Chapter 2
Application Layer

Prepared by T. Le-Duc and H. Choo

Presentation Outline

- 2.1 Introduction
- 2.2 Client-Server Paradigm
- 2.3 Standard Client-Server Applications
- 2.4 Peer-To-Peer Paradigm
- 2.5 Socket-Interface Programming

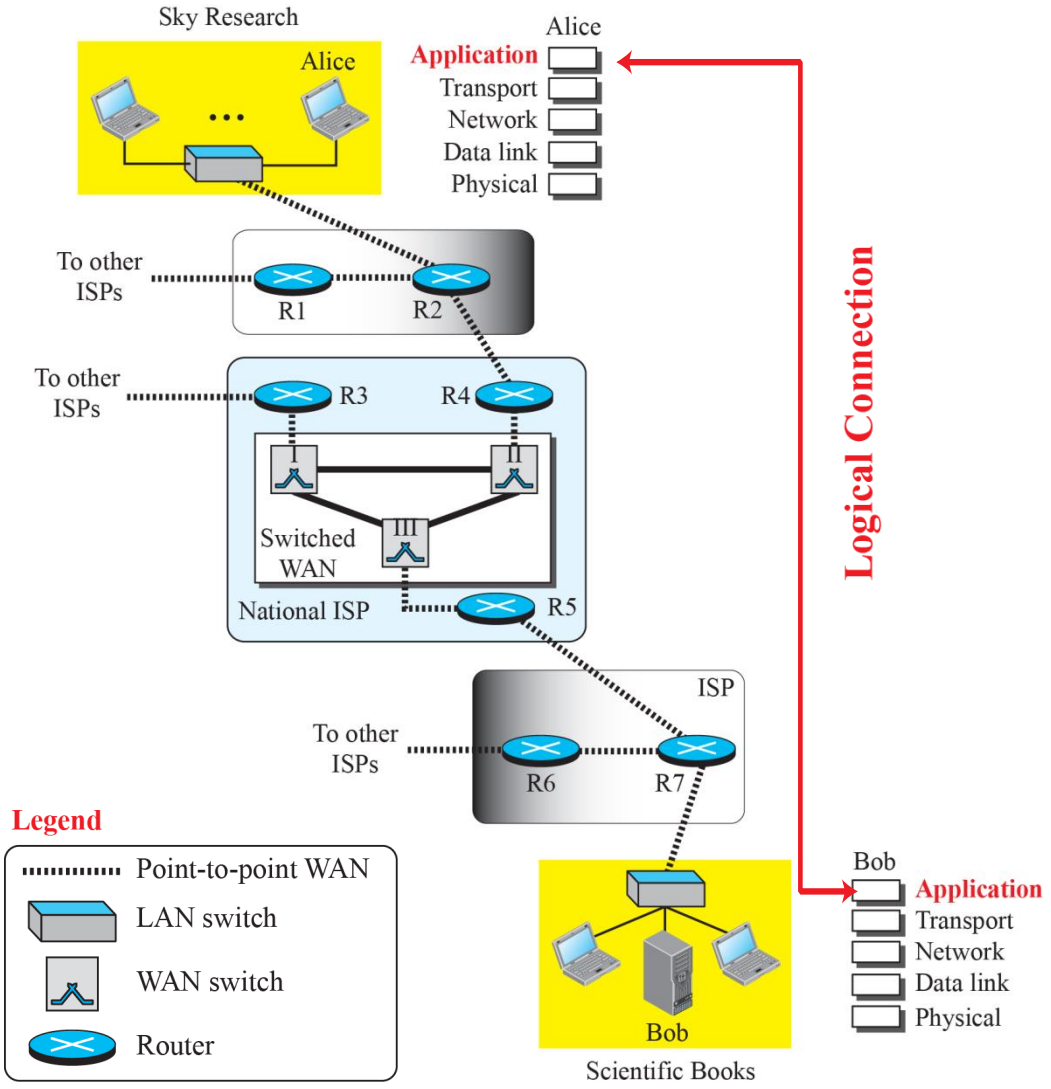
2.1 Introduction

- The whole Internet was designed and developed to provide services at the application layer
- The application layer provides services to the Internet user
- Communication is provided using a logical connection
  - Two applications assume that there is an imaginary direct connection through which they can send and receive messages

2.1 Introduction

Scenario

- A scientist at Sky Research needs to order a book from an online bookseller, Scientific Books



[Figure 2.1 Logical connection at the application layer]

2.1 Introduction

Providing Services

- Applications and services are added to the Internet constantly
- Standard Application-Layer protocols
  - Protocols that have been standardized and documented by the Internet authority
  - E.g. HTTP, FTP, SMTP, POP3, Telnet, …
- Nonstandard Application-Layer protocols
  - Protocols do not need the approval of the Internet authority
  - E.g. customized protocol developed by a company to communicate with all its offices

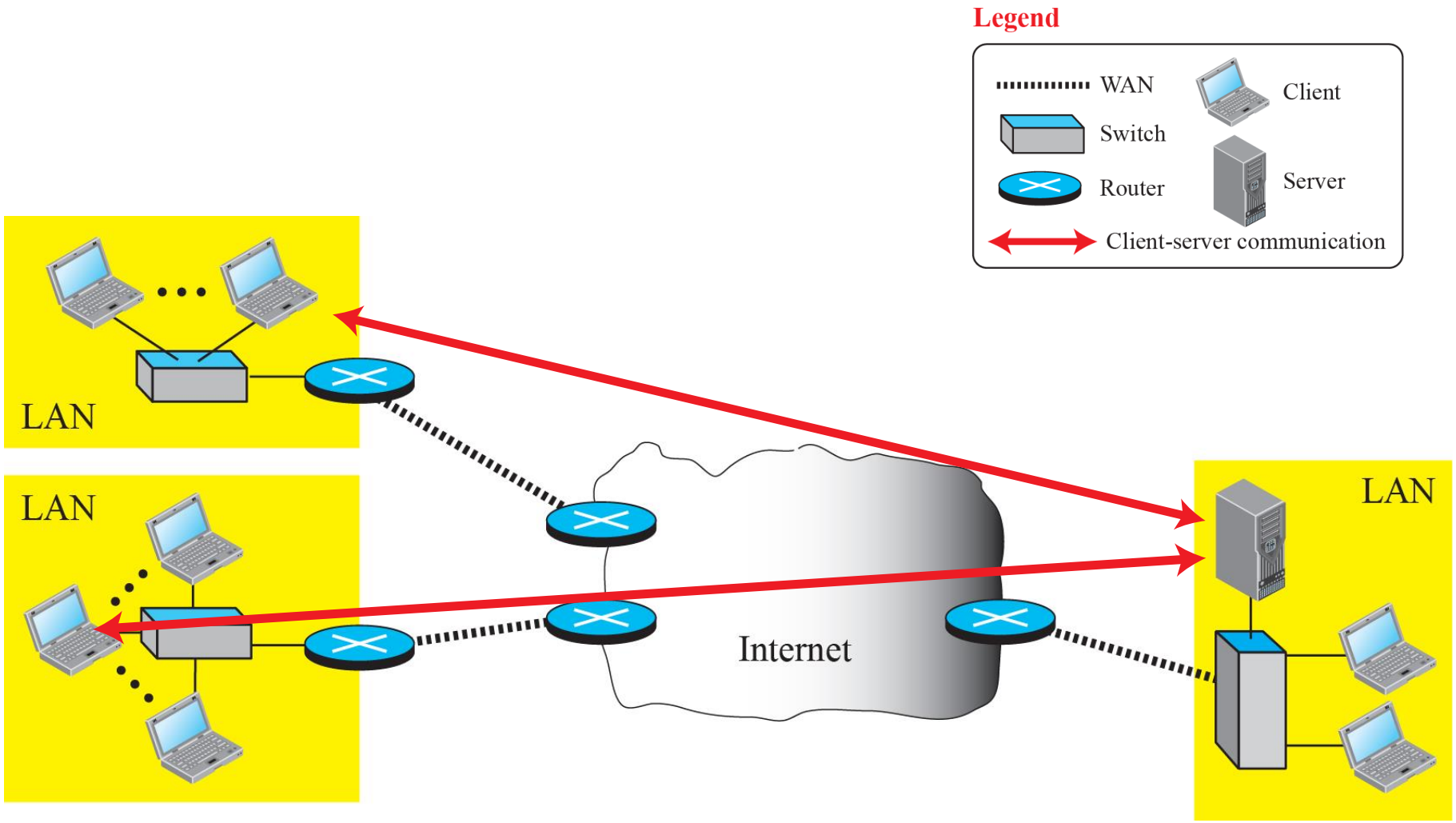
2.1 Introduction

Application-Layer Paradigms: Client-Server paradigm (1/2)

- Server
  - Always-on host
  - Provides requested service to client
  - Permanent IP address
- Client
  - Initiates contact with server
  - Typically requests service to server
  - Do not communicate directly with each other
- WWW, FTP, Remote login (Telnet, SSH), e-mail, …

2.1 Introduction

Application-Layer Paradigms: Client-Server paradigm (2/2)



[Figure 2.2 Example of a client-server paradigm]

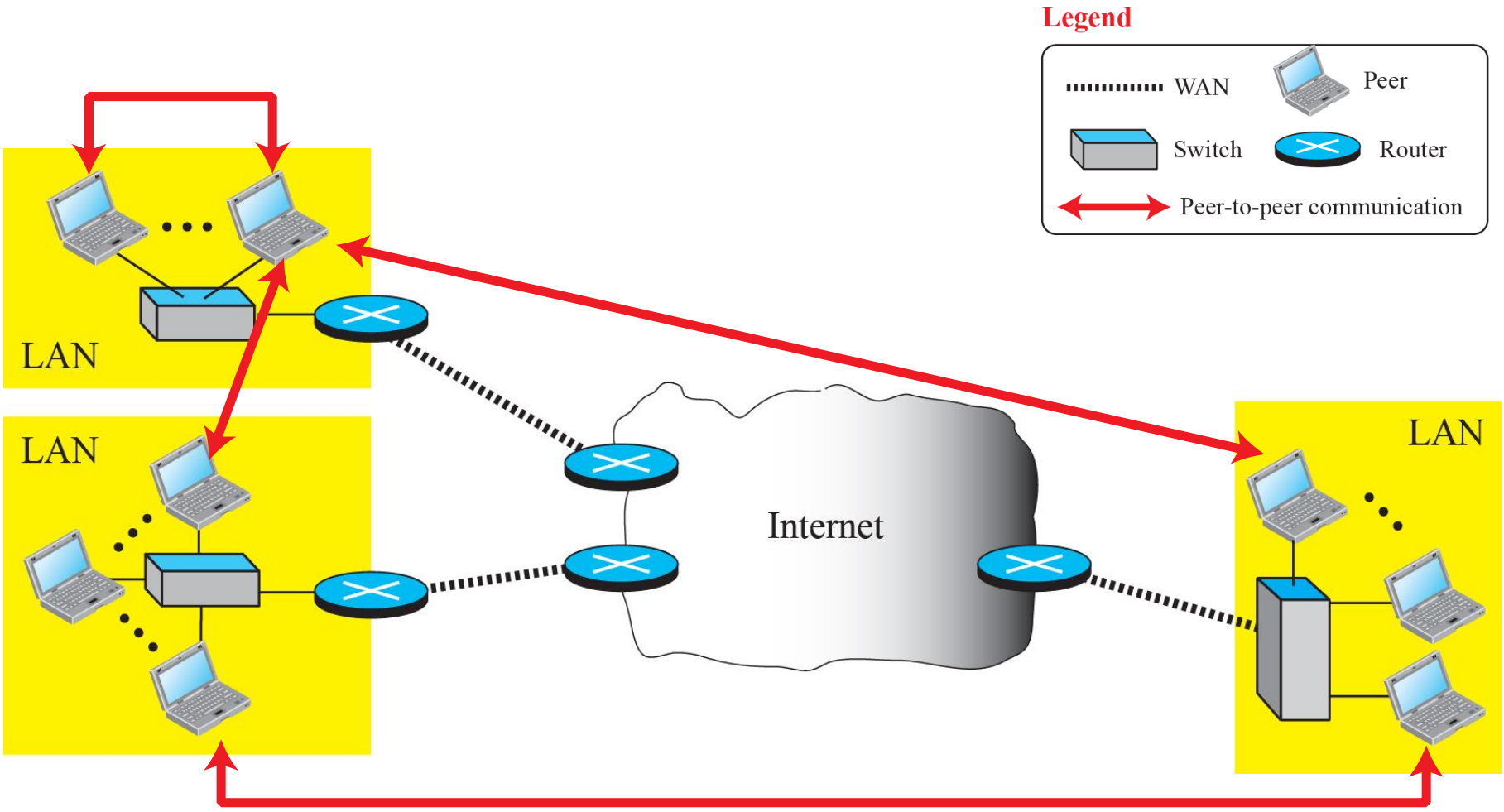
2.1 Introduction

Application-Layer Paradigms: Peer-to-Peer Paradigm (1/2)

- Peer-to-peer (P2P) paradigm
  - No always-on server
  - Arbitrary end systems directly communicate
  - Peers are intermittently connected and change their IP address
  - Highly scalability, increase service capability
  - Highly distributed and decentralized nature
  - Difficult to manage

2.1 Introduction

Application-Layer Paradigms: Peer-to-Peer Paradigm (2/2)

