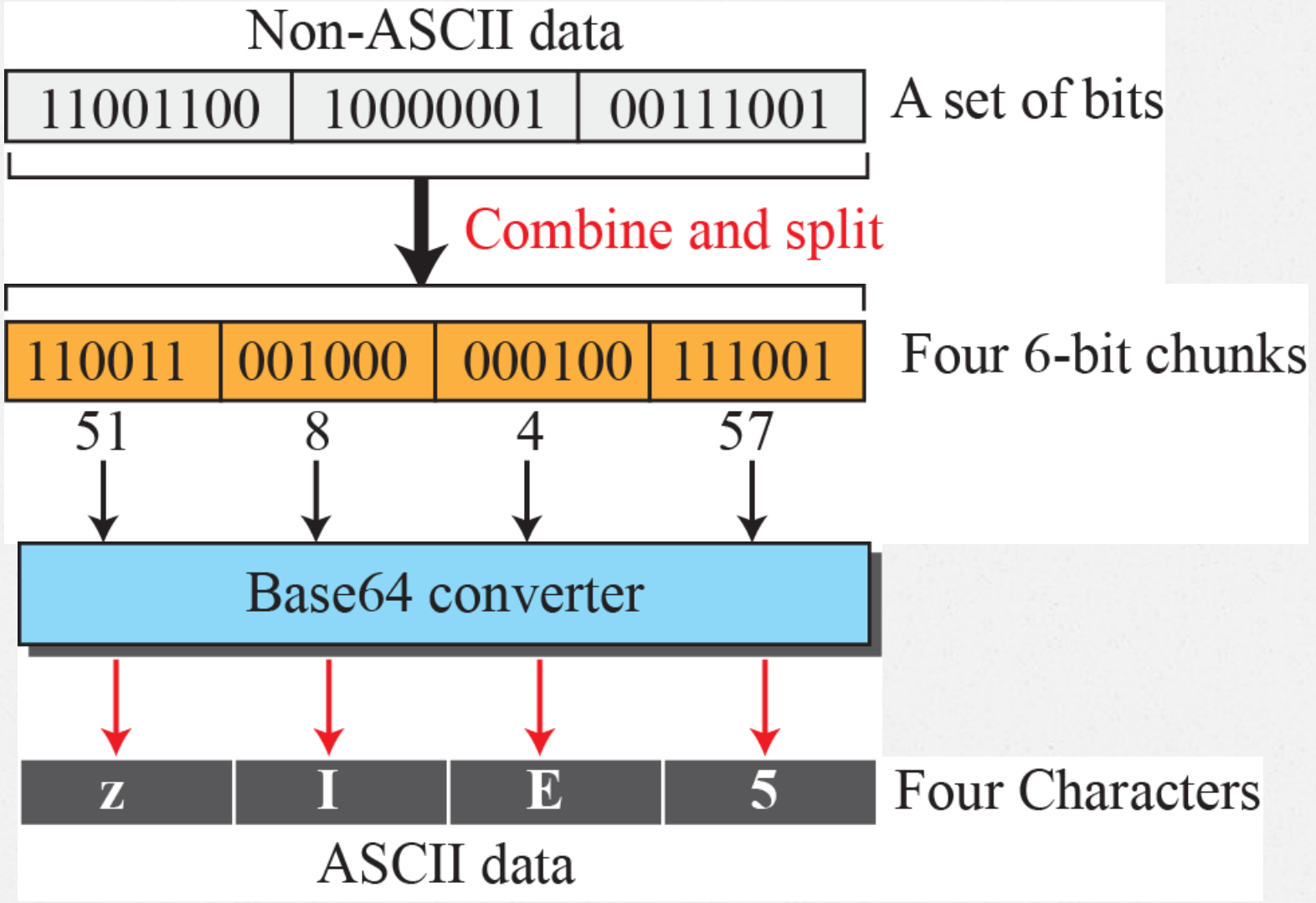
Methods for Content-Transfer-Encoding

Content-Transfer-Encoding: This header defines the method used to encode the messages into 0s and 1s for transport. The five types of encoding methods are listed in Table.

The last two encoding methods are interesting. In the Base64 encoding, data, as a string of bits, is first divided into 6-bit chunks

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

Base64 conversion



Base64 Converting Table

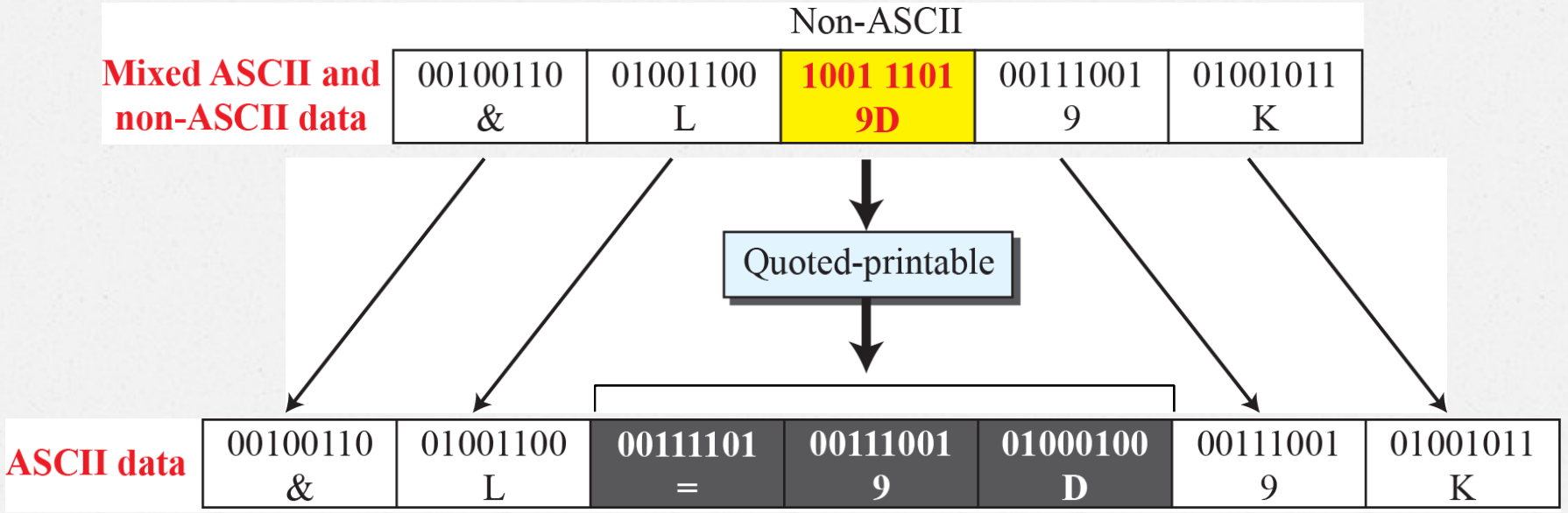
Value	*Code*	*Value*	*Code*	*Value*	*Code*	*Value*	*Code*	*Value*	*Code*	*Value*	*Code*
0	**A**	11	**L**	22	**W**	33	**h**	44	**s**	55	**3**
1	**B**	12	**M**	23	**X**	34	**i**	45	**t**	56	**4**
2	**C**	13	**N**	24	**Y**	35	**j**	46	**u**	57	**5**
3	**D**	14	**O**	25	**Z**	36	**k**	47	**v**	58	**6**
4	**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**		

- Base64 is a redundant encoding scheme; that is, every six bits become one ASCII character and are sent as eight bits. We have an overhead of 25 percent.
- If the data consist mostly of ASCII characters with a small non-ASCII portion, we can use quoted-printable encoding.
- In quoted-printable, if a character is ASCII, it is sent as is.
- If a character is not ASCII, it is sent as three characters. The first character is the equal sign (=). The next two characters are the hexadecimal representations of the byte.

Content-ID: This header uniquely identifies the whole message in a multiple message environment.

Content-Description: This header defines whether the body is image, audio, or video.

Quoted-printable



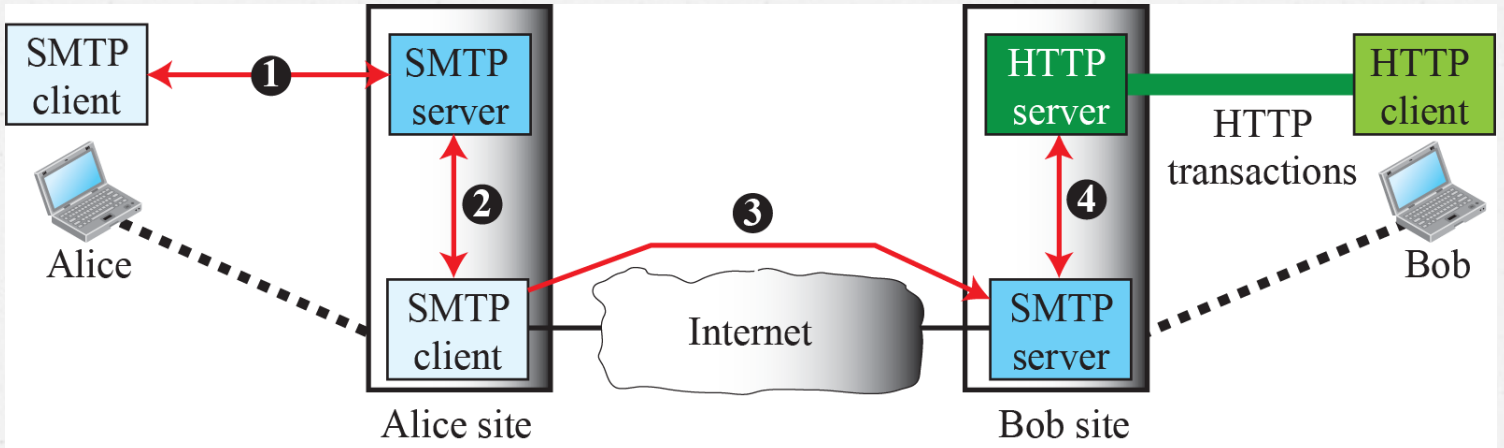
Web-Based Mail

- E-mail is such a common application that some websites today provide this service to anyone who accesses the site. Three common sites are Hotmail, Yahoo, and Google mail.
- The idea is very simple. It has two cases:

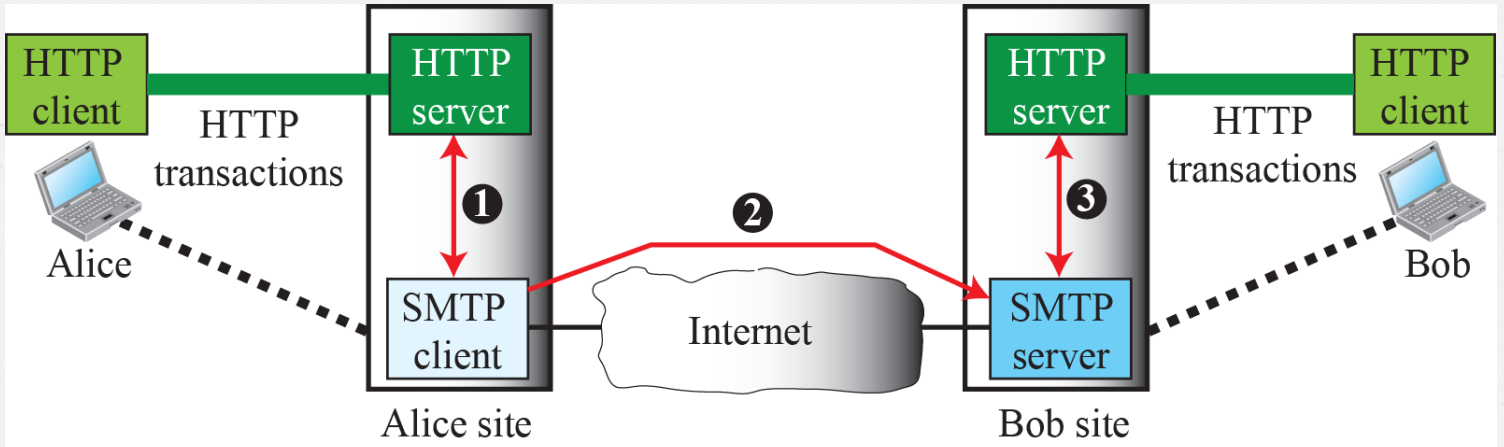
Case-1

Case-2

Web-based e-mail, cases I and II



Case 1: Only receiver uses HTTP



Case 2: Both sender and receiver use HTTP

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)

Chapter 5

Application Layer

- STANDARD CLIENT SERVER APPLICATIONS
  - HTTP
  - FTP
  - SMTP
  - **Telnet**
  - **SSH**
  - DNS

TErminaL NETwork (*TELNET*)

- Local versus Remote Logging
- Network Virtual Terminal (NVT)
- Options & User Interface

TELNET

A server program can provide a specific service to its corresponding client program. However, it is impossible to have a client/server pair for each type of service we need. Another solution is to have a specific client/server program for a set of common scenarios, but to have some generic client/server programs that allow a user on the client site to log into the computer at the server site and use the services available there. We refer to these generic client/server pairs as remote logging applications. One of the original remote logging protocols is TELNET.

- One of the original remote logging protocols is TELNET, which is an abbreviation for **TE**rmina**L NET**work
- Connection Oriented Protocol (Well defined Port Number: 23)
- TELNET requires a logging name and password
- It sends all data including the password in plaintext (not encrypted)
- Simple plaintext architecture of TELNET allows us to explain the issues and challenges related to the concept of remote logging
- Network administrators often use TELNET for diagnostic and debugging purposes

Local logging

When a user logs into a local system, it is called **local login**

- User types at a terminal or at a workstation running a terminal emulator, the keystrokes are accepted by the terminal driver.
- The terminal driver passes the characters to the operating system.
- The operating system, in turn, interprets the combination of characters and invokes the desired application program or utility.

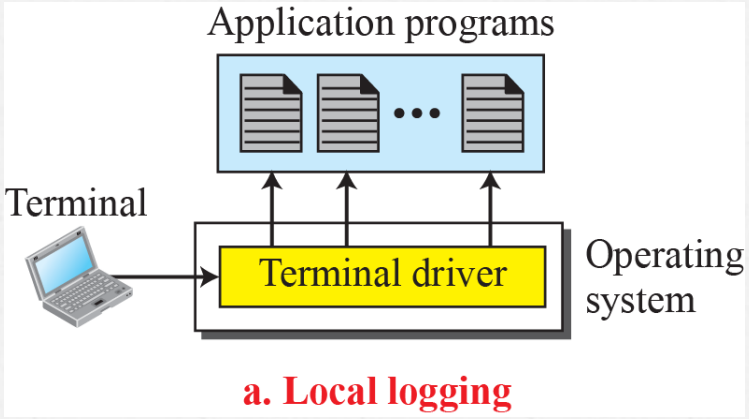
Remote logging

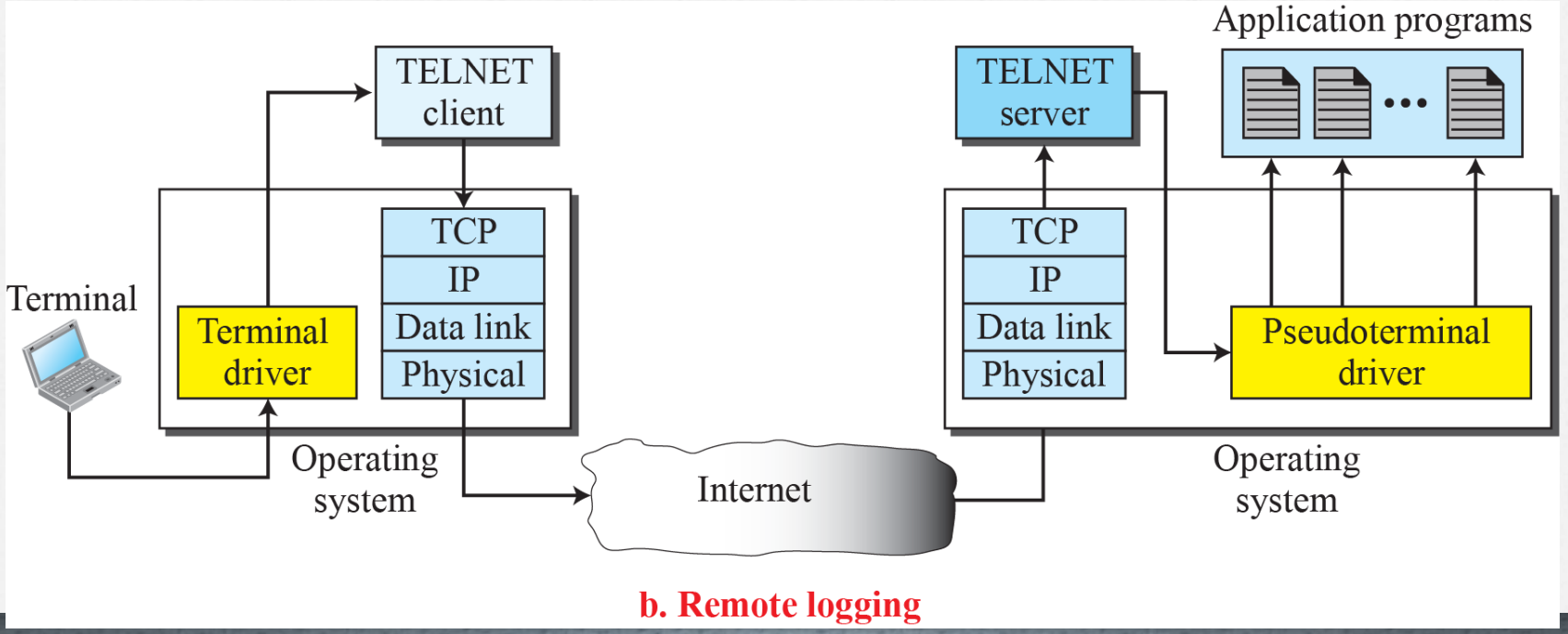
When a user wants to access an application program or utility located on a remote machine, she performs **remote logging**

Here the TELNET client and server programs come into use.

- The user sends the keystrokes to the terminal driver where the local operating system accepts the characters but does not interpret them.
- The characters are sent to the TELNET client, which transforms the characters into a universal character set called Network Virtual Terminal (NVT) characters (discussed below) and delivers them to the local TCP/IP stack.
- The commands or text, in NVT form, travel through the Internet and arrive at the TCP/IP stack at the remote machine.
- Here the characters are delivered to the operating system and passed to the TELNET server.
- However, the characters cannot be passed directly to the operating system because the remote operating system is not designed to receive characters from a TELNET server; it is designed to receive characters from a terminal driver
- The solution is to add a piece of software called a pseudoterminal driver, which pretends that the characters are coming from a terminal.
- The operating system then passes the characters to the appropriate application program

Local versus remote logging



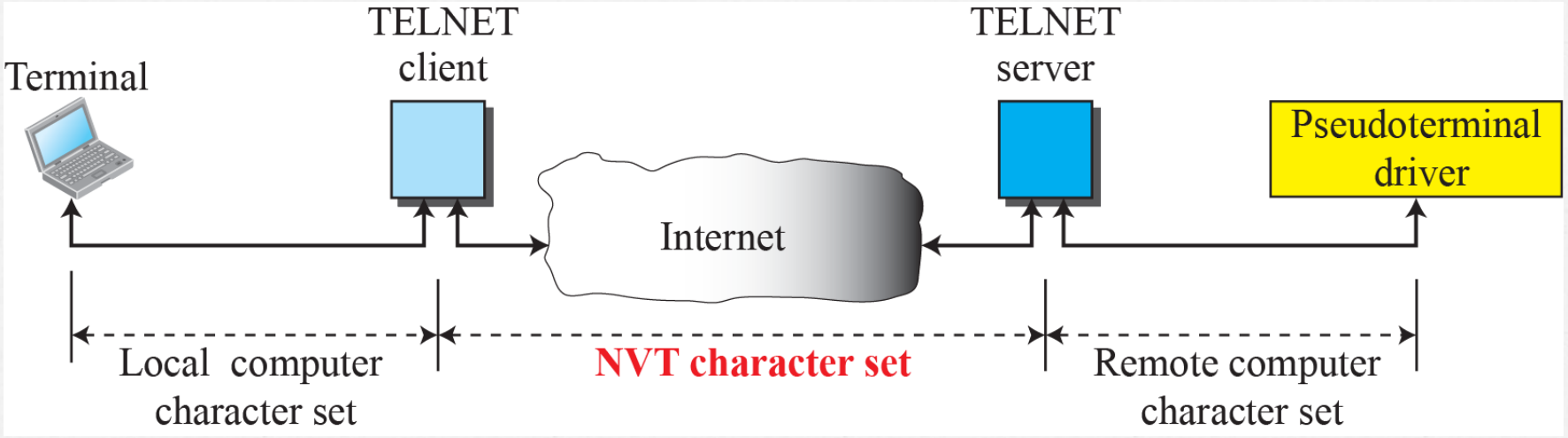


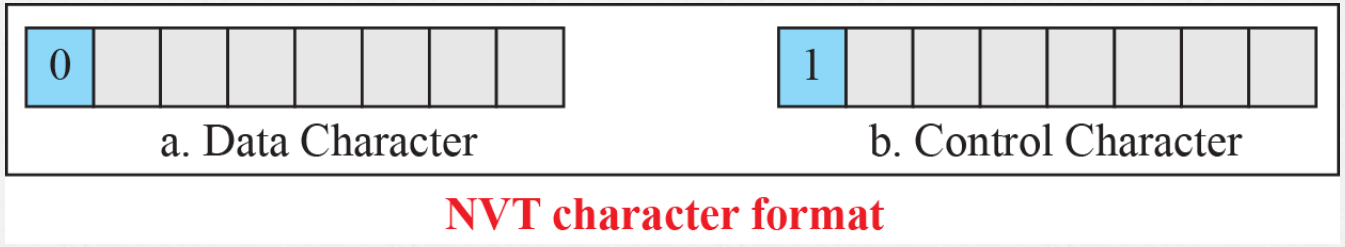
Network Virtual Terminal (NVT)

- In a heterogeneous system, the mechanism to access a remote computer is complex.
- This is because every computer and its operating system accepts a special combination of characters as tokens.
- If we want to access any remote computer in the world, we must first know what type of computer we will be connected to, and we must also install the specific terminal emulator used by that computer.
- TELNET solves this problem by defining a universal interface called the **Network Virtual Terminal (NVT)** character set
- The client TELNET translates characters (data or commands) that come from the local terminal into NVT form and delivers them to the network.
- The server TELNET, on the other hand, translates data and commands from NVT form into the form acceptable by the remote computer

Concept of NVT

NVT uses two sets of characters, one for data and one for control
Both are 8-bit bytes.(NVT ASCII)





NVT character format

Options & User interface commands

TELNET lets the client and server negotiate options before or during the use of the service. Options are extra features available to a user with a more sophisticated terminal.

The operating system (UNIX, for example) defines an interface with user-friendly commands.

Command	*Meaning*	*Command*	*Meaning*
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)

- Components
  - SSH Transport-Layer Protocol (SSH-TRANS)
  - SSH Authentication Protocol (SSH-AUTH)
  - SSH Connection Protocol (SSH-CONN)
- Applications
  - SSH for Remote Logging
  - SSH for File Transfer
- Port Forwarding
- Format of the SSH Packets

Secure Shell (SSH)

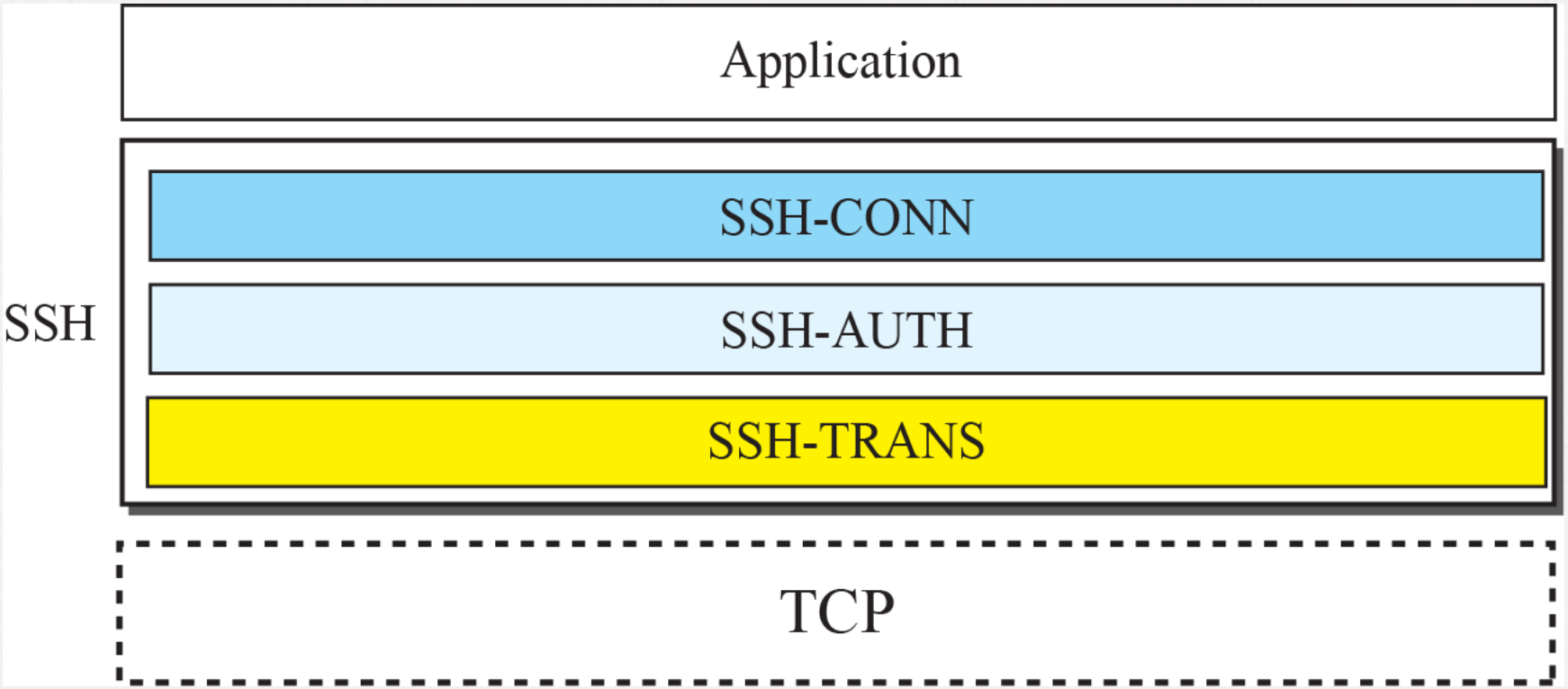
Although Secure Shell (SSH) is a secure application program that can be used today for several purposes such as remote logging and file transfer, it was originally designed to replace TELNET.

There are two versions of SSH: SSH-1 and SSH-2, which are totally incompatible. The first version, SSH-1, is now deprecated because of security flaws in it.

In this section, we discuss only SSH-2.

Connection Oriented protocol (Well Known Port No. : 22)

Components of SSH



SSH Transport-Layer Protocol (SSH-TRANS)
SSH Authentication Protocol (SSH-AUTH)
SSH Connection Protocol (SSH-CONN)

SSH Transport-Layer Protocol (SSH-TRANS)

- TCP is not a secured transport-layer protocol, SSH first uses a protocol that creates a secured channel on top of the TCP
- This new layer is an independent protocol referred to as **SSH-TRANS**
- When the procedure implementing this protocol is called, the client and server first use the TCP protocol to establish an insecure connection.
- Then they exchange several security parameters to establish a secure channel on top of the TCP
- Services Provided by SSH-Trans are:
  1. Privacy or confidentiality of the message exchanged
  2. Data integrity, which means that it is guaranteed that the messages exchanged between the client and server are not changed by an intruder
  3. Server authentication, which means that the client is now sure that the server is the one that it claims to be
  4. Compression of the messages, which improves the efficiency of the system and makes attack more difficult

SSH Authentication Protocol (SSH-AUTH)

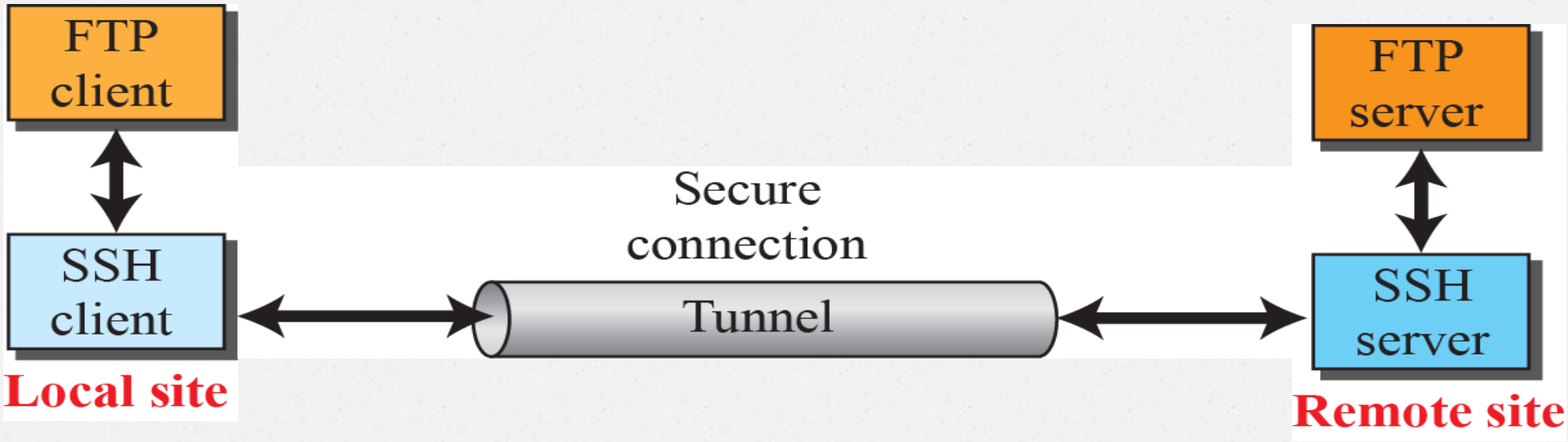
- After a secure channel is established between the client and the server and the server is authenticated for the client, SSH can call another procedure that can authenticate the client for the server.
- The client authentication process in SSH is very similar to what is done in Secure Socket Layer (SSL)
- Authentication starts with the client, which sends a request message to the server.
- The request includes the user name, server name, the method of authentication, and the required data.
- The server responds with either a success message, which confirms
- that the client is authenticated, or a failed message, which means that the process needs to be repeated with a new request message.

SSH Connection Protocol (SSH-CONN)

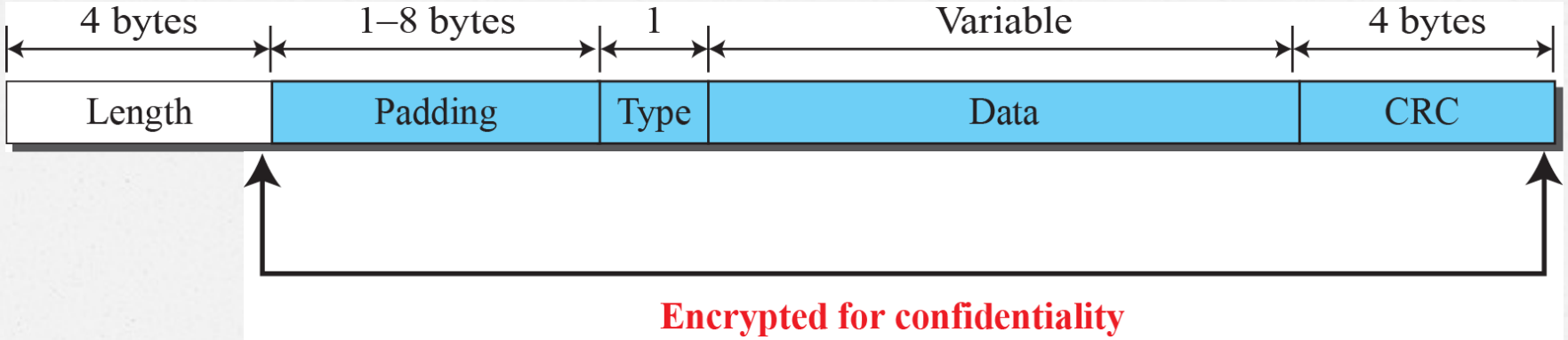
- After the secured channel is established and both server and client are authenticated for each other, SSH can call a piece of software that implements the third protocol, SSHCONN.
- One of the services provided by the SSH-CONN protocol is multiplexing.
- SSH-CONN takes the secure channel established by the two previous protocols and lets the client create multiple logical channels over it.
- Each channel can be used for a different purpose, such as remote logging, file transfer, and so on.