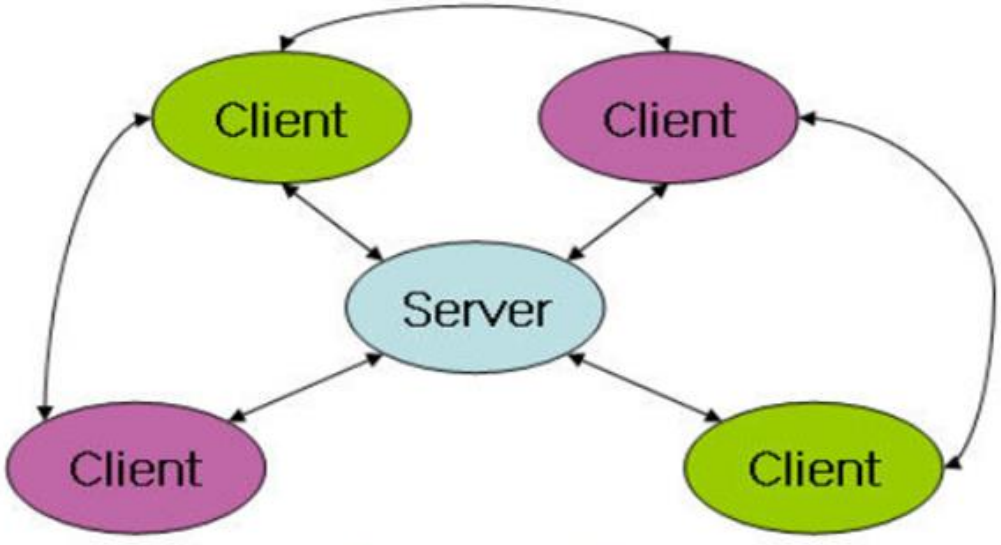


[Figure 2.3 Example of a peer-to peer paradigm]

2.1 Introduction

Application-Layer Paradigms: Mixed Paradigm

- Mixed paradigm (Client-Server paradigm + P2P)
  - Client-Server paradigm: a client get the information from a server
  - P2P: the client can share the information with other clients
  - E.g. Nate-on messenger, Yahoo messenger



[Example of Mixed paradigm]

2.2 Client-Server Paradigm

Application Programming Interface

- Communication at the application layer is between two applications 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
- Application Programming Interface (API)
  - 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
  - A set of instructions to tell the lowest four layers of the TCP/IP suite to open connection, send and receive data, and close the connection is referred as an API for networking applications

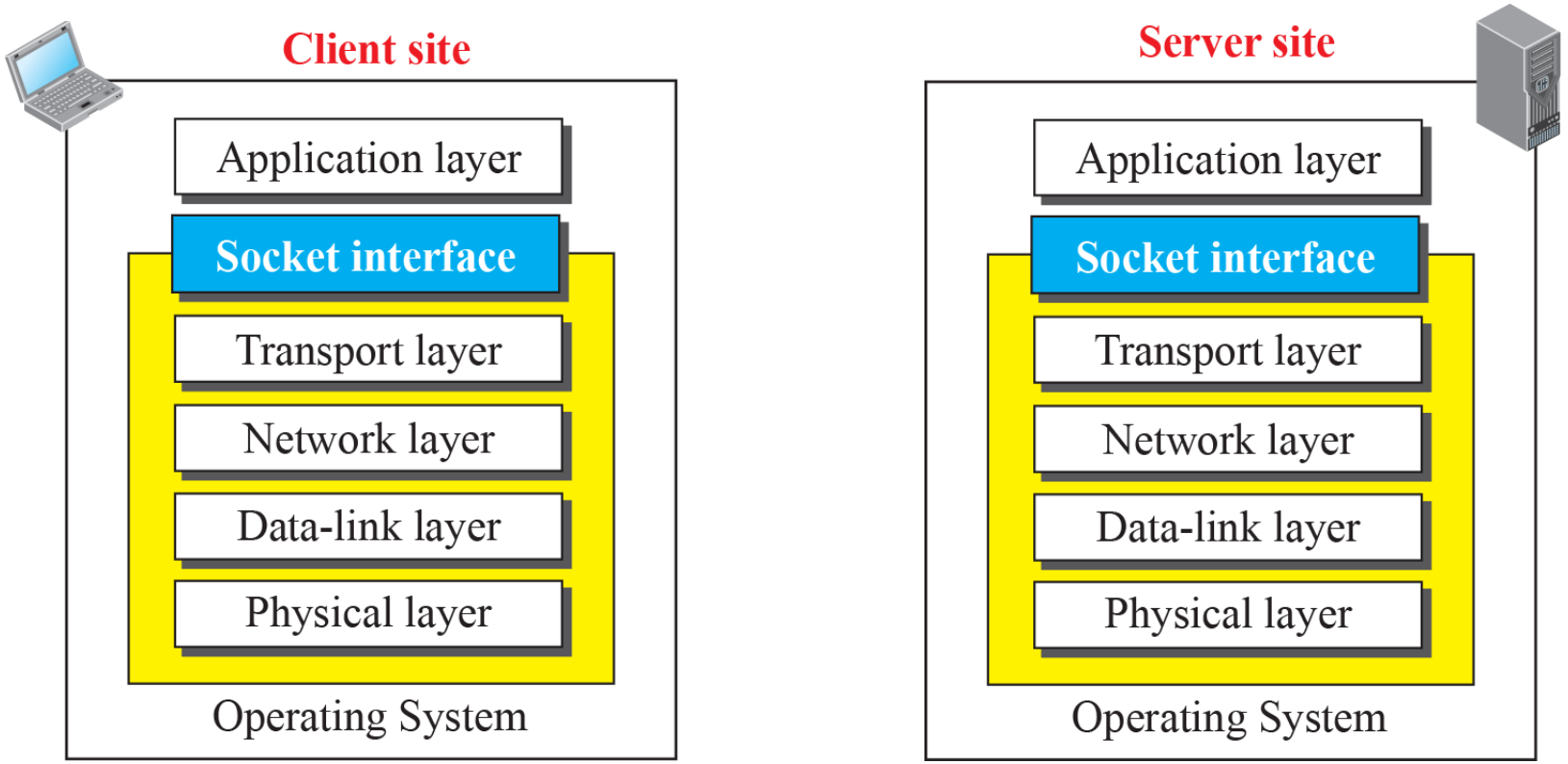
2.2 Client-Server Paradigm

Application Programming Interface: Sockets (1/4)

- Socket interface started in the early 1980s at UC Berkeley as part of a UNIX environment
- The socket interface is a set of instructions that provide communication between the application layer and the operating system
- Socket
  - Although a socket is supposed to behave like a terminal or a file, it is not a physical entity like them
  - It is a data structure that is created and used by the application program

2.2 Client-Server Paradigm

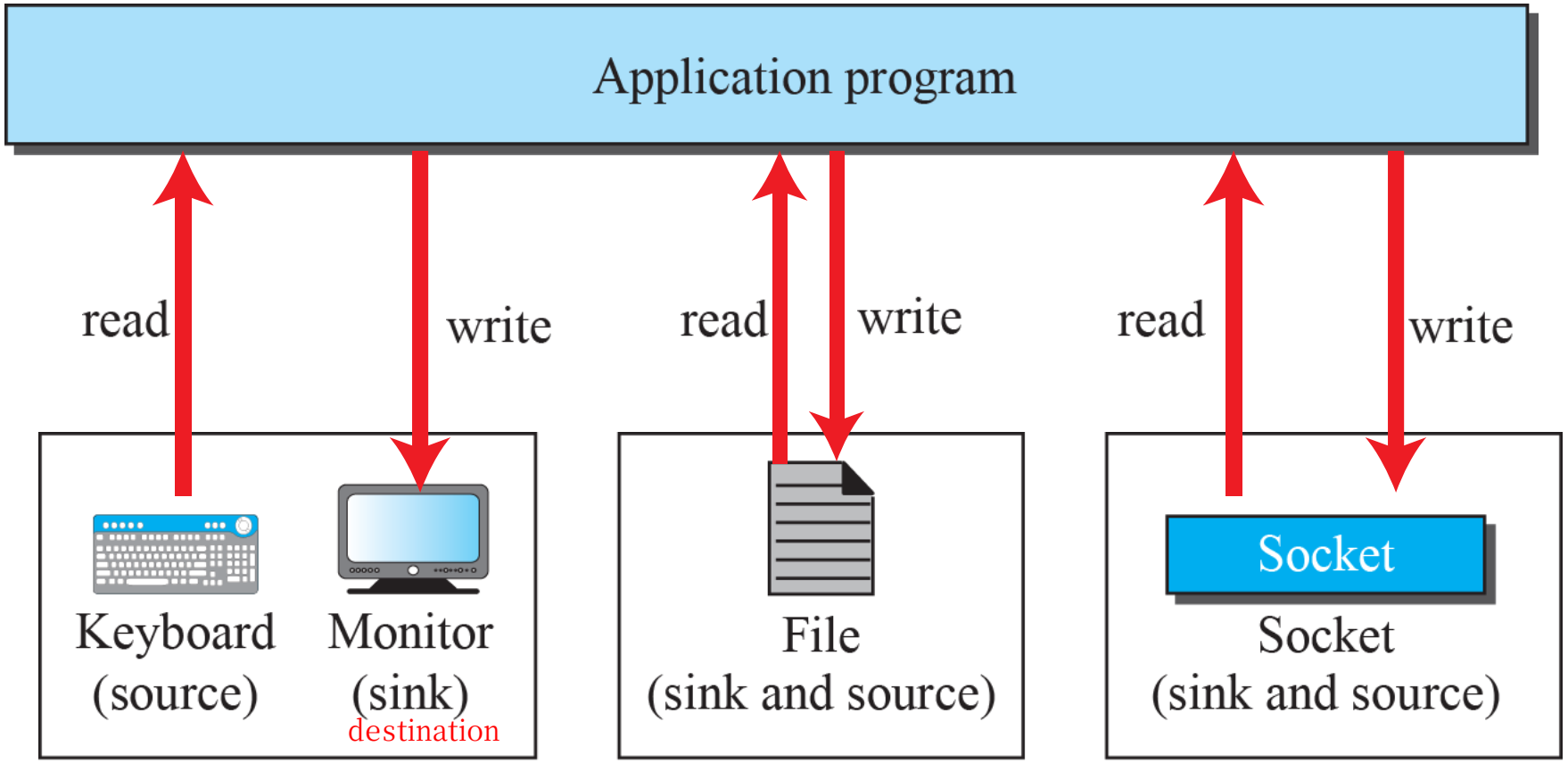
Application Programming Interface: Sockets (2/4)



[Figure 2.4 Position of the socket interface]

2.2 Client-Server Paradigm

Application Programming Interface: Sockets (3/4)

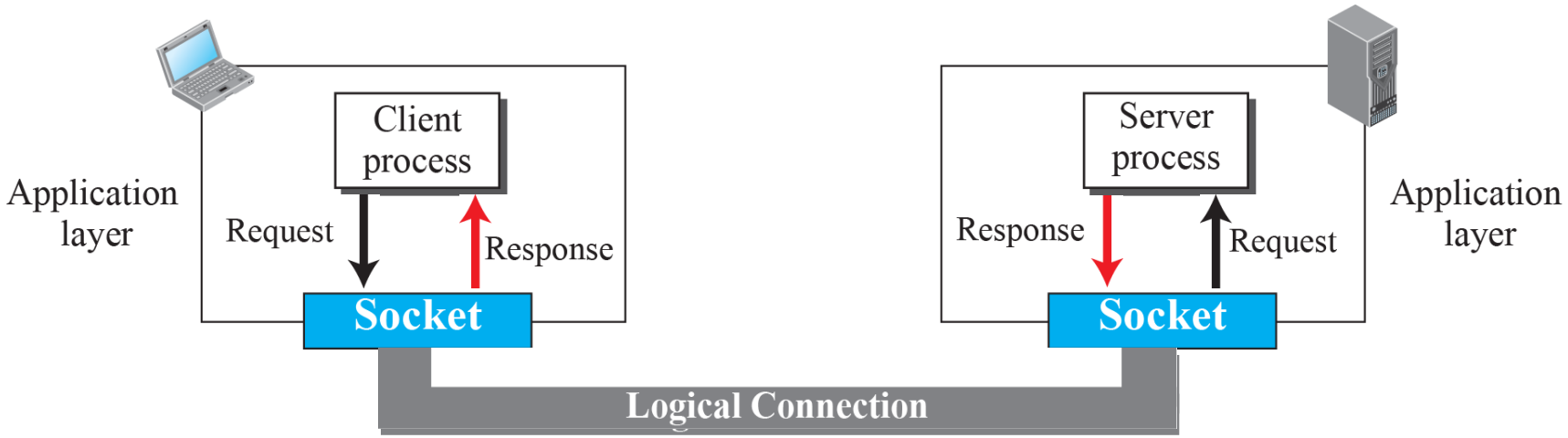


[Figure 2.5 Sockets used the same way as other sources and sinks]

2.2 Client-Server Paradigm

Application Programming Interface: Sockets (4/4)

- Sockets can be used for the communication between a client process and sever process
- The client thinks that the socket is the entity that receives the request and gives the response
- The server thinks that the socket is the one that has a request and needs the response



[Figure 2.6 Use of socket in process-to-process communication]

2.2 Client-Server Paradigm

Application Programming Interface: Sockets Addresses

- The interaction between a client and a server is two-way communication
- In a two-way communication, we need a pair of addresses: local (sender) and remote (receiver)
- Since communication in the client-server paradigm is between two sockets, we need a pair of **socket addresses** for communication: a local socket address and a remote socket address