Port Forwarding

We can use the secured channels available in SSH to access an application program that does not provide security services.
Applications such as TELNET and Simple Mail Transfer Protocol (SMTP), which are discussed, can use the services of the SSH port forwarding mechanism. The SSH port forwarding mechanism creates a tunnel through which the messages belonging to other protocols can travel. For this reason, this mechanism is sometimes referred to as **SSH tunneling**.



SSH Packet Format



- The **length field** defines the length of the packet but does not include the padding.
- One to eight bytes of **padding** is added to the packet to make the attack on the security provision more difficult.
- The **cyclic redundancy check (CRC) field** is used for error detection.
- The **type field** designates the type of the packet used in different SSH protocols.
- The **data field** is the data transferred by the packet in different protocols.

SSH Applications

SSH for Remote Logging

Several free and commercial applications use SSH for remote logging. Among them, we can mention PuTTy, by Simon Tatham, which is a client SSH program that can be used for remote logging. Another application program is Tectia, which can be used on several platforms.

SSH for File Transfer

One of the application programs that is built on top of SSH for file transfer is the Secure File Transfer Program (sftp). The sftp application program uses one of the channels provided by the SSH to transfer files. Another common application is called Secure Copy (scp). This application uses the same format as the UNIX copy command, cp, to copy files.

Chapter 5

Application Layer

- STANDARD CLIENT SERVER APPLICATIONS
  - HTTP
  - FTP
  - SMTP
  - Telnet
  - SSH
  - DNS

Domain Name System (DNS)

- Name Space
  - Domain Name Space
  - Domain
  - Distribution of Name Space
  - Zone
  - Root Server
- DNS in the Internet
  - Generic Domains
  - Country Domains

- Resolution
  - Recursive Resolution
  - Iterative Resolution
  - Caching
- Resource Records
- DNS Messages
- Encapsulation
- Registrars
- DDNS
- Security of DNS

Domain Name System (DNS)

To identify an entity, TCP/IP protocols use the IP address, which uniquely identifies the connection of a host to the Internet. However, people prefer to use names instead of numeric addresses. Therefore, the Internet needs to have a directory system that can map a name to an address. This is analogous to the telephone network. A telephone network is designed to use telephone numbers, not names. People can either keep a private file to map a name to the corresponding telephone number or can call the telephone directory to do so.

Since the Internet is so huge today, a central directory system cannot hold all the mapping. In addition, if the central computer fails, the whole communication network will collapse.

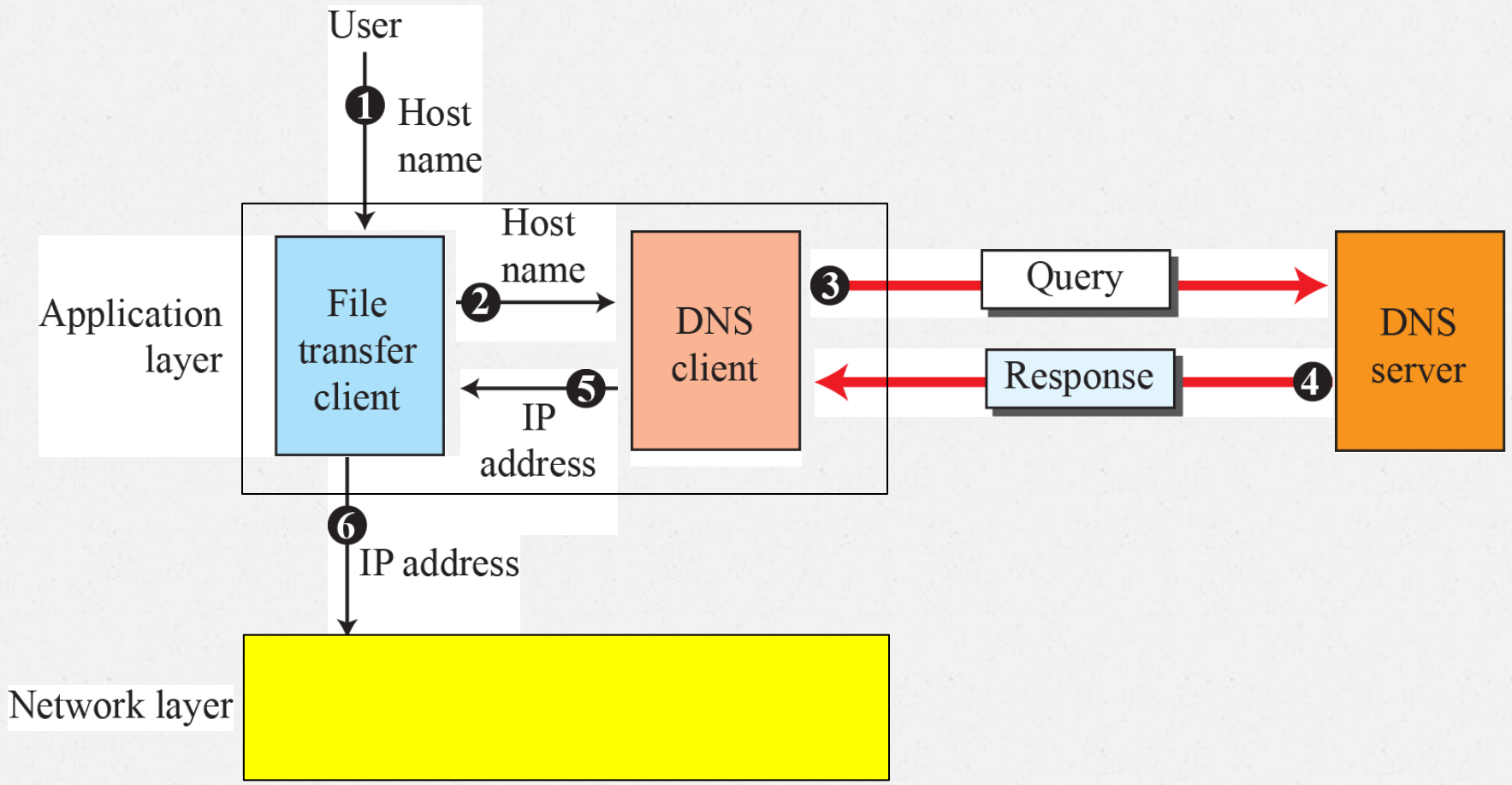
A better solution is to distribute the information among many computers in the world. In this method, the host that needs mapping can contact the closest computer holding the needed information. This method is used by the **Domain Name System (DNS).**

TCP/IP uses a DNS client and a DNS server to map a name to an address

A user wants to use a file transfer client to access the corresponding file transfer server running on a remote host. The user knows only the file transfer server name, such as **afilesource.com**. However, the TCP/IP suite needs the IP address of the file transfer server to make the connection. The following six steps map the host name to an IP address:

1. The user passes the host name to the file transfer client.
2. The file transfer client passes the host name to the DNS client.
3. Each computer, after being booted, knows the address of one DNS server. The DNS client sends a message to a DNS server with a query that gives the file transfer server name using the known IP address of the DNS server.
4. The DNS server responds with the IP address of the desired file transfer server.
5. The DNS client passes the IP address to the file transfer server.
6. The file transfer client now uses the received IP address to access the file transfer server.

Purpose of DNS



Name Space

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. 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.

In a **flat name space**, a name is assigned to an address. A name in this space is a sequence of characters without structure. The main disadvantage of a flat name space is that it cannot be used in a large system such as the Internet because it must be centrally controlled to avoid ambiguity and duplication.

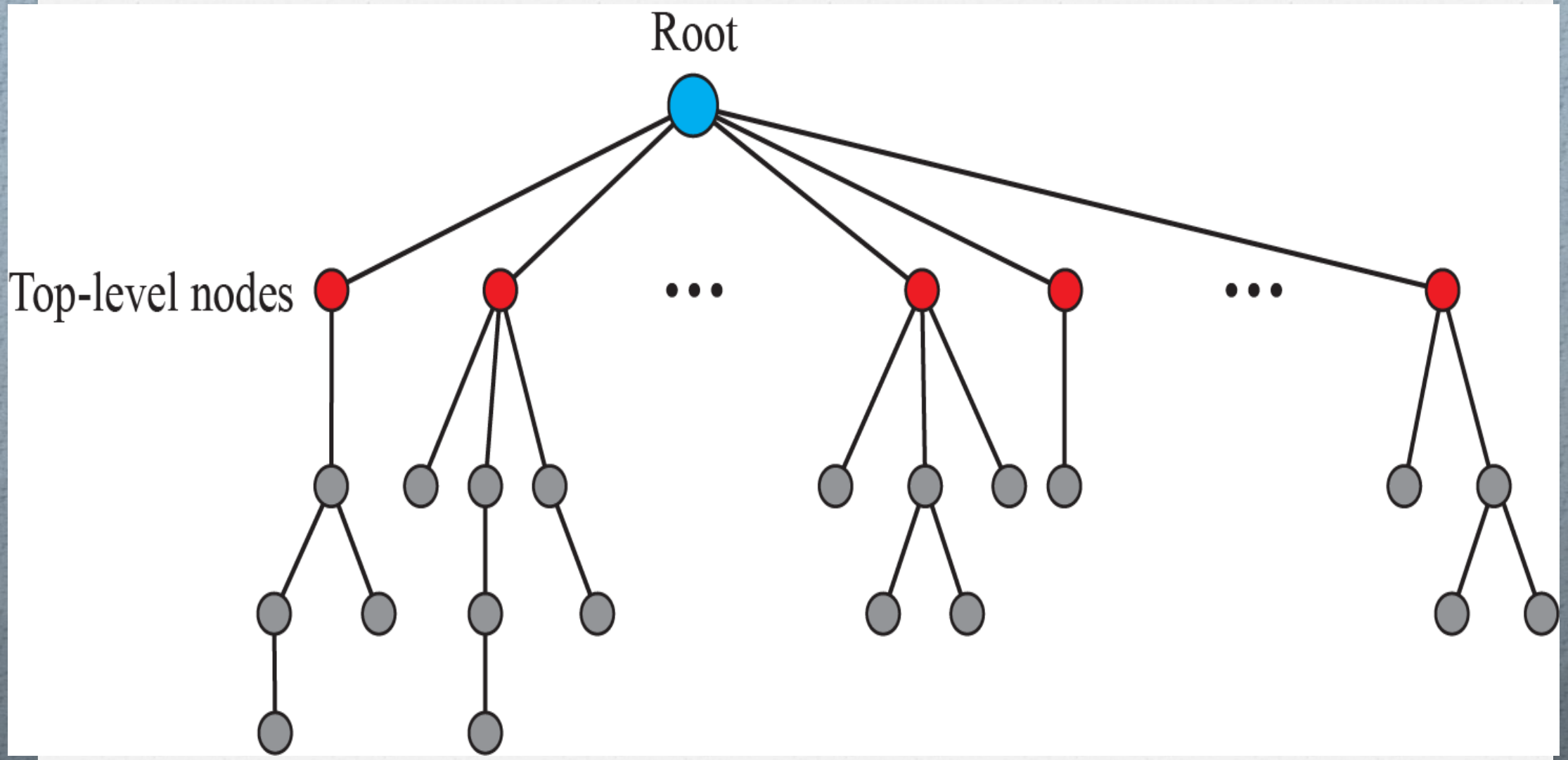
Hierarchical Name Space

Each name is made of several parts. The first part can define the nature of the organization, the second part can define the name of an organization, the third part can define departments in the organization, and so on. In this case, the authority to assign and control the name spaces can be decentralized.

Domain Name Space

To have a hierarchical name space, a domain name space was designed. In this design the names are defined in an inverted-tree structure with the root at the top. The tree can have only 128 levels: level 0 (root) to level 127.

Domain name space



Label

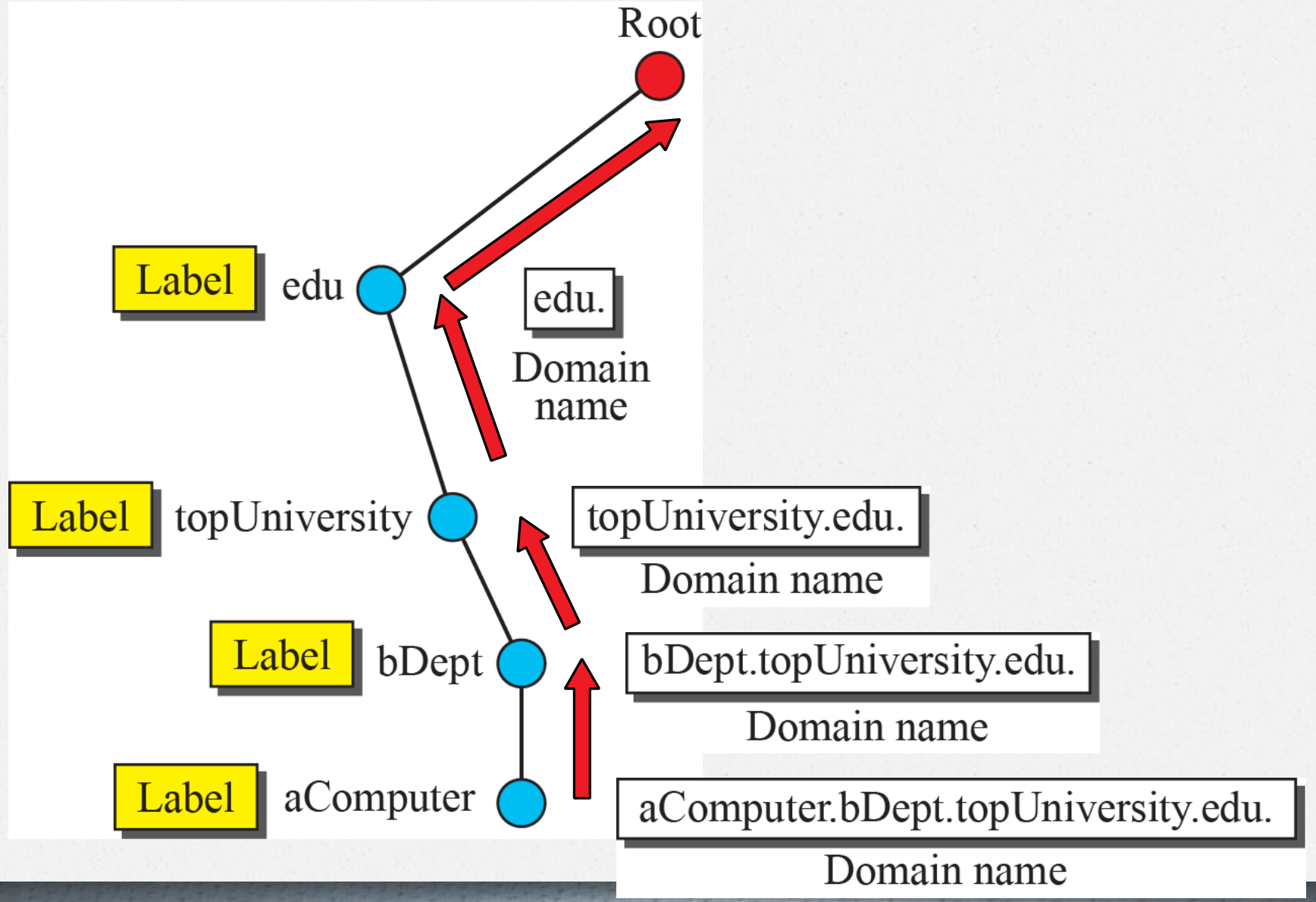
Each node in the tree has a label, which is a string with a maximum of 63 characters. The root label is a null string (empty string). DNS requires that children of a node (nodes that branch from the same node) have different labels, which guarantees the uniqueness of the domain names.

Domain Name

Each node in the tree has a domain name. A full domain name is a sequence of labels separated by dots (.). The domain names are always read from the node up to the root. The last label is the label of the root (null). This means that a full domain name always ends in a null label, which means the last character is a dot because the null string is nothing.

If a label is terminated by a null string, it is called a **fully qualified domain name (FQDN)**. The name must end with a null label, but because null means nothing, the label ends with a dot. If a label is not terminated by a null string, it is called a **partially qualified domain name (PQDN)**. A PQDN starts from a node, but it does not reach the root. It is used when the name to be resolved belongs to the same site as the client. Here the resolver can supply the missing part, called the suffix, to create an FQDN.

Domain names and labels



Domain

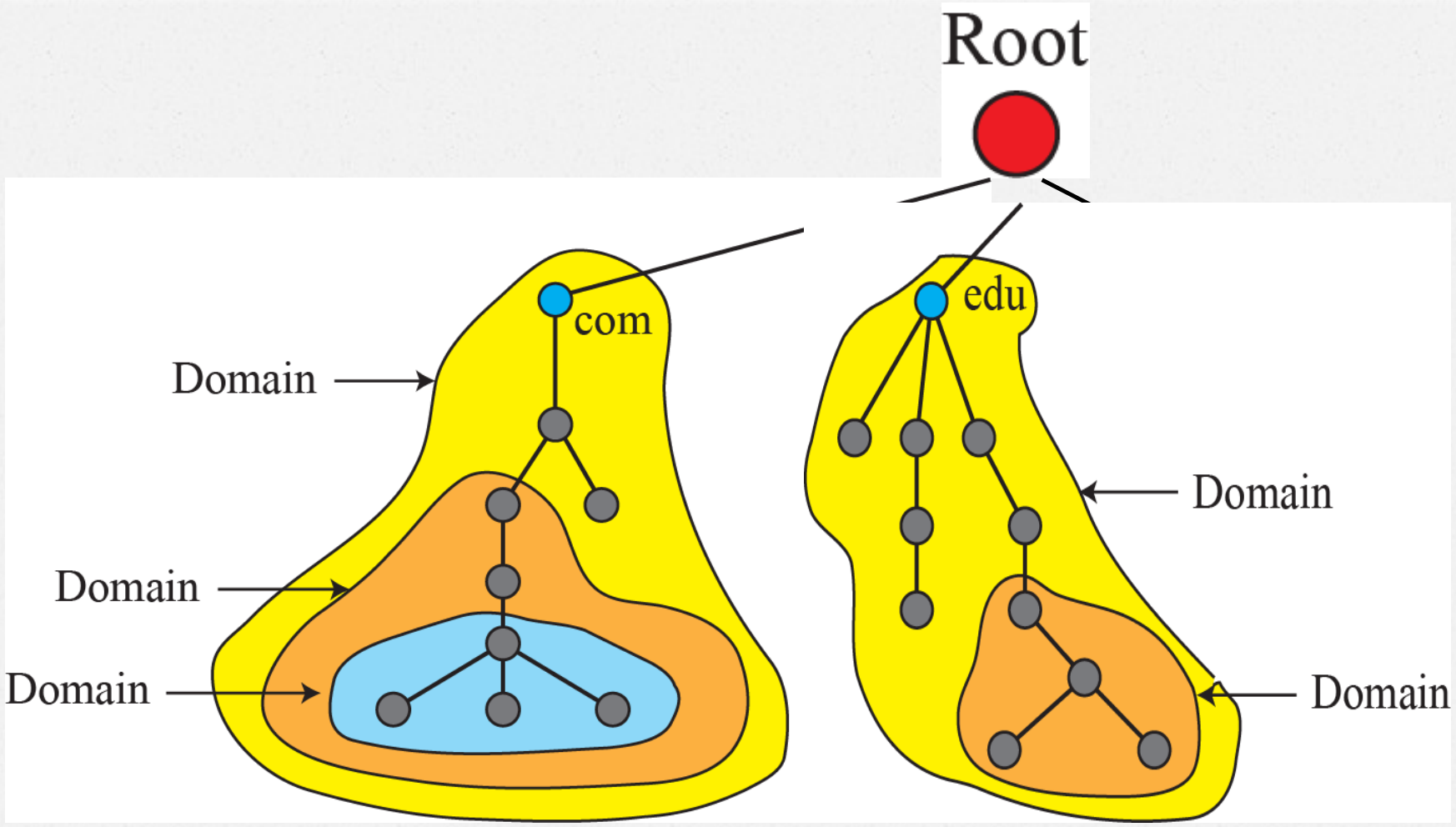
A domain is a subtree of the domain name space. The name of the domain is the name of the node at the top of the subtree.

A domain may itself be divided into domains.

Distribution of Name Space

The information contained in the domain name space must be stored. However, it is very inefficient and also not reliable to have just one computer store such a huge amount of information. It is inefficient because responding to requests from all over the world places a heavy load on the system. It is not reliable because any failure makes the data inaccessible.