[Figure 2.7 A socket address]

2.2 Client-Server Paradigm

Application Programming Interface: Server Site

- Server needs a local (server) and a remote (client) socket address for communication

- Local Socket Address
  - Being provided by the operating system
    - IP address of the computer on which the server is running
    - Port is determined and assigned when the server starts running
  - Being fixed and used during the server's life time

- Remote Socket Address
  - Being determined when a client tries to connect to the server
  - Corresponding to one particular client (one-to-one)

2.2 Client-Server Paradigm

Application Programming Interface: Client Site

- Client needs a local (client) and a remote (server) socket address for communication
- Local Socket Address
  - Being provided by the operating system
    - IP address of the computer on which the client is running
    - Port is assigned when the client needs to start the communication
- Remote Socket Address
  - IP address and port (of the server) should be provided by the user
    - IP address can be given directly or through domain name (discussed later)

2.2 Client-Server Paradigm

Using Services of the Transport Layer

- A pair of processes provide services to the Internet users, 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 3 common transport layer protocols
  - UDP
  - TCP
  - SCTP

2.2 Client-Server Paradigm

Using Services of the Transport Layer: UDP Protocol

- UDP provides connectionless, unreliable, datagram service
  - Connectionless service means that there is no logical connection between the two ends exchanging messages
- Each message is an independent entity encapsulated in a packet called a datagram
- UDP does not see any relation (connection) between consequent datagrams coming from the same source and going to the same destination
- UDP is suitable for applications
  - Small messages usage
  - Simplicity and speed are more important than reliability
  - E.g.: management and multimedia applications

2.2 Client-Server Paradigm

Using Services of the Transport Layer: TCP Protocol

- TCP provides connection-oriented, reliable, byte-stream service
- TCP requires that two ends first create a logical connection between themselves by exchanging some connection-establishment packets
  - Handshaking process: establishes some parameters between the two ends including the size of the data packets to be exchanged
- Most of the standard applications that need to send long messages and require reliability may benefit from the service of the TCP
  - E.g.: File transfer applications

2.2 Client-Server Paradigm

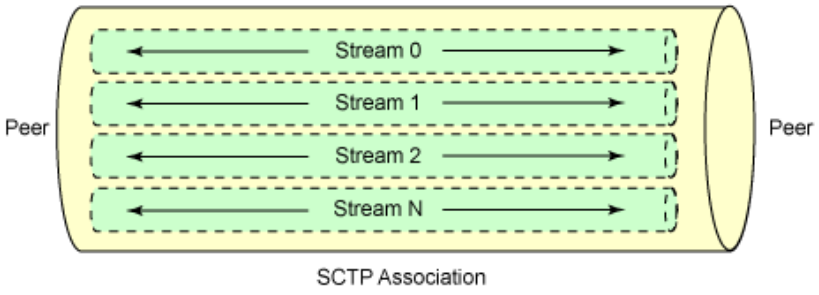
Using Services of the Transport Layer: SCTP Protocol (1/2)

- SCTP provides a service which is a combination of the two other protocols
- Like TCP, SCTP provides a connection-oriented, reliable service, but it is not byte-stream oriented
- It is a message-oriented protocol like UDP
- In addition, SCTP can provide multi-stream service by providing multiple network-layer connections
- SCTP is normally suitable for any application that needs reliability and at the same time needs to remain connected, even if a failure occurs in one network-layer connection

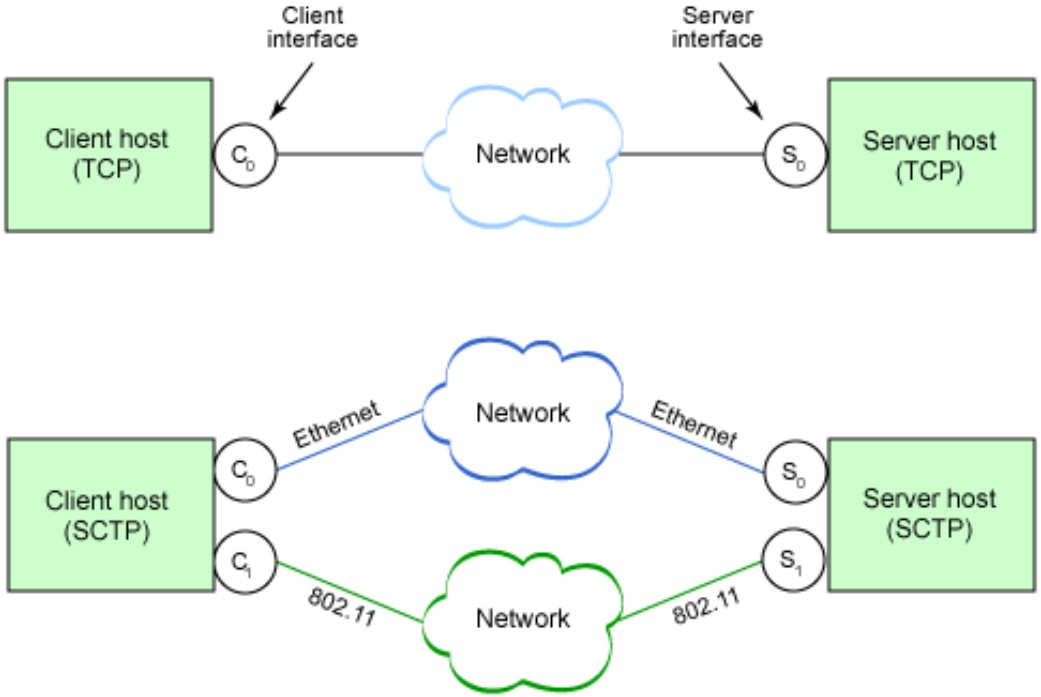
2.2 Client-Server Paradigm

Using Services of the Transport Layer: SCTP Protocol (2/2)

- Benefits of SCTP
  - Multi-homing
  - Multi-streaming



[Multi-streaming]



[Multi-homing]

Practice Problems

1. List three non-proprietary Internet applications and the application-layer protocols that they use.

WWW: HTTP
FTP
SMTP, POP3 and IMAP

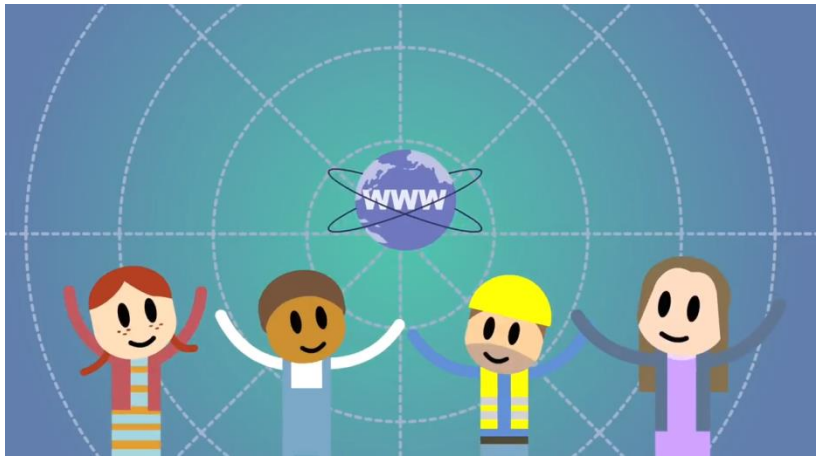
2. For a communication session between two hosts, which host is the client and which is the server?

요청하는 애가 client www service: a browser is a client process

2.3 Standard Client-Server Application

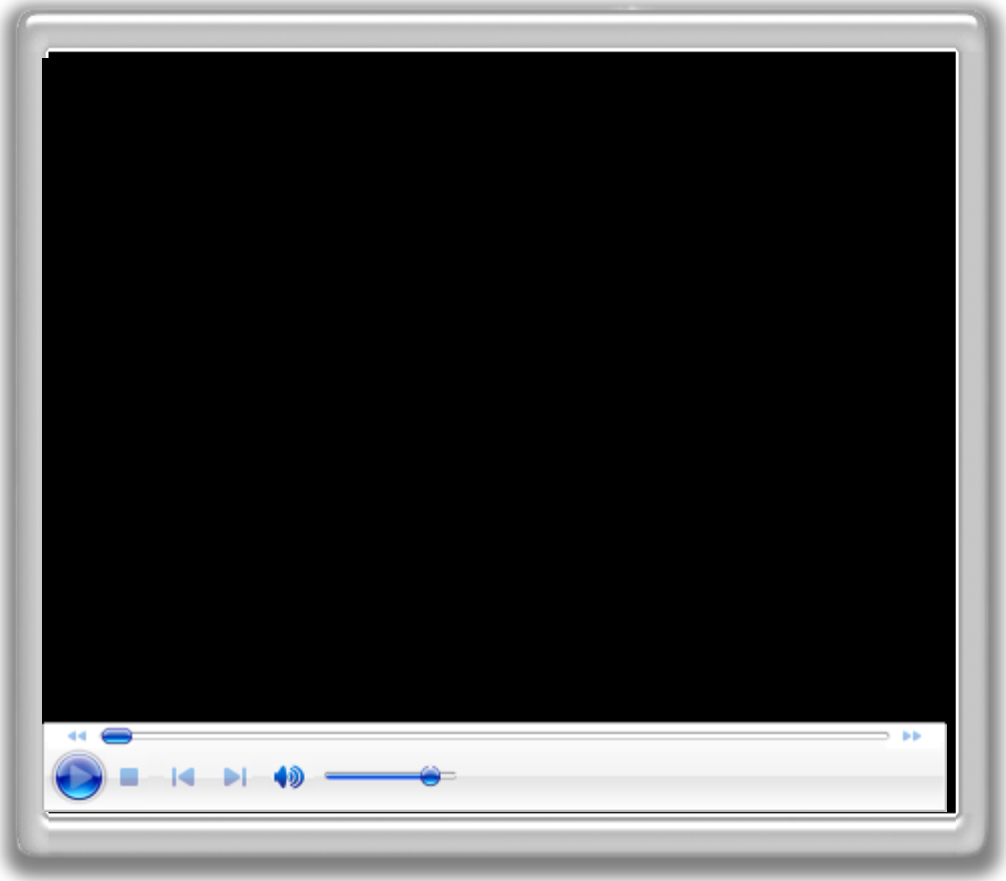
World Wide Web and HTTP: Introduction (1/2)

- Video Content
  - The World Wide Web (WWW) is used every day by millions of people for everything from checking the weather to sharing videos
  - This interconnected information system (WWW) is described as a virtual city that everyone owns and explains how it is organized in a way that mimics our brain's natural way of thinking
  - Link: https://www.youtube.com/watch?v=J8hzJxb0rpc



2.3 Standard Client-Server Application

World Wide Web and HTTP: Introduction (2/2)



2.3 Standard Client-Server Application

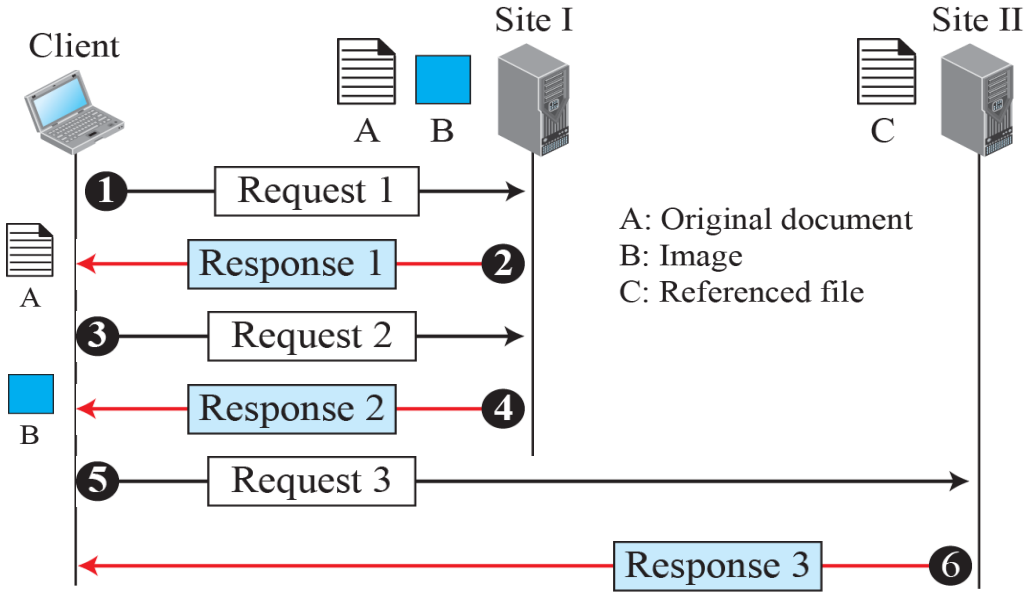
World Wide Web and HTTP: World Wide Web (1/6)

- The idea of the Web was first proposed by Tim Berners-Lee in 1989 at *Conseil European pour la Recherche Nuclear (CERN)*
- The Web today is a repository of information in which the documents, called *Web pages*, are distributed all over the world and related documents are linked together
- The purpose of the Web is not only to retrieve linked documents but also to provide electronic shopping and gaming
- World Wide Web (WWW) today is a distributed client-server service, in which a client using a browser can access a service provided by a server

2.3 Standard Client-Server Application

World Wide Web and HTTP: World Wide Web (2/6)