Domains

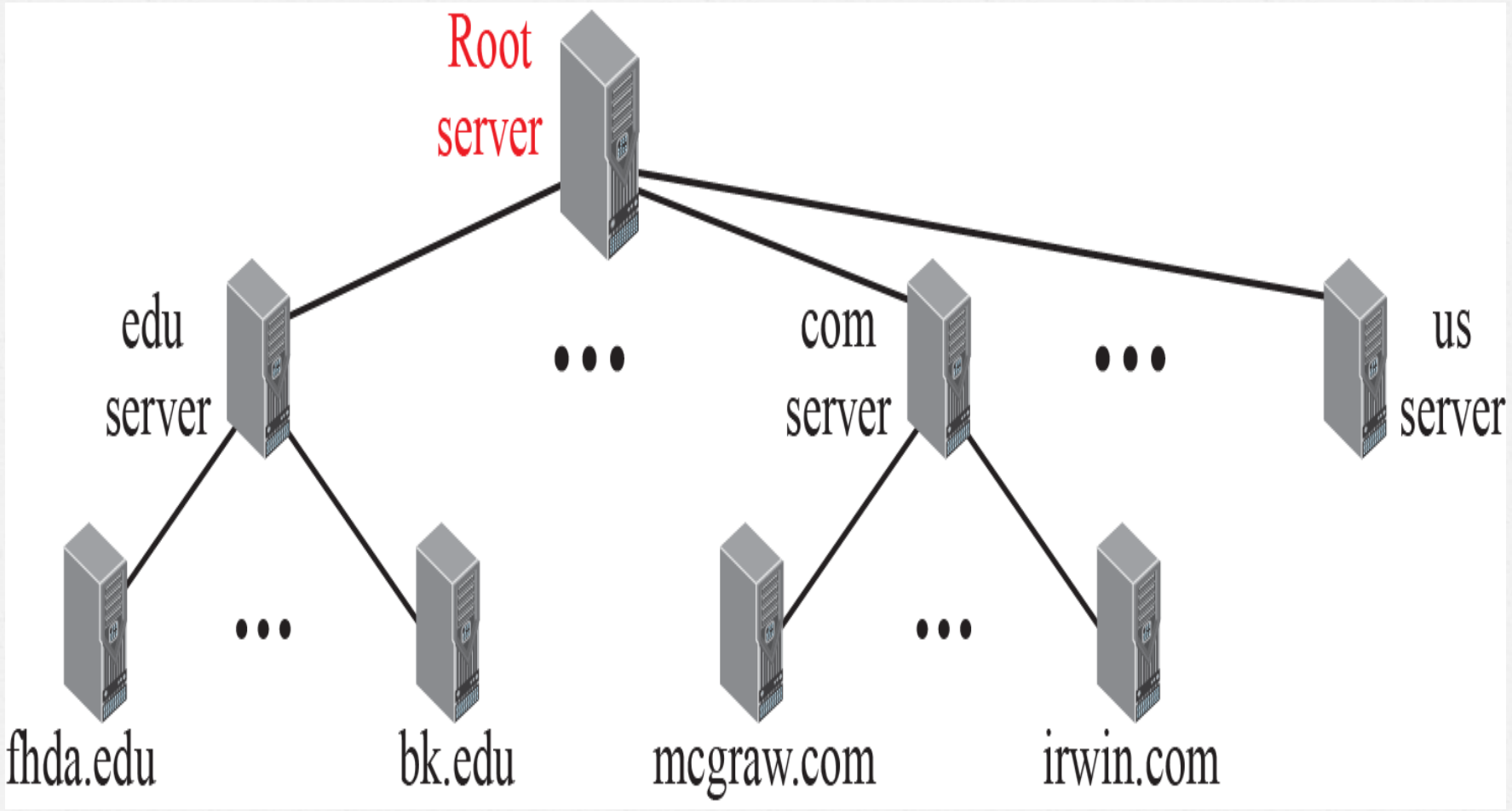


Hierarchy of Name Servers

One way to do this is to divide the whole space into many domains among many computers called DNS servers. One way to do this is to divide the whole space into many domains based on the first level. In other words, we let the root stand alone and create as many domains (subtrees) as there are first-level nodes.

Because a domain created this way could be very large, DNS allows domains to be divided further into smaller domains (subdomains). Each server can be responsible (authoritative) for either a large or small domain.

Hierarchy of name servers



Zone

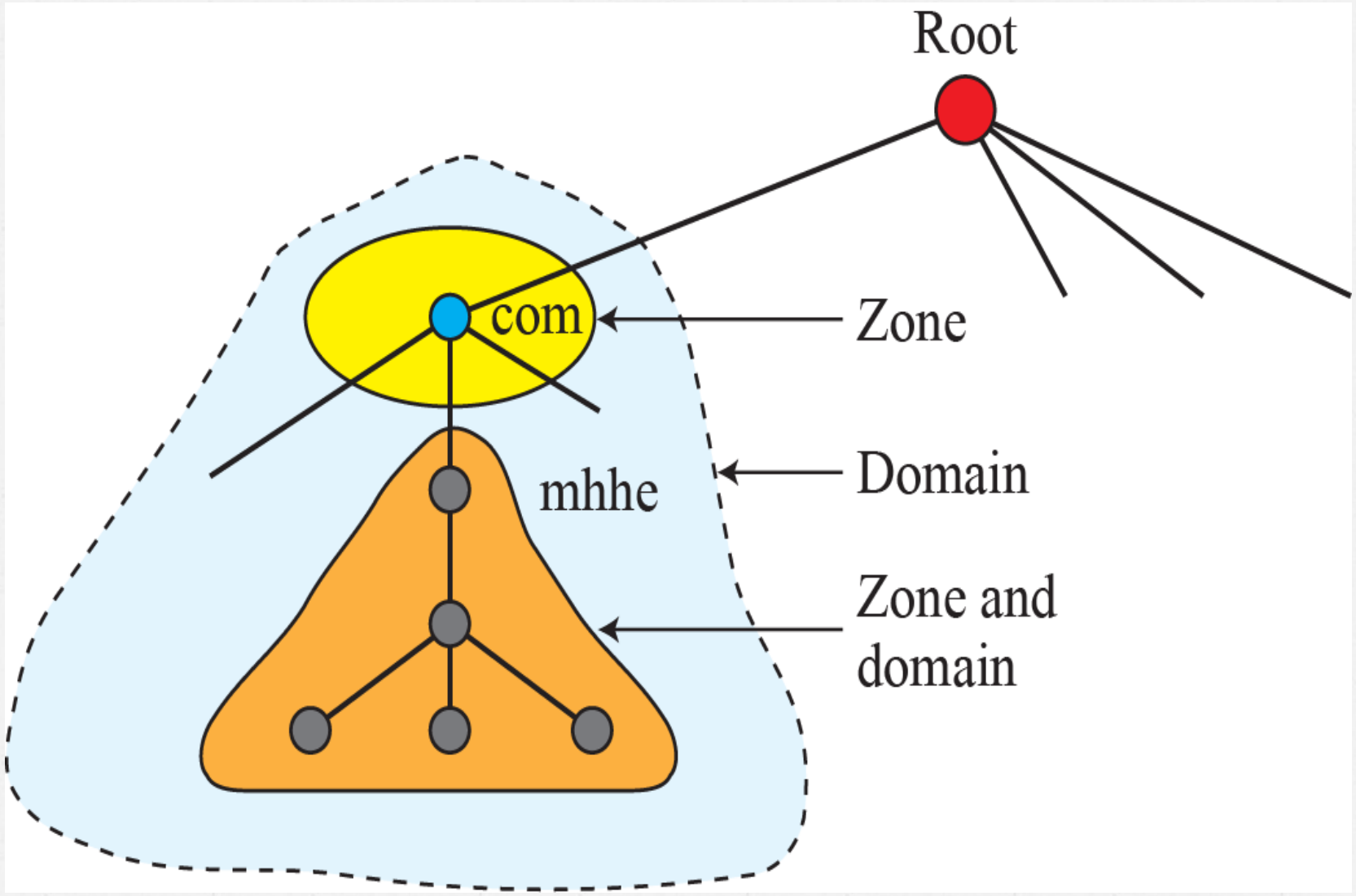
Since the complete domain name hierarchy cannot be stored on a single server, it is divided among many servers. What a server is responsible for or has authority over is called a zone. We can define a zone as a contiguous part of the entire tree.

The server makes a database called a zone file and keeps all the information for every node under that domain.

Root Server

A root server is a server whose zone consists of the whole tree. A root server usually does not store any information about domains but delegates its authority to other servers, keeping references to those servers. There are several root servers, each covering the whole domain name space.

Zone



Primary and Secondary Servers

DNS defines two types of servers: primary and secondary.

- **A primary server** is a server that stores a file about the zone for which it is an authority. It is responsible for creating, maintaining, and updating the zone file. It stores the zone file on a local disk.

- **A secondary server** is a server that transfers the complete information about a zone from another server (primary or secondary) and stores the file on its local disk. The secondary server neither creates nor updates the zone files. If updating is required, it must be done by the primary server, which sends the updated version to the secondary.

A primary server loads all information from the disk file; the secondary server loads all information from the primary server.

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 domain.

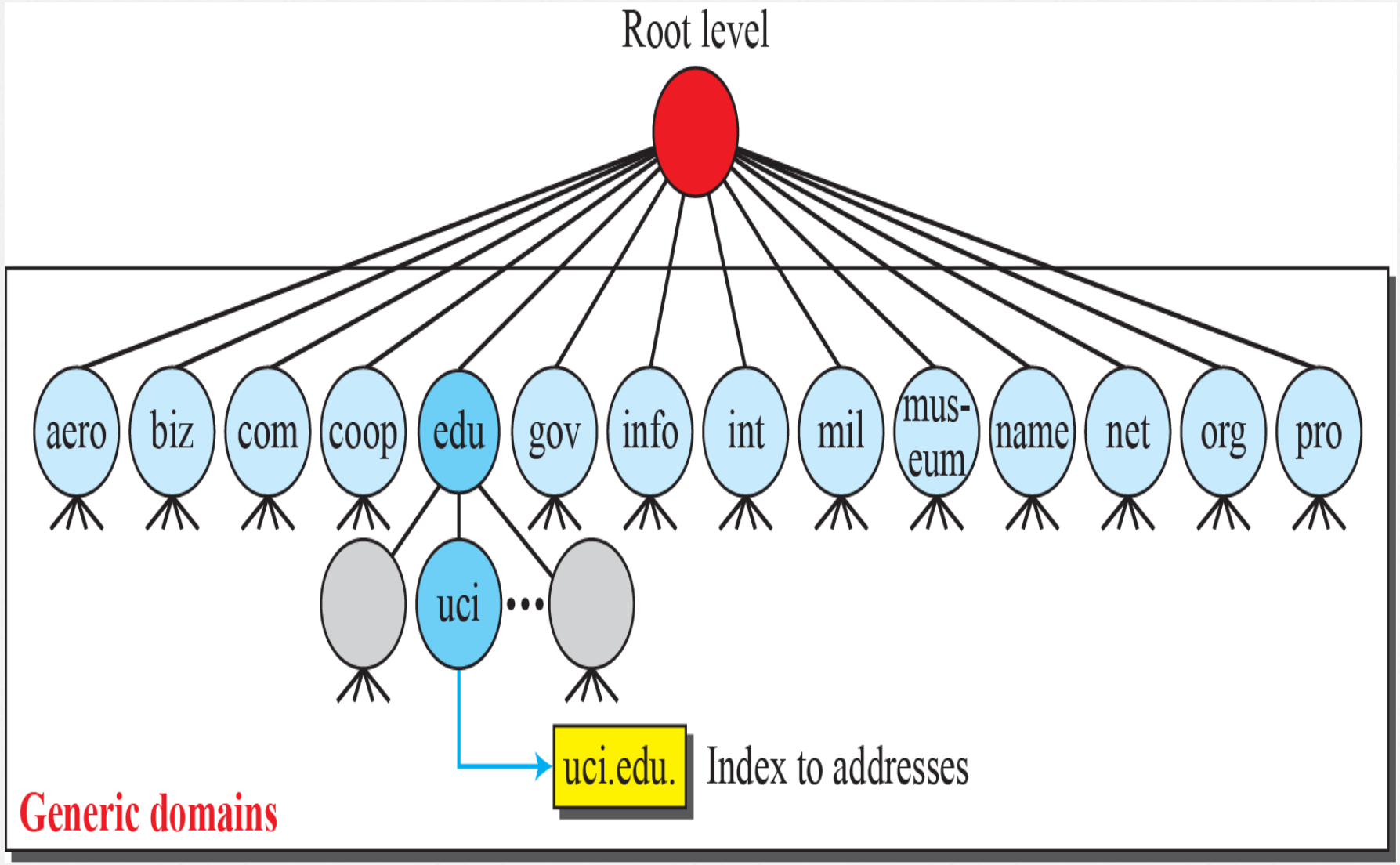
Generic Domains

The generic domains define registered hosts according to their generic behavior. Each node in the tree defines a domain, which is an index to the domain name space database.

Country Domains

The country domains section uses two-character country abbreviations (e.g., us for United States). Second labels can be organizational, or they can be more specific, national designations. The United States, for example, uses state abbreviations as a subdivision of us (e.g., ca.us.).

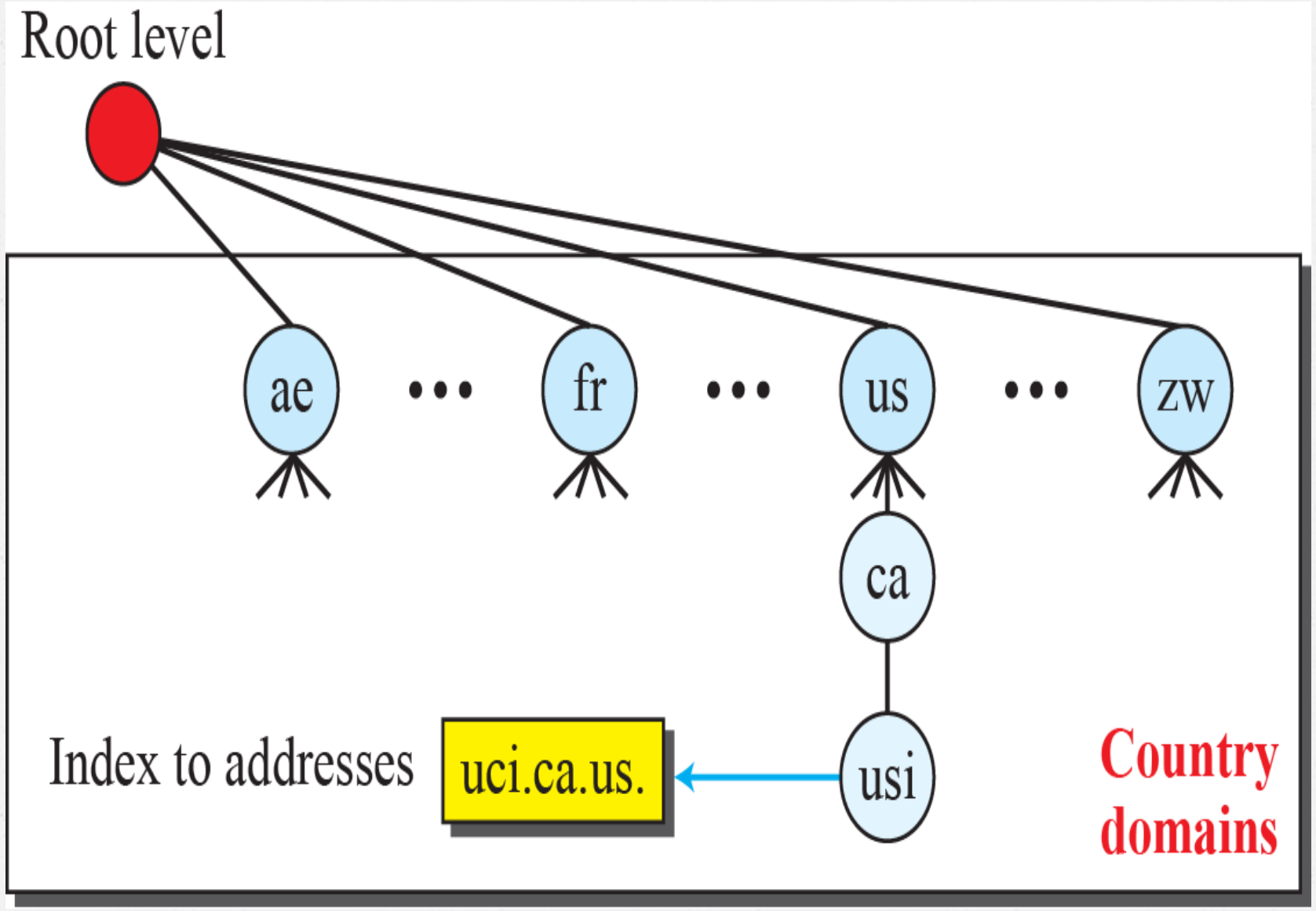
Generic domains



Generic domain labels

Label	*Description*	*Label*	*Description*
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