- Example 2.2
  - 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

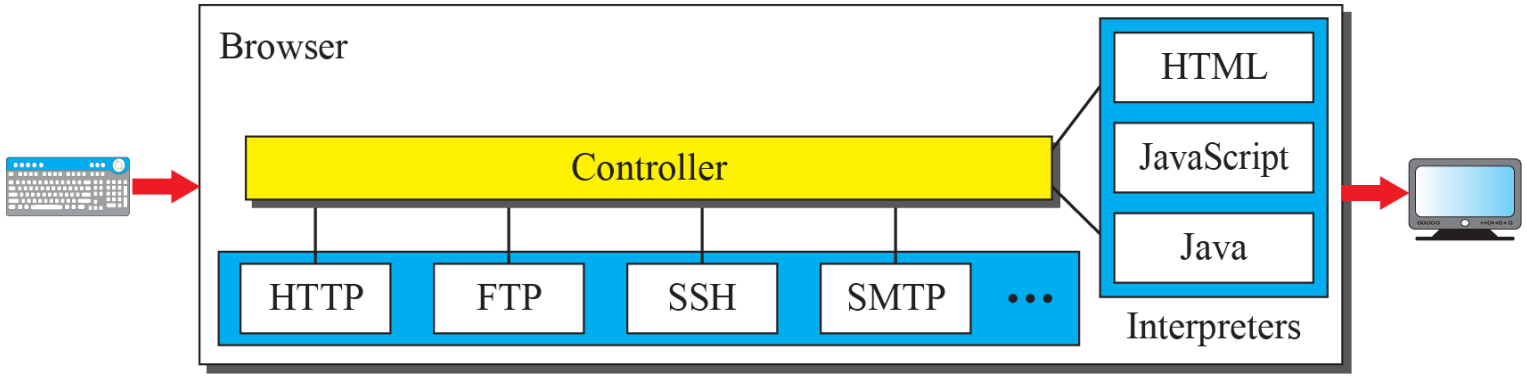


[Figure 2.8 Example 2.2]

2.3 Standard Client-Server Application

World Wide Web and HTTP: World Wide Web (3/6)

- Web Client (Browser)
  - A variety of vendors offer commercial browsers that interpret and display a web page, and all of them use nearly the same architecture
    - E.g.: Microsoft Internet Explorer, Mozilla Firefox, Google Chrome
  - Each browser usually consists of three parts: *a controller*, *client protocols*, and *interpreters*



[Figure 2.9 Browser]

2.3 Standard Client-Server Application

World Wide Web and HTTP: World Wide Web (4/6)

- Web Server
  - The web page is stored at the server
  - Each time a request arrives, the corresponding document is sent to the client
  - To improve efficiency, servers normally store requested files in a cache in memory; memory is faster to access than disk
  - A server can also become more efficient through multithreading or multiprocessing
  - Some popular web servers include **Apache** and **Microsoft Internet Information Server**

2.3 Standard Client-Server Application

World Wide Web and HTTP: World Wide Web (5/6)

- Uniform Resource Locator (URL)
  - A web page, as a file, needs to have a unique identifier to distinguish it from other web pages.
  - To define a web page, we need four identifiers, including *protocol*, *host*, *port*, and *path*
    - **Protocol**: the abbreviation for the client-server program that we need in order to access the web page
    - **Host**: the IP address of the server or the unique name given to the server
    - **Port**: a 16-bit integer, is normally predefined for the client-server application
    - **Path**: the location and the name of the file in the underlying operating system

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

2.3 Standard Client-Server Application

World Wide Web and HTTP: World Wide Web (6/6)

- Web Documents
  - **Static Documents**
    - Fixed-content documents that are created and stored in a server
    - Client can get a copy of the document only
    - Language: *Hypertext Markup Language* (HTML), *Extensible Markup Language* (XML), *Extensible Style Language* (XSL), and *Extensible Hypertext Markup Language* (XHTML)
  - **Dynamic documents**
    - Created by a web server whenever a browser requests the document
    - A very simple example of a dynamic document is the retrieval of the time and date from a server
    - Language: *Common Gateway Interface* (CGI), *Java Server Pages* (JSP), *Active Server Pages* (ASP), and *ColdFusion*
  - **Active Documents**
    - Programs or scripts being run at the client site
    - Language: JavaScript or Java (Applet)

2.3 Standard Client-Server Application

World Wide Web and HTTP: HTTP (1/20)

- The **HyperText Transfer Protocol (HTTP)** is a protocol that 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

2.3 Standard Client-Server Application

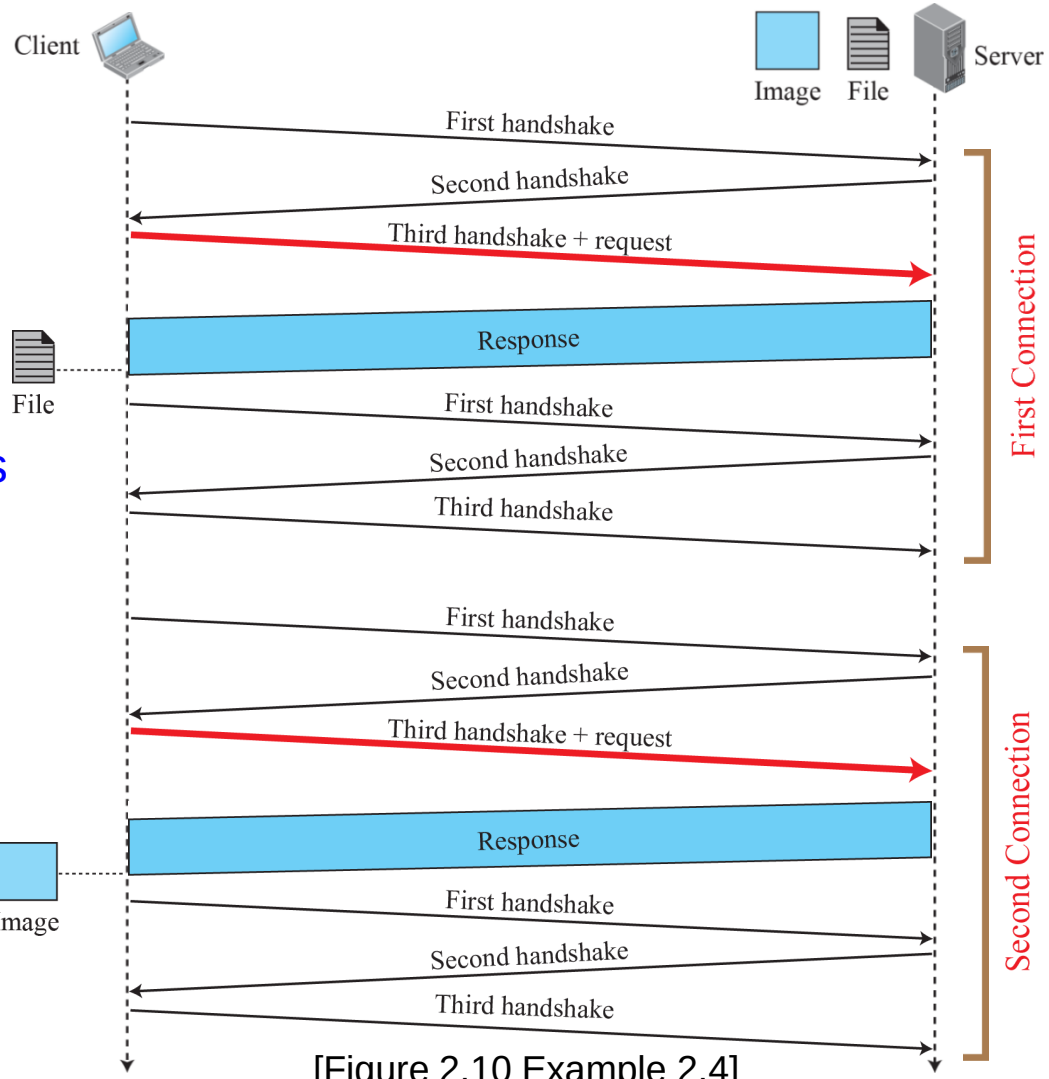
World Wide Web and HTTP: HTTP (2/20)

- Nonpersistent Connections
  - In a nonpersistent connection, one TCP connection is made for each request/response
  - The following lists the steps in this strategy
    - The client opens a TCP connection and sends a request
    - The server sends the response and closes the connection
    - The client reads the data until it encounters an end-of-file marker and then it closes the connection

2.3 Standard Client-Server Application

World Wide Web and HTTP: HTTP (3/20)

- Example 2.4
  - The client accesses 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



[Figure 2.10 Example 2.4]

2.3 Standard Client-Server Application

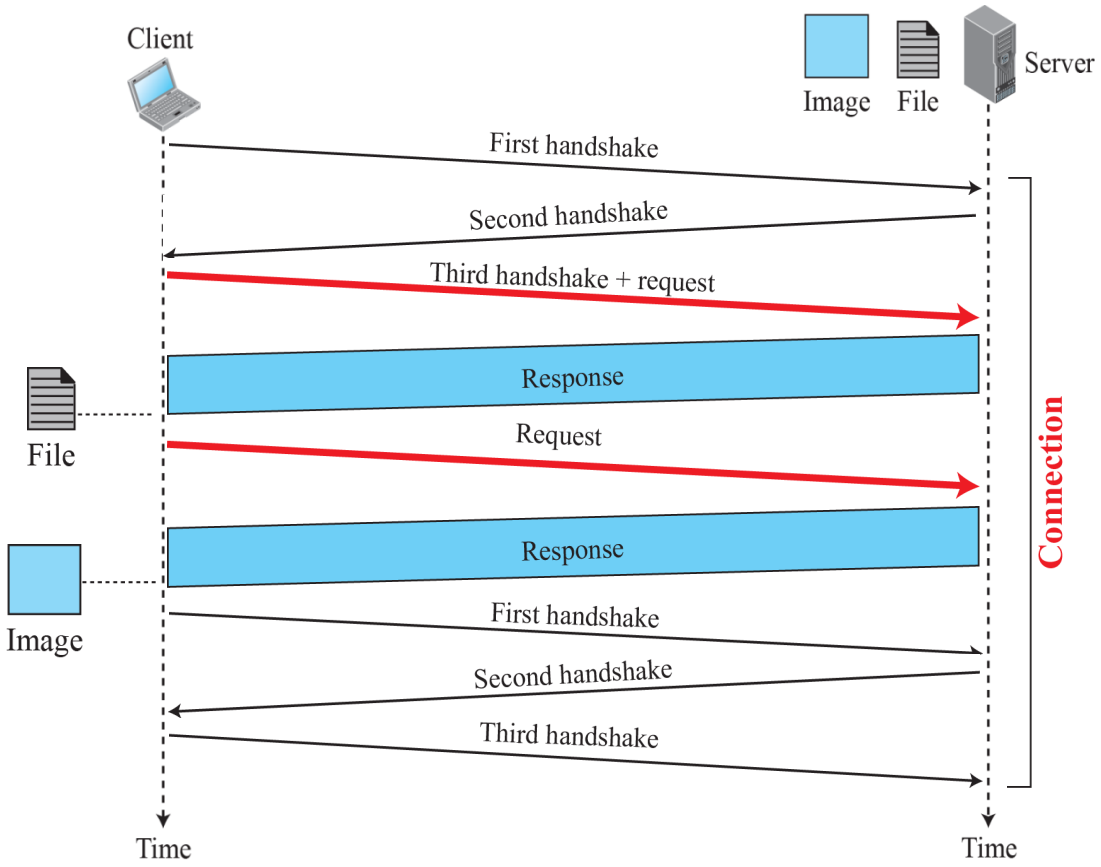
World Wide Web and HTTP: HTTP (4/20)

- Persistent Connections
  - In a persistent connection, the server leaves the connection open for more requests after sending a response
  - The server can close the connection at the request of a client or if a time-out has been reached
  - The server usually sends the length of the data with each response
    - If document is created dynamically or actively, the length of data is unknown; so server closes the connection after sending the data to notify client of the end of data
  - Time and resources are saved using persistent connections

2.3 Standard Client-Server Application

World Wide Web and HTTP: HTTP (5/20)

- Example 2.5
  - The same scenario as in Example 2.4.
  - Only one connection establishment and one connection termination for two separate requests