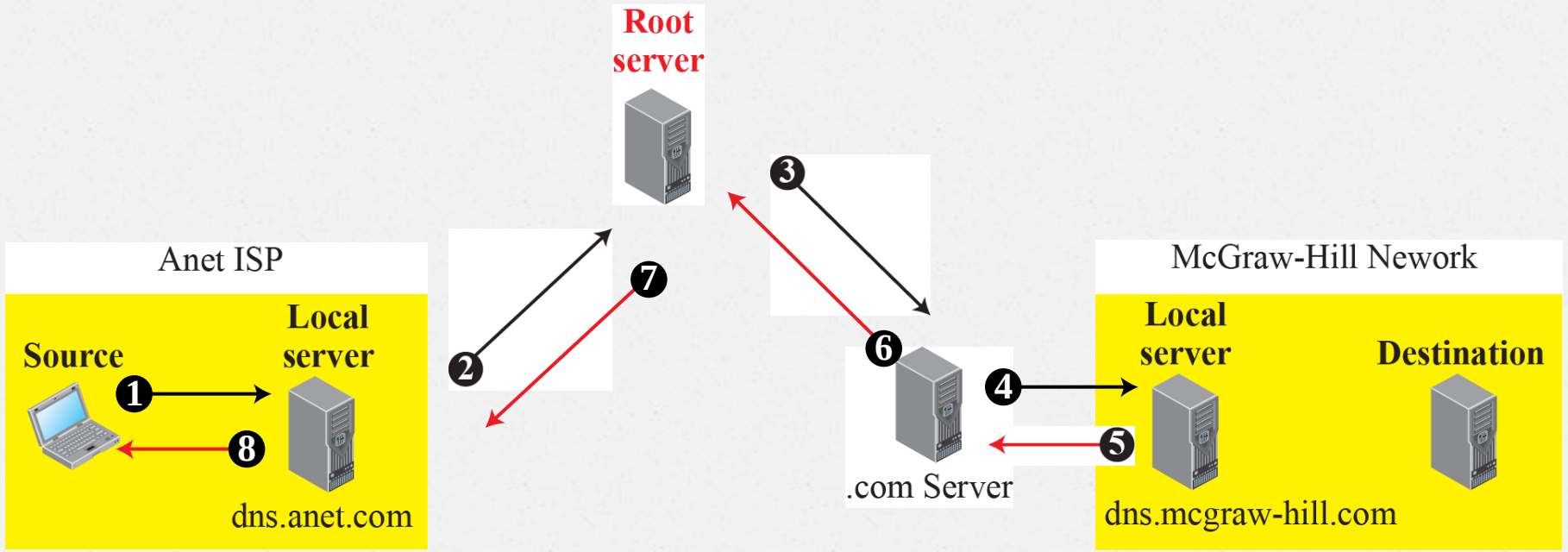
Country domains



Resolution

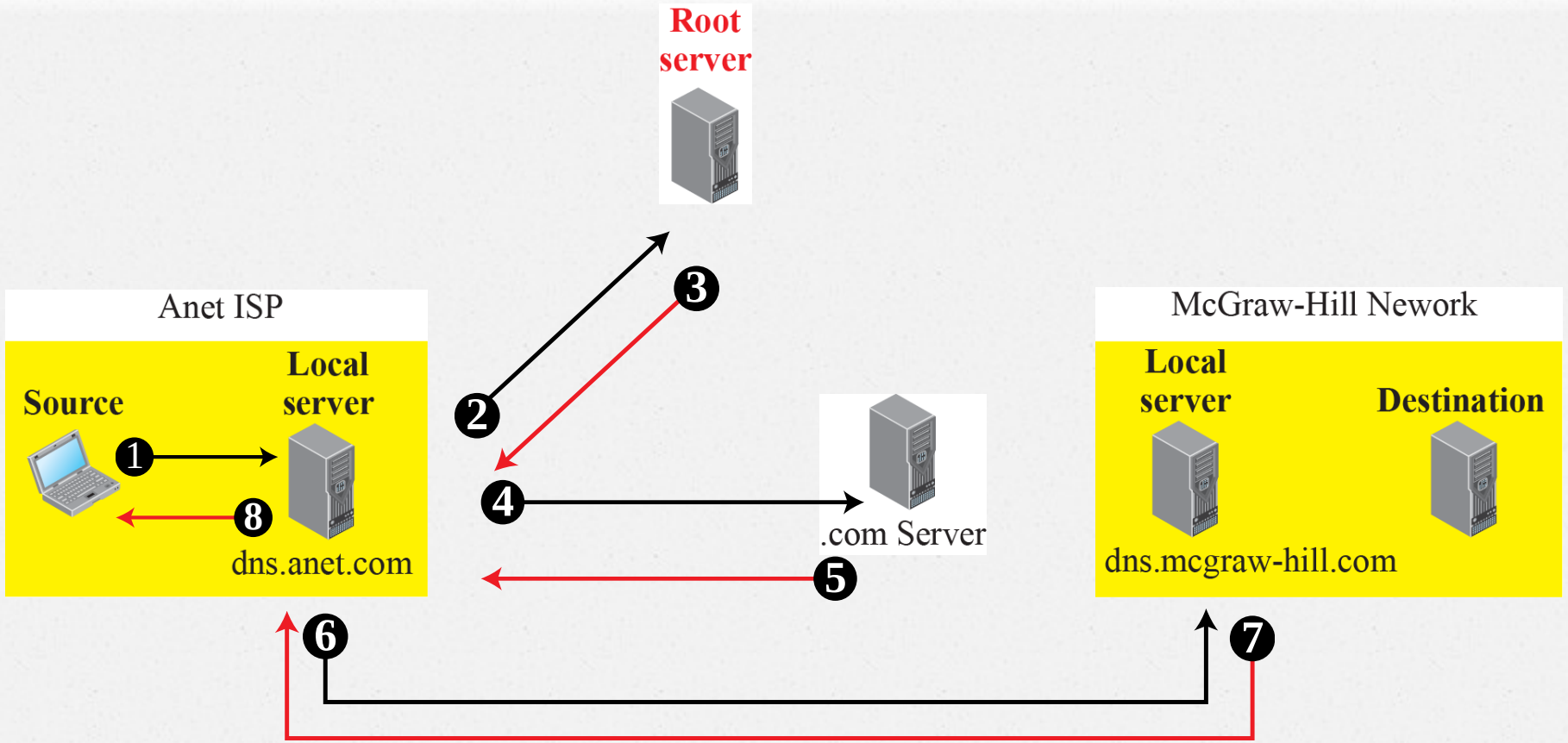
- Mapping a name to an address is called **name-address resolution**. DNS is designed as a client-server application.
- A host that needs to map an address to a name or a name to an address calls a DNS client called a resolver. The resolver accesses the closest DNS server with a mapping request.
- If 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.
- After the resolver receives the mapping, it interprets the response to see if it is a real resolution or an error, and finally delivers the result to the process that requested it.
- A resolution can be either **recursive** or **iterative**.

Recursive resolution



Source: some.anet.com
Destination: engineering.mcgraw-hill.com

Iterative resolution



Source: some.anet.com
Destination: engineering.mcgraw-hill.com

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**.

When a server asks for a mapping from another server and receives the response, it stores this information in its cache memory before sending it to the client.

Caching speeds up resolution, but it can also be problematic. If a server caches a mapping for a long time, it may send an outdated mapping to the client To counter this, two techniques are used.

- **Time To Live (TTL).** It defines the time in seconds that the receiving server can cache the information. After that time, the mapping is invalid and any query must be sent again to the authoritative server.
- DNS requires that each server keep a **TTL counter** for each mapping it caches. The cache memory must be searched periodically and those mappings with an expired TTL must be purged

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)

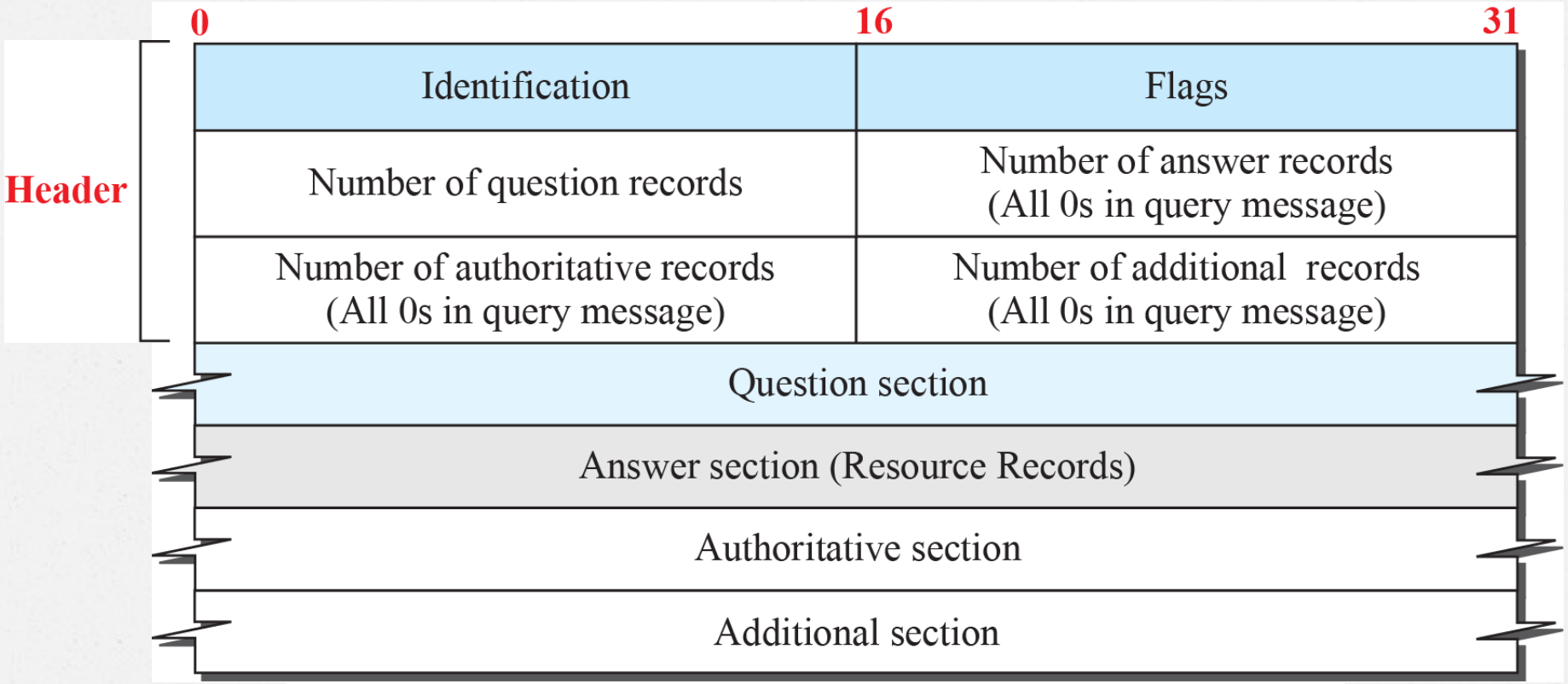
- The domain name field is what identifies the resource record.
- The value defines the information kept about the domain name.
- The TTL defines the number of seconds for which the information is valid.
- The class defines the type of network class address
- The type defines how the value should be interpreted.

DNS types

Type	*Interpretation of value*
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)

DNS message

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



Note:

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

DNS message

- The **identification field** is used by the client to match the response with the query.
- The **flag field** defines whether the message is a query or response. It also includes status of error.
- The next four fields in the header define the number of each record type in the message.
- The **question section**, which is included in the query and repeated in the response message, consists of one or more question records. It is present in both query and response messages.
- The **answer section** consist of one or more resource records. It is present only in response messages.
- The **authoritative section** gives information (domain name) about one or more authoritative servers for the query.
- The **additional information section** provides additional information that may help the resolver

Encapsulation

- DNS can use either UDP or TCP. In both cases the well-known port used by the server is **port 53**.
- UDP is used when the size of the response message is less than 512 bytes because most UDP packages have a 512-byte packet size limit.
- If the size of the response message is more than 512 bytes, a TCP connection is used. In that case, one of two scenarios can occur:
  - If the resolver has prior knowledge that the size of the response message is more than 512 bytes, it uses the TCP connection. For example, if a secondary name server (acting as a client) needs a zone transfer from a primary server.
  - If the resolver does not know the size of the response message, it can use the UDP port. However, if the size of the response message is more than 512 bytes, the server truncates the message and turns on the TC bit. The resolver now opens a TCP connection and repeats the request to get a full response from the server.

Registrars

- New domains added to DNS 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. A fee is charged.
- To register, the organization needs to give the name of its server and the IP address of the server.

Dynamic Domain Name System (DDNS)

- 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. The DNS master file must be updated dynamically. The **Dynamic Domain Name System (DDNS)** therefore was devised to respond to this need.
- The primary server updates the zone. The secondary servers are notified either actively or passively.
  - In **active notification**, the primary server sends a message to the secondary servers about the change in the zone, whereas
  - in **passive notification**, the secondary servers periodically check for any changes.

In either case, after being notified about the change, the secondary server requests information about the entire zone (called the **zone transfer**).

Security of DNS

- DNS is one of the most important systems in the Internet infrastructure; it provides crucial services to Internet users.
- DNS can be attacked in several ways: flooding, message interception,
- To protect DNS, IETF has devised a technology named **DNS Security (DNSSEC)** that provides message origin authentication and message integrity using a security service called digital signature.
- DNSSEC, however, does not provide confidentiality for the DNS messages.
- There is no specific protection against the denial-of service attack in the specification of DNSSEC. However, the caching system protects the upper-level servers against this attack to some extent.

Chapter 2: Summary

- ***Applications in the Internet are designed using either a client-server paradigm or a peer-to-peer paradigm. In a client-server paradigm, an application program, called a server, provides services and another application program, called a client, receives services. A server program is an infinite program; a client program is finite. In a peer-to- peer paradigm, a peer can be both a client and a server.***

- ***The World Wide Web (WWW) is a repository of information linked together from points all over the world. Hypertext and hypermedia documents are linked to one another through pointers. The HyperText Transfer Protocol (HTTP) is the main protocol used to access data on the World Wide Web (WWW).***

Chapter 2: Summary (continued)

- *File Transfer Protocol (FTP) is a TCP/IP client-server application for copying files from one host to another. FTP requires two connections for data transfer: a control connection and a data connection. FTP employs NVT ASCII for communication between dissimilar systems.*

- *Electronic mail is one of the most common applications on the Internet. The e-mail architecture consists of several components such as user agent (UA), main transfer agent (MTA), and main access agent (MAA). The protocol that implements MTA is called Simple Main Transfer Protocol (SMTP). Two protocols are used to implement MAA: Post Office Protocol, version 3 (POP3) and Internet Mail Access Protocol, version 4 (IMAP4).*

Chapter 2: Summary (continued)

- *File Transfer Protocol (FTP) is a TCP/IP client-server application for copying files from one host to another. FTP requires two connections for data transfer: a control connection and a data connection. FTP employs NVT ASCII for communication between dissimilar systems.*

- *TELNET is a client-server application that allows a user to log into a remote machine, giving the user access to the remote system. When a user accesses a remote system via the TELNET process, this is comparable to a time-sharing environment.*

Chapter 2: Summary (continued)

- *The Domain Name System (DNS) is a client-server application that identifies each host on the Internet with a unique name. DNS organizes the name space in a hierarchical structure to decentralize the responsibilities involved in naming. TELNET is a client-server application that allows a user to log into a remote machine, giving the user access to the remote system.*