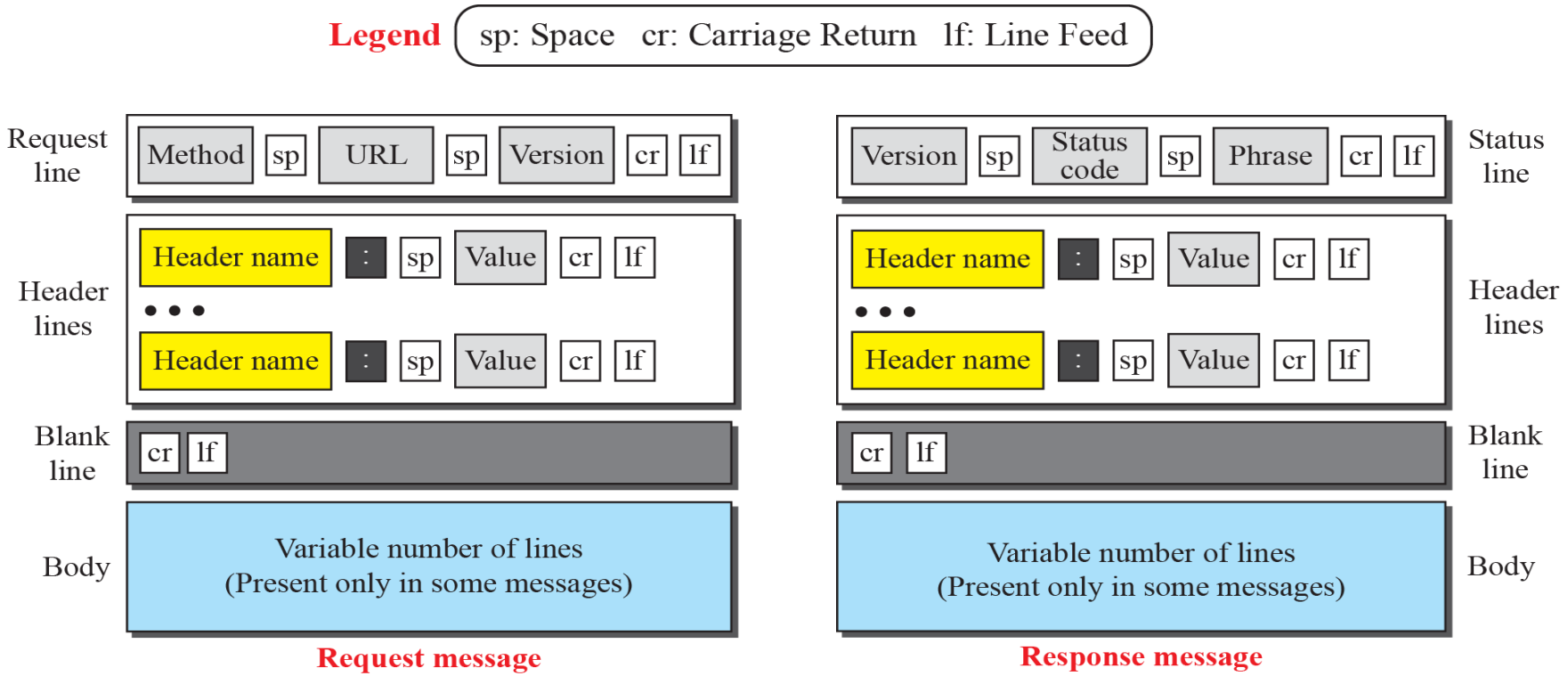


[Figure 2.11 Example 2.5]

2.3 Standard Client-Server Application

World Wide Web and HTTP: HTTP (6/20)

- Message formats (request message + response message)
  - HTTP protocol defines the format of the request and response messages as follows



[Figure 2.12 Formats of the request and response messages]

2.3 Standard Client-Server Application

World Wide Web and HTTP: HTTP (7/20)

- Message formats: **request message** (1/2)
  - The first line in a request message is called a request line
  - There are three fields in this line called *method*, URL, and *version*
  - In version 1.1 of HTTP, several methods are defined as shown in Table 2.1

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

[Table 2.1 Methods]

2.3 Standard Client-Server Application
World Wide Web and HTTP: HTTP (8/20)

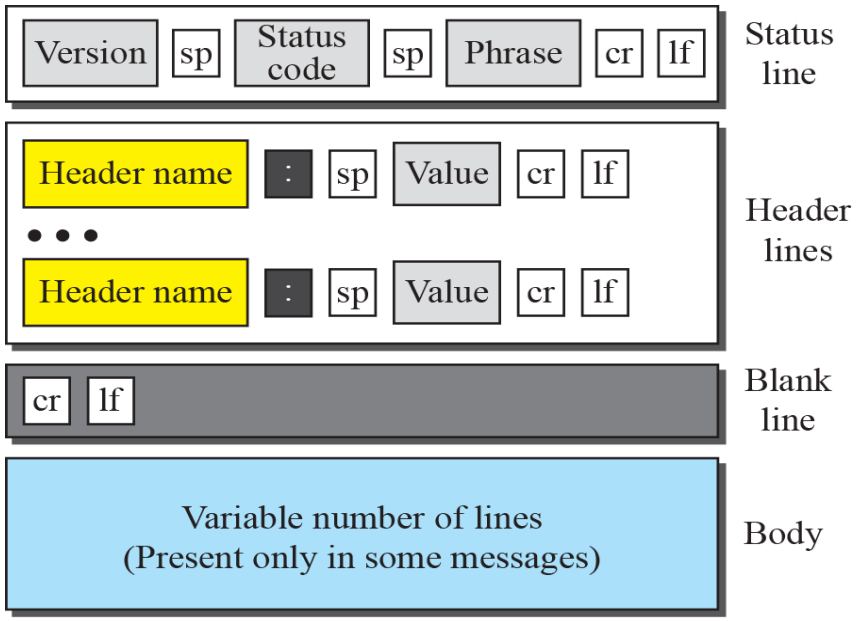
- Message formats: **request message** (2/2)
  - After the request line, we can have zero and more *request header lines*
  - Each *header line* sends additional information from the client to the server
    - A header line includes a header name and a value
  - The body can be present with the comment to be sent or the file to be published for POST or PUT

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

[Table 2.2 Request Header Names]

2.3 Standard Client-Server Application
World Wide Web and HTTP: HTTP (9/20)

- Message formats: **response message** (1/2)
  - A response message consists of a status line, header lines, a blank line, and sometimes a body
  - Status code in status line includes
    - In 100 range: informational code
    - In 200 range: successful request
    - In 300 range: redirect the client to another URL
    - In 400 range: error at client site
    - In 500 range: error at the server site



[Format of a response message]

2.3 Standard Client-Server Application
World Wide Web and HTTP: HTTP (10/20)

- Message formats: **response message** (2/2)
  - After the status line, we can have zero or more response header lines
  - Each header line sends additional information from the server to the client
    - A header line includes a header name and a value
  - The body contains the document to be sent from the server to the client

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

[Table 2.2 Response Header Names]

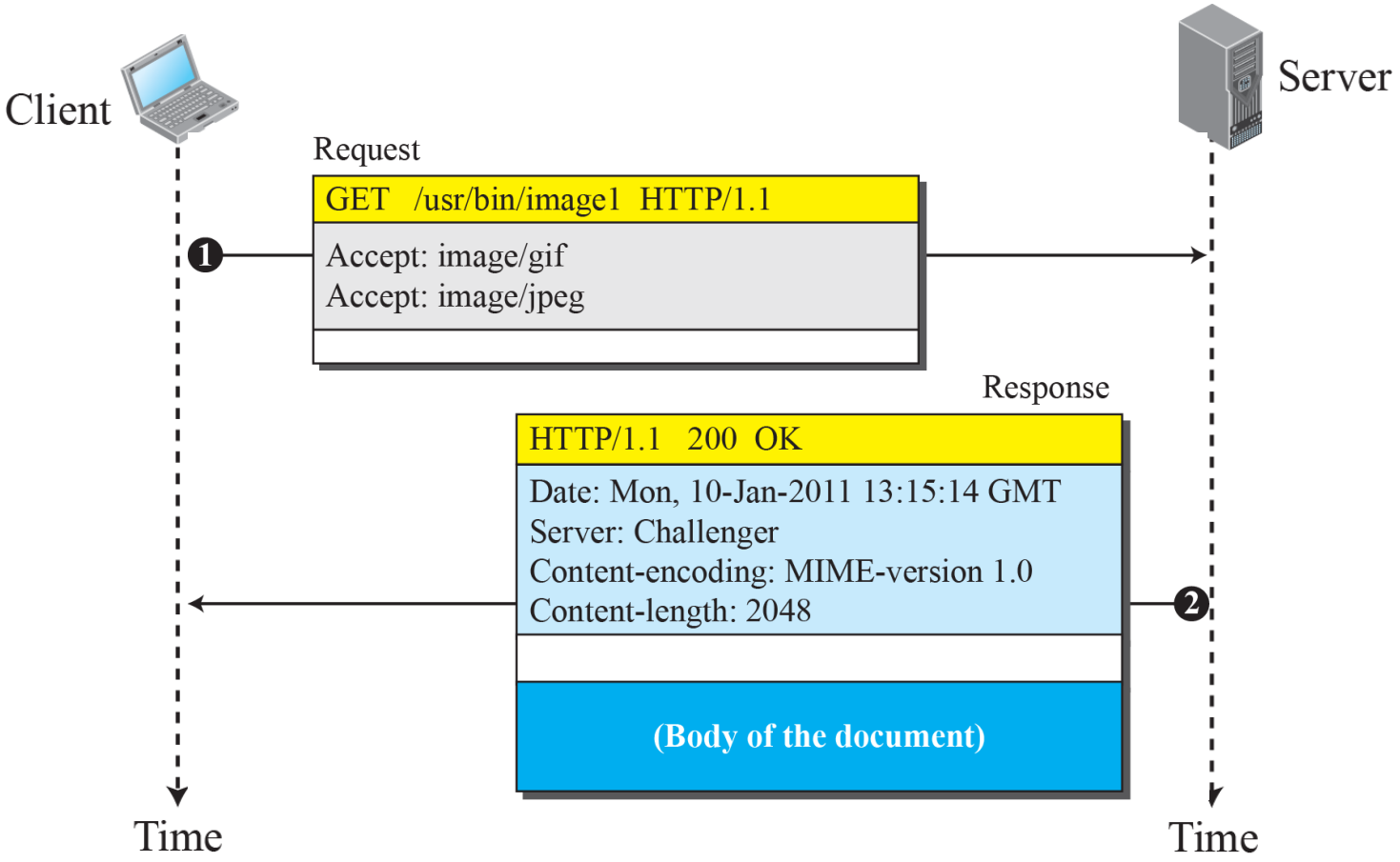
2.3 Standard Client-Server Application
World Wide Web and HTTP: HTTP (11/20)

- **Example 2.6**: Retrieving a document (see Figure 2.13)
  - GET method is used to retrieve an image with the path /usr/bin/image1
  - The request line shows the method (GET), URL, and 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 and length of the document

2.3 Standard Client-Server Application

World Wide Web and HTTP: HTTP (12/20)

■ **Example 2.6** (contd)



[Figure 2.13 Example 2.6]

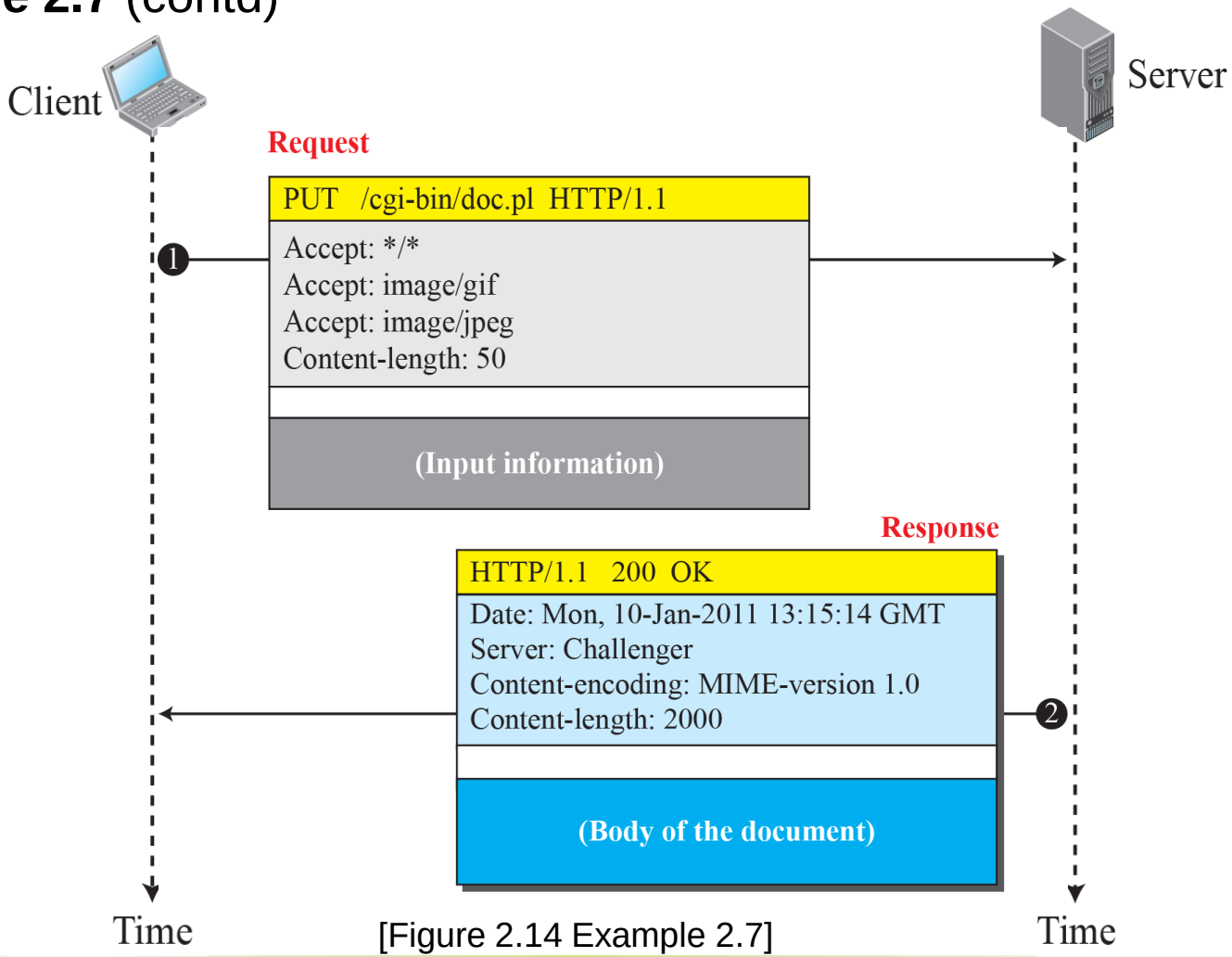
2.3 Standard Client-Server Application
World Wide Web and HTTP: HTTP (13/20)

- **Example 2.7:** Client sends a web page to be posted on the server (see Figure 2.14)
  - 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 is included as the body

2.3 Standard Client-Server Application

World Wide Web and HTTP: HTTP (14/20)

■ **Example 2.7** (contd)



[Figure 2.14 Example 2.7]

2.3 Standard Client-Server Application

World Wide Web and HTTP: HTTP (15/20)

- **Example 2.8:** Conditional Request
  - Client request the server to return the page only if it is modified after a certain point in time

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

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

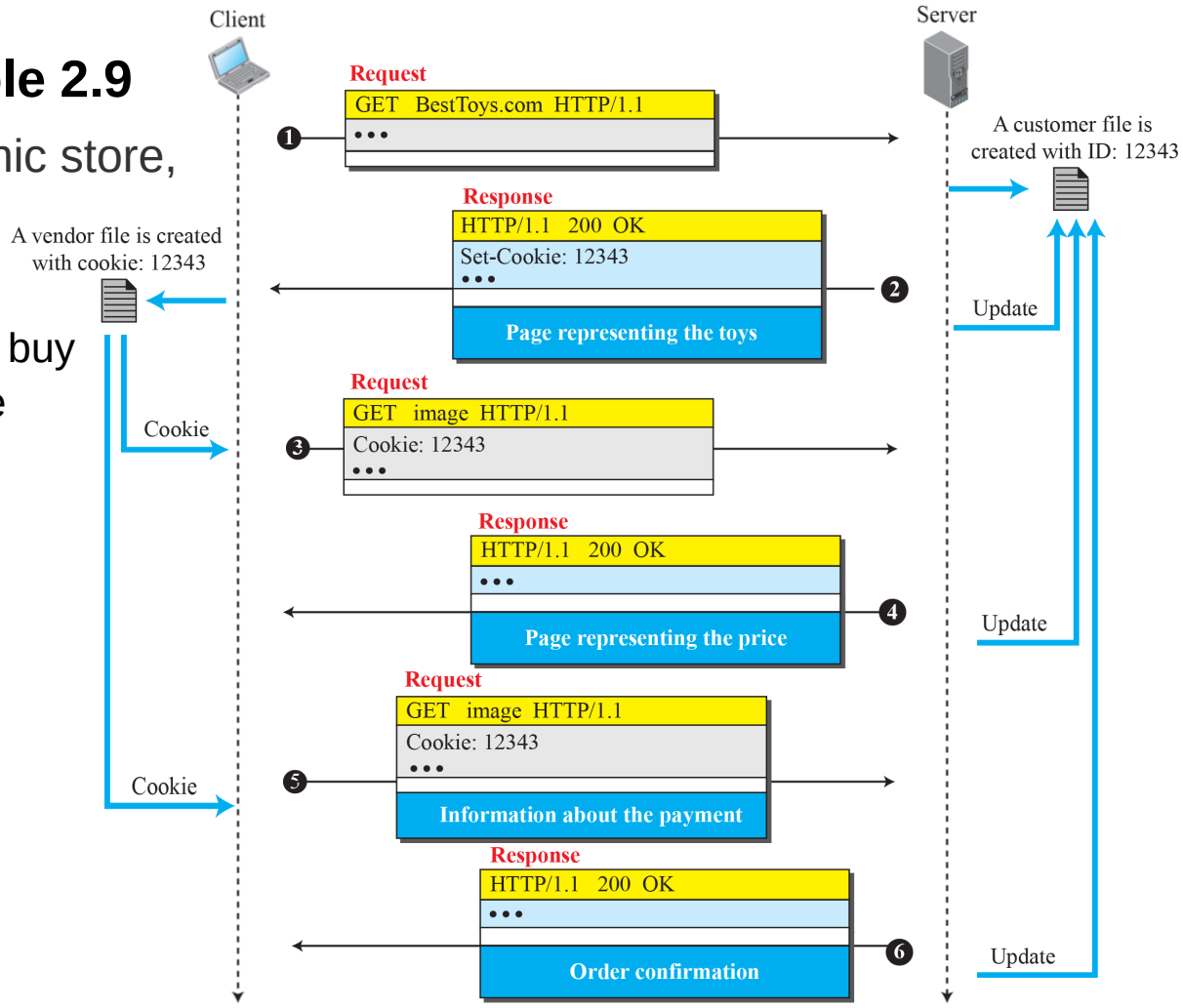
2.3 Standard Client-Server Application
World Wide Web and HTTP: HTTP (16/20)

- **Cookies (1/2)**
  - Cookies are used for web server to remember some information about the clients
  - Creating and Storing Cookies
    - Server includes the cookie containing information about the client in the response and send it to the client
    - Client stores the cookie in the cookie directory
  - 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 cookies 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
  - The contents of the cookie are never read by the browser or disclosed to the user; it is made by the server and eaten by the server

2.3 Standard Client-Server Application

World Wide Web and HTTP: HTTP (17/20)

- **Cookies (2/2): Example 2.9**
  - Scenario of a electronic store, BestToys.com, using cookies
    - A shopper wants to buy a toy from this store



[Figure 2.15 Example 2.9]

2.3 Standard Client-Server Application
World Wide Web and HTTP: HTTP (18/20)

- **Web Caching: Proxy Server**
  - HTTP supports **proxy servers**, which 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

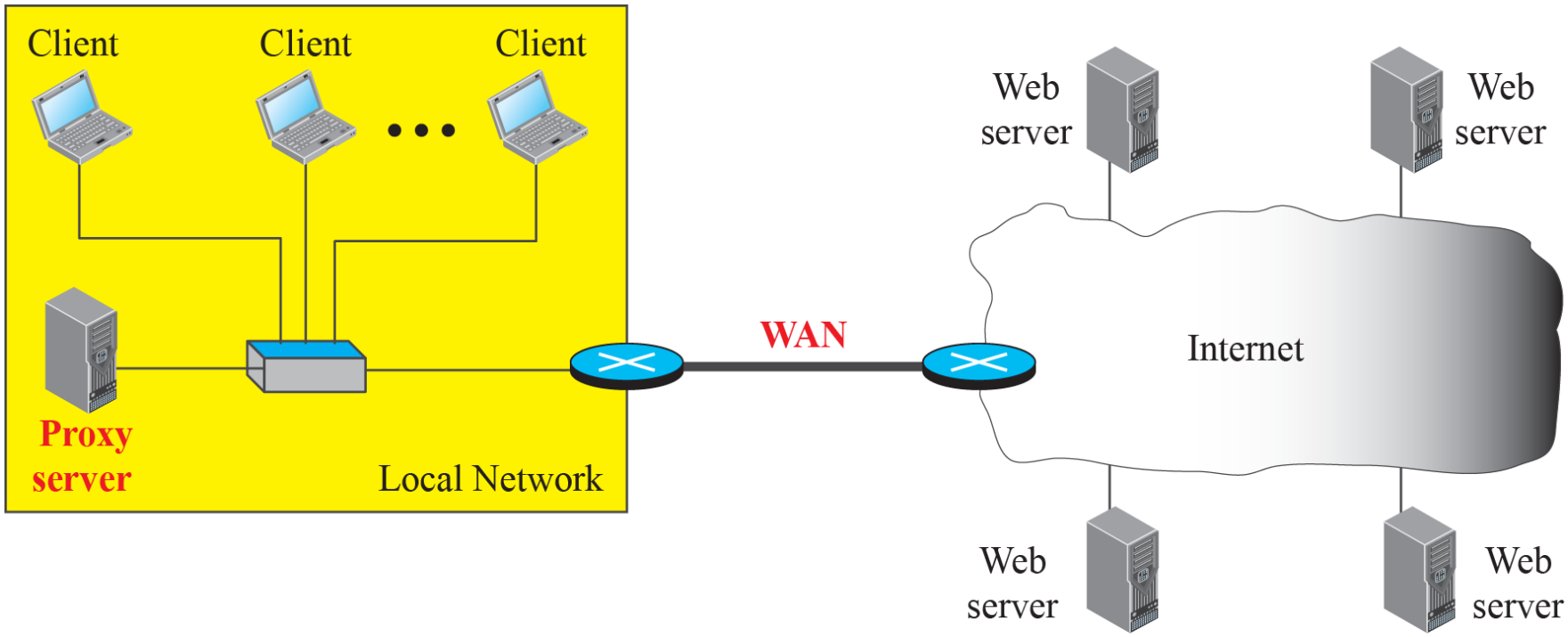
2.3 Standard Client-Server Application
World Wide Web and HTTP: HTTP (19/20)

- **Web Caching: Proxy Server Location**
  - The proxy servers are normally located at the client site
  - This means that we can have a hierarchy of proxy servers as shown below:
    - A client computer can also be used as a proxy server, in a small capacity, that stores responses to requests often invoked by the client
    - In a company, a proxy server may be installed on the computer LAN to reduce the load going out of and coming into the LAN (example in next page)
    - An ISP with many customers can install a proxy server to reduce the load going out of and coming into the ISP network
- **Web Caching: Cache Update**
  - Several different strategies are used for determining when the responses stored in the cache need to be updated
    - Storing the list of sites whose information remains the same for a period of time
    - Adding some new headers to show the last modifications time of the information

2.3 Standard Client-Server Application

World Wide Web and HTTP: HTTP (20/20)

- **Web Caching: Example 2.10**

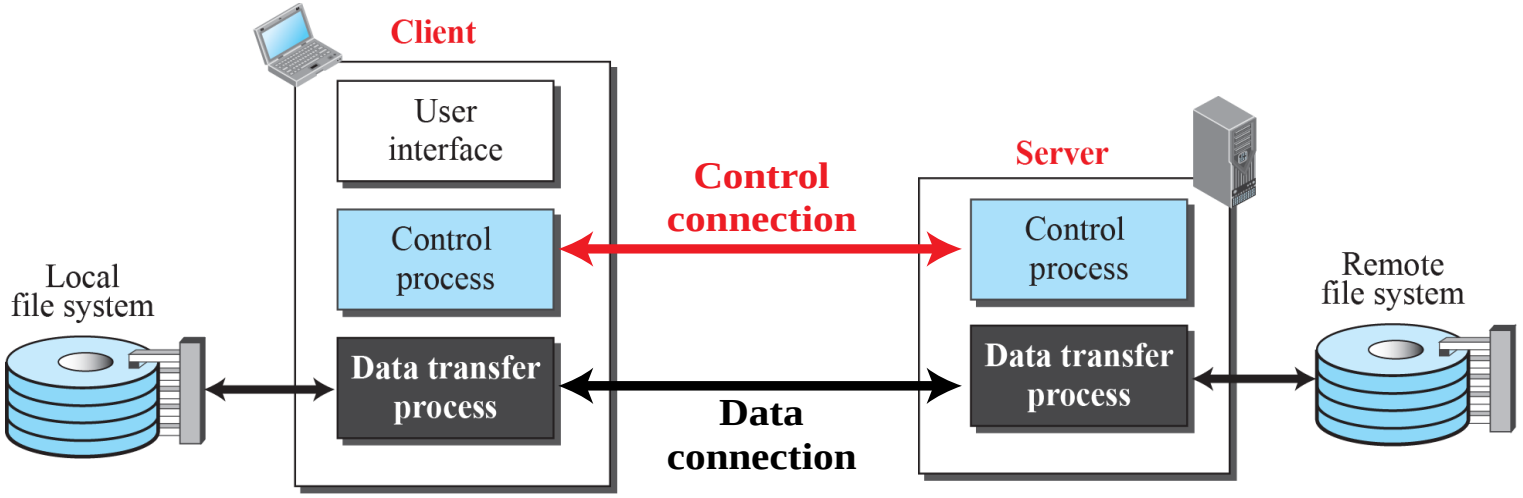


[Figure 2.16 Example of a proxy server]

2.3 Standard Client-Server Application

File Transfer Protocol

- **File Transfer Protocol (FTP)** is the standard protocol provided by TCP/IP for copying a file from one host to another
- Although file transferring can be done using HTTP, FTP is a better choice to transfer large files or to transfer files using different formats
- FTP server has two components, server control process and server data process, along with two kinds of connection, control connection and data connection



[Figure 2.17 FTP]

2.3 Standard Client-Server Application

FTP: Lifetimes of Two Connections

- The two connections in FTP have different lifetimes
- Control connection
  - Being established when a user starts an FTP session
  - Remaining connected during the entire interactive FTP session
- Data connection
  - Being opened each time commands that involve transferring files are used
  - Being closed when the file is transferred

2.3 Standard Client-Server Application

FTP: Control Connection (1/2)

- During the control connection, **commands** are sent from the client to the server and responses are sent from the server to the client

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

[Table 2.4 Some FTP commands]

2.3 Standard Client-Server Application

FTP: Control Connection (2/2)

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

[Table 2.4 Some FTP commands (continued)]

- Every FTP command generates at least one response which has two parts, a code and a description

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

[Table 2.5 Some responses in FTP]

2.3 Standard Client-Server Application

FTP: Data Connection

- The data connection uses the well-known port 20 at the sever site
- However, the creation of a data connection is different from the control connection and the following shows the steps:
  - The client, not the server, issues a passive open using an ephemeral port
  - The client sends this port number to the server using the PORT command
  - The server receives the port number and issues an active open using the well-known port 20 and the received ephemeral port number

2.3 Standard Client-Server Application

FTP: File Transfer (1/3)

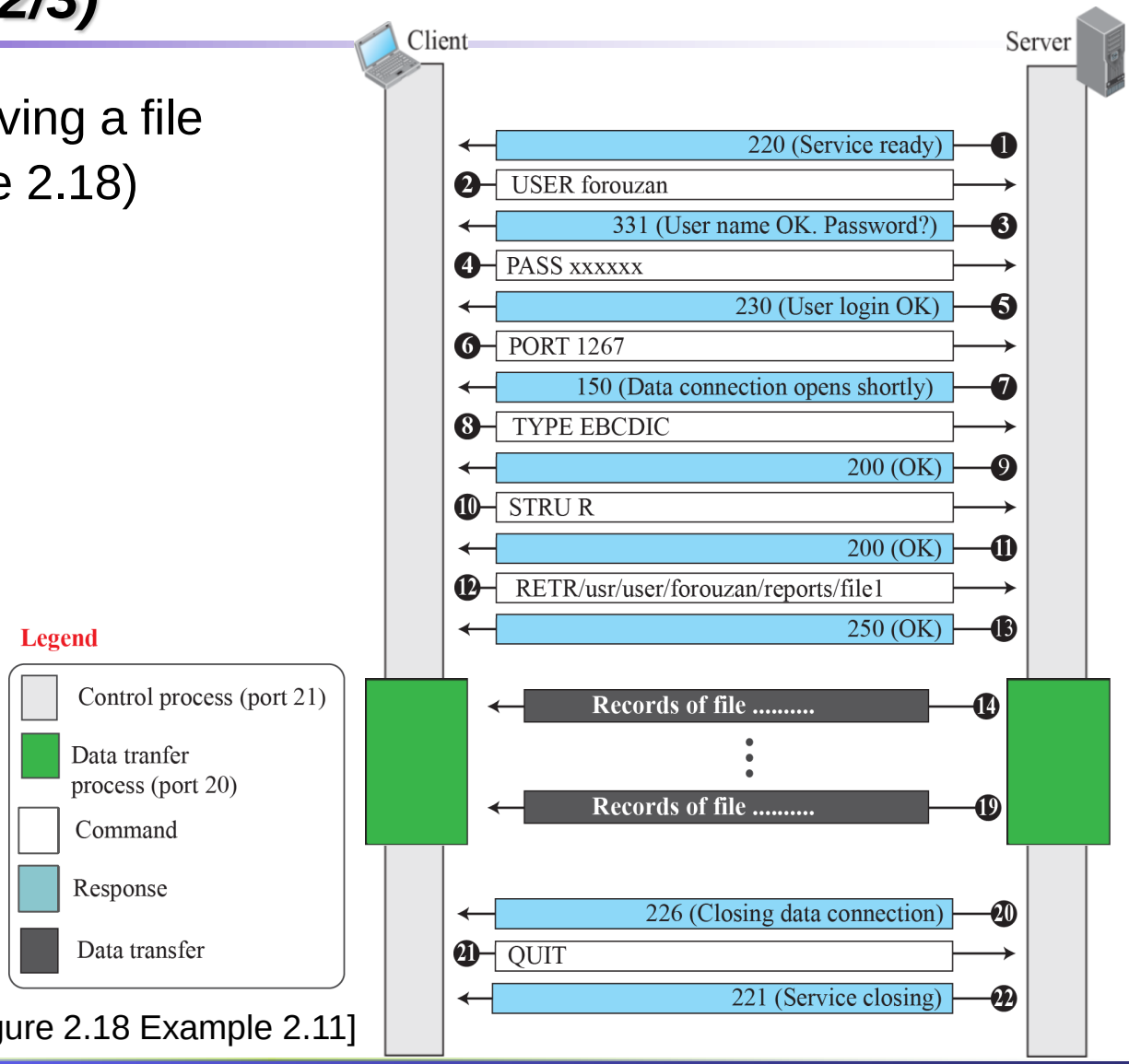
data보내는 data connection과 command를 보내는 control connection이 나눠져 있다.

- File transfer occurs over the data connection under the control of the commands sent over the control connection
- However, we should remember that file transfer in FTP means one of three things:
  - Retrieving a file (server to client)
  - Storing a file (client to server)
  - Directory listing (server to client)

2.3 Standard Client-Server Application

FTP: File Transfer (2/3)

- **Example 2.11**: Retrieving a file using FTP (see Figure 2.18)



[Figure 2.18 Example 2.11]

2.3 Standard Client-Server Application

FTP: File Transfer (3/3)

- **Example 2.12**: 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.
```

[Example 2.12]

Practice Problems

1. What is an HTTP cookie? List some common use of cookie.

To keep track of visitors. To provide better services

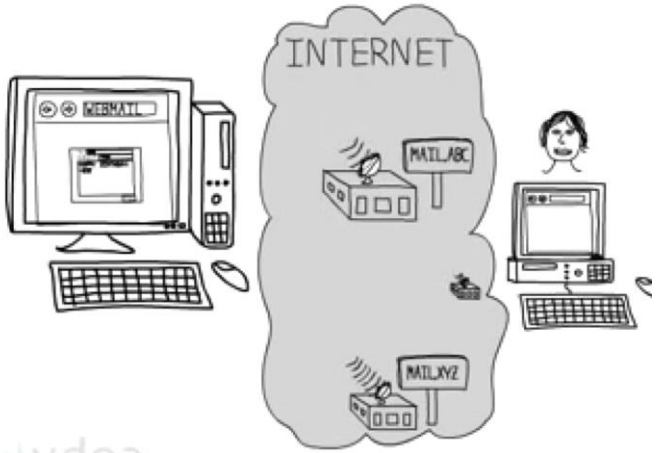
2. Why is it said that FTP sends control information "out of band"?

Control messages는 data와 다른 채널을 사용하기 때문에

2.3 Standard Client-Server Application

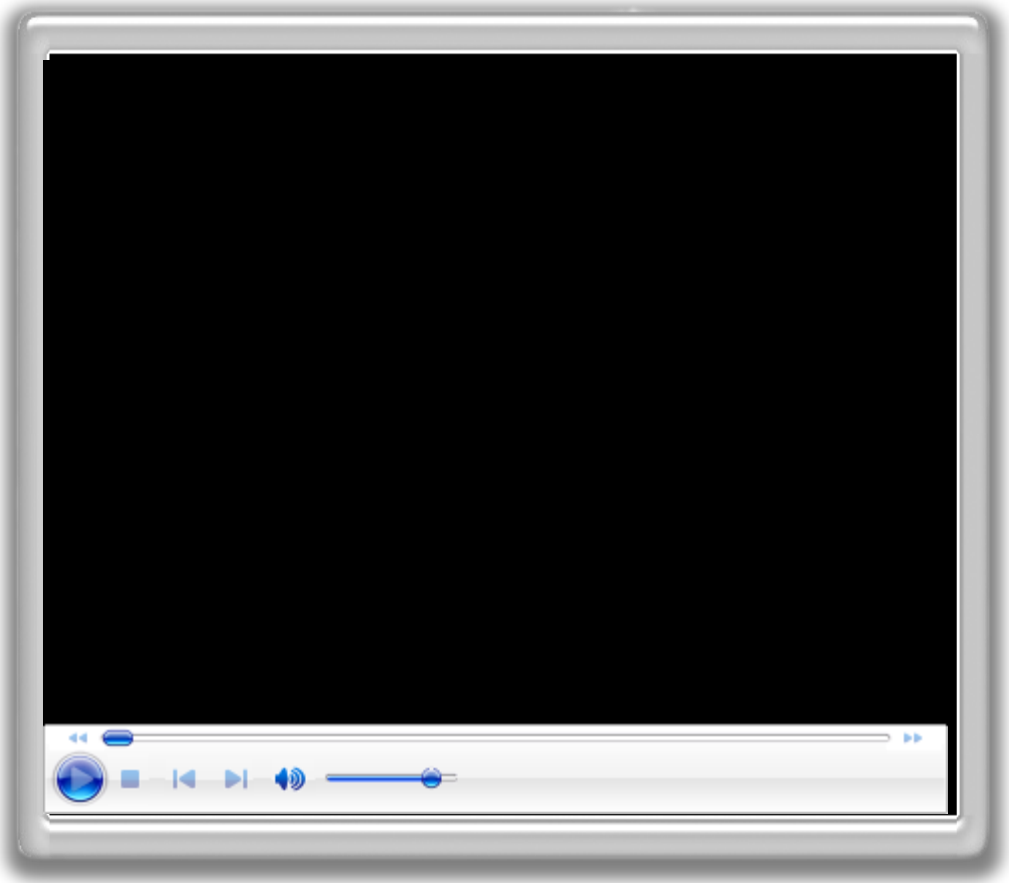
Electronic Mail: Introduction (1/2)

- Video Content
  - How an email is handled on the Internet? Where an email travels?
  - How private/secure email is?
  - Link: https://www.youtube.com/watch?v=YBzLPmx3xTU



2.3 Standard Client-Server Application

Electronic Mail: Introduction (2/2)



2.3 Standard Client-Server Application

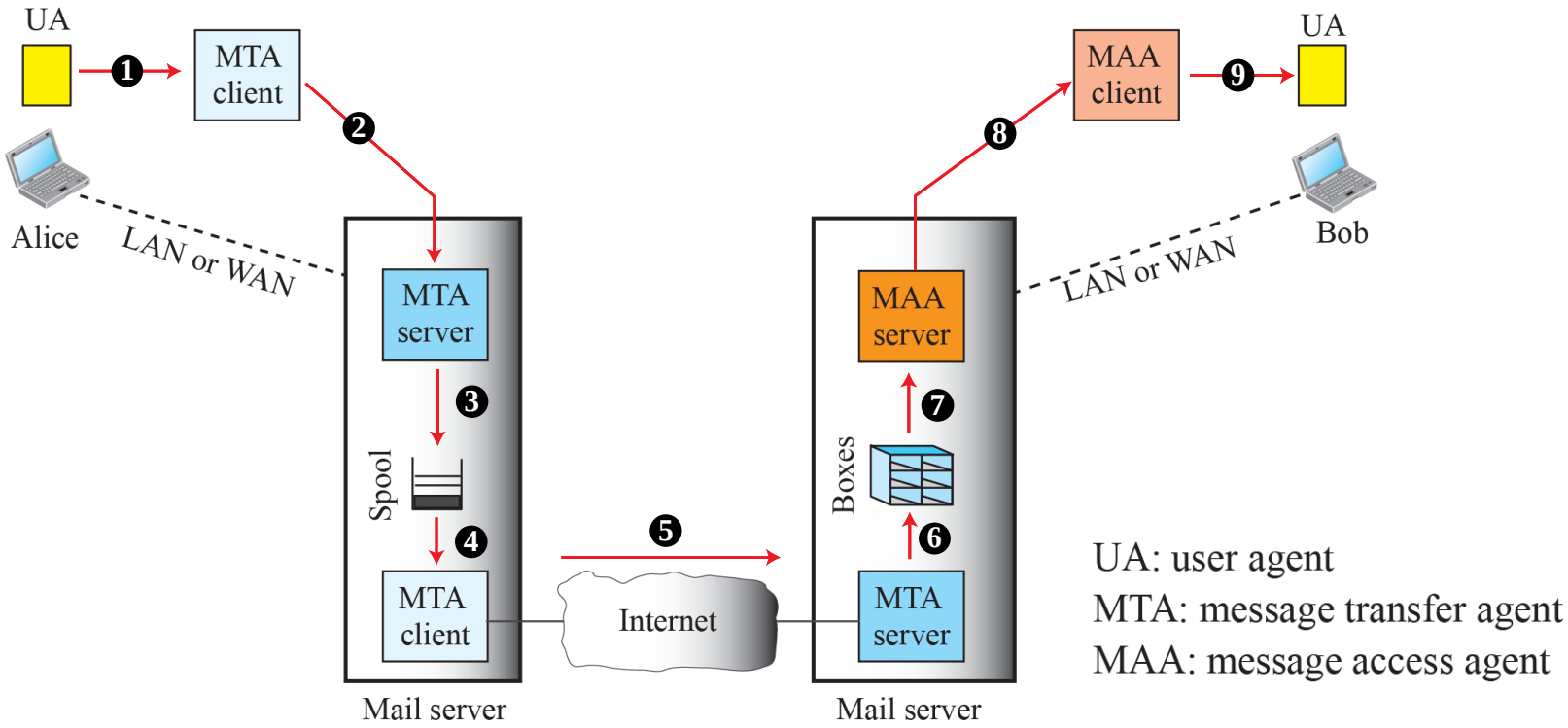
Electronic Mail

- **Electronic mail** (or e-mail) allows users to exchange messages
- The nature of this application is different from other applications such as HTTP or FTP
  - E-mail is considered a one-way transaction
  - Two users at two ends exchange messages through some intermediate servers
  - The users run only client programs when they want and the intermediate servers apply the client/server paradigm

2.3 Standard Client-Server Application

Electronic Mail: Architecture

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



[Figure 2.19 Common scenario]

2.3 Standard Client-Server Application

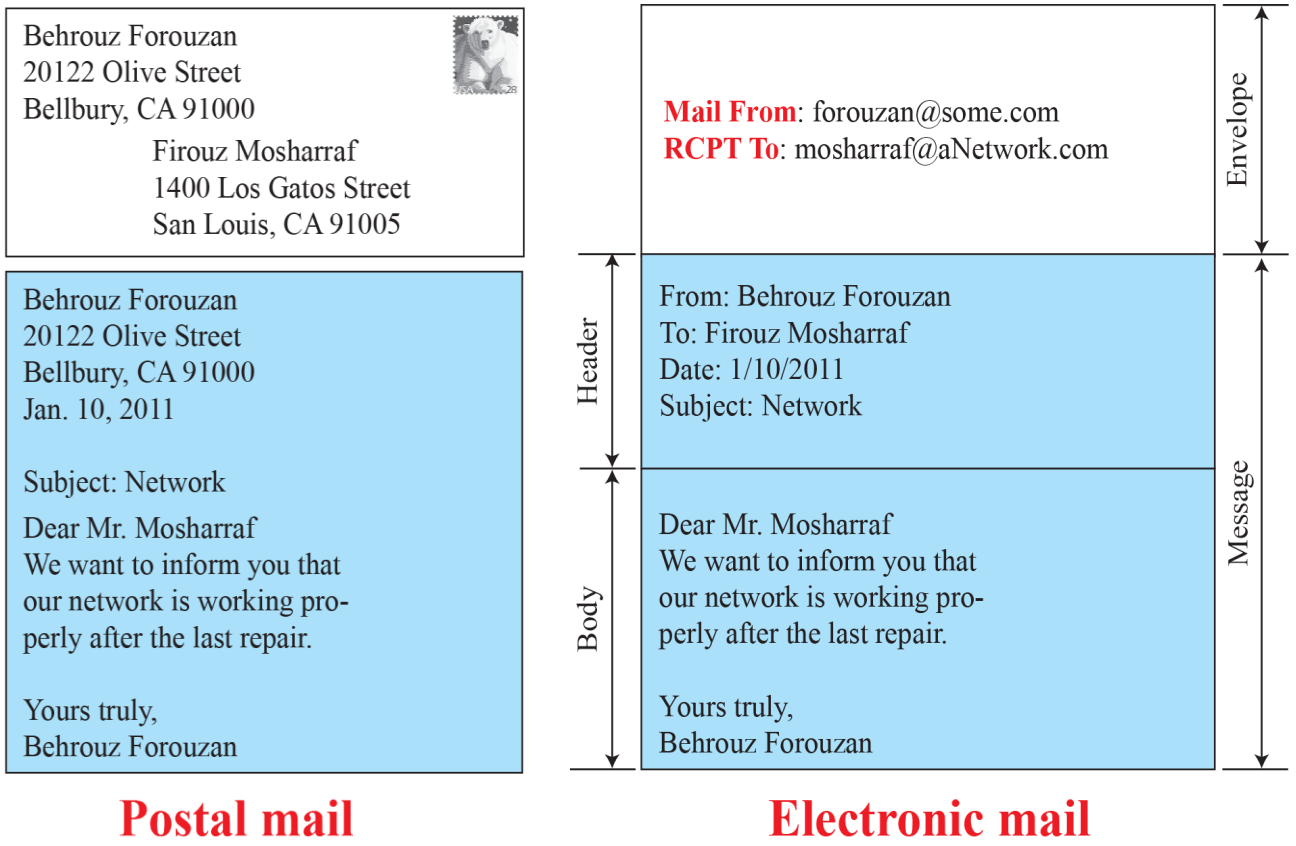
Electronic Mail: User Agent (1/3)

- UA provides service to the user to make the process of sending and receiving a message easier
- A UA is a program that composes, reads, replies to, forwards messages, and also handles local mail boxes on the user computers
- There are two types of UA: command-driven UA and GUI-based UA
  - Command-driven UA: *mail*, *pine*, and *elm*
  - GUI-based UA: *Eudora* and *Outlook*

2.3 Standard Client-Server Application

Electronic Mail: User Agent (2/3)

- To send mail, the user, through the UA, first creates the mail that looks very similar to postal mail



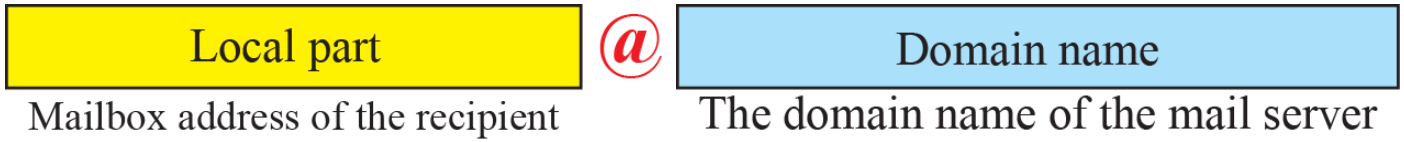
[Figure 2.20 Format of an e-mail]

2.3 Standard Client-Server Application

Electronic Mail: User Agent (3/3)

■ E-mail Address

- To deliver mail, a mail handling system must use an addressing system with unique addresses
- In the Internet, the address consists of two part: a local part and a domain name, separated by an @ sign



[Figure 2.21 E-mail address]

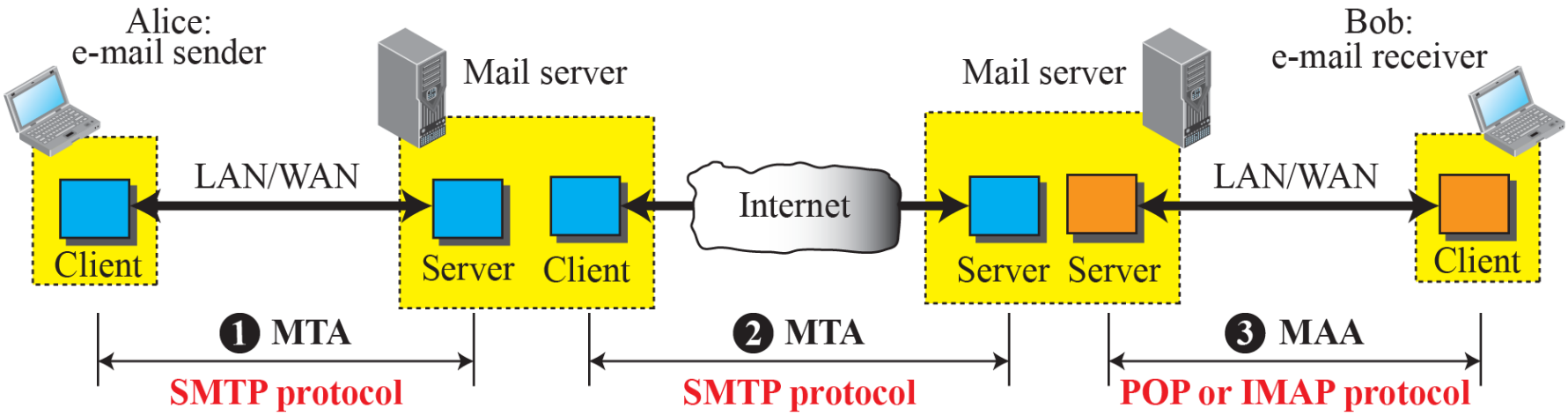
■ Mailing List or Group List

- E-mail allows one name, an alias, to represent several different e-mail addresses, called a mailing list
- Based on the mailing list, separate copies of the message that needs to be sent, one for each entry in the list, must be prepared and handed to the MTA

2.3 Standard Client-Server Application

Electronic Mail: Message Transfer Agent – SMTP (1/6)

- E-mail can be seen to be composed of three client-server applications
  - Two Message Transfer Agents (MTAs)
  - Message Access Agent (MAA)
- Protocols are used
  - MTA client and MTA server: Simple Mail Transfer Protocol (SMTP)
  - MAA client and MAA server: POP or IMAP



[Figure 2.22 Protocols used in electronic email]

2.3 Standard Client-Server Application

Electronic Mail: Message Transfer Agent – SMTP (2/6)

■ Commands and Responses (1/3)

- 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 and the response is from an MTA server to the MTA client
- **Commands**: are sent from the client to the server, listed in following table

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

[Table 2.6 SMTP commands]

2.3 Standard Client-Server Application

Electronic Mail: Message Transfer Agent – SMTP (3/6)

■ Commands and Responses (2/3)

Keyword	*Argument(s)*	*Description*
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

[Table 2.6 SMTP Commands (continued)]

2.3 Standard Client-Server Application

Electronic Mail: Message Transfer Agent – SMTP (4/6)

- **Commands and Responses (3/3)**
  - 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
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

[Table 2.7 Responses]

2.3 Standard Client-Server Application

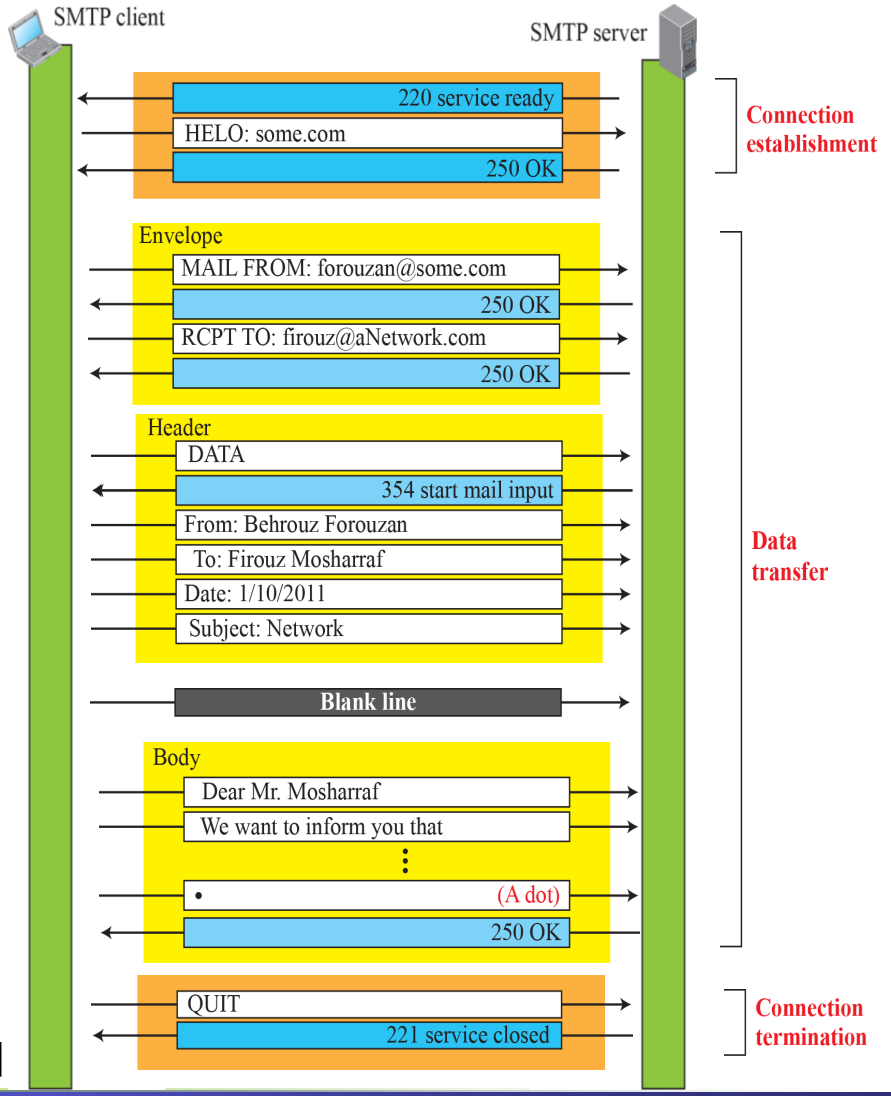
Electronic Mail: Message Transfer Agent – SMTP (5/6)

- **Mail transfer phases:** the process of transferring a mail message occurs in three phases:
  - **Connection establishment**: after a client has made a TCP connection to the well known port 25, the SMTP server starts the connection phase
  - **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
  - **Connection termination**: after the message is transferred successfully, the client terminates the connection

2.3 Standard Client-Server Application

Electronic Mail: Message Transfer Agent – SMTP (6/6)

- **Example 2.13:** Three mail transfer phases
  - All steps for transferring an email through three phases

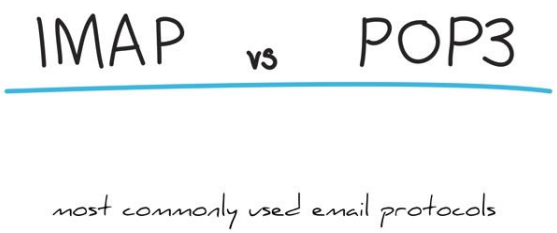


[Figure 2.23: Example 2.13]

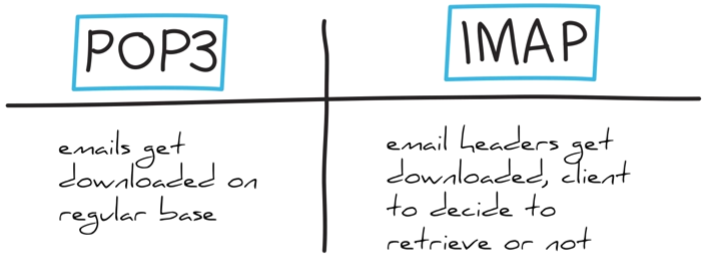
2.3 Standard Client-Server Application

Electronic Mail: POP and IMAP (1/4)

- Video Content
  - An explanation of the two email-retrieval protocols POP and IMAP
  - How to select one for our services?
  - Link: https://www.youtube.com/watch?v=z5epBMEVUpk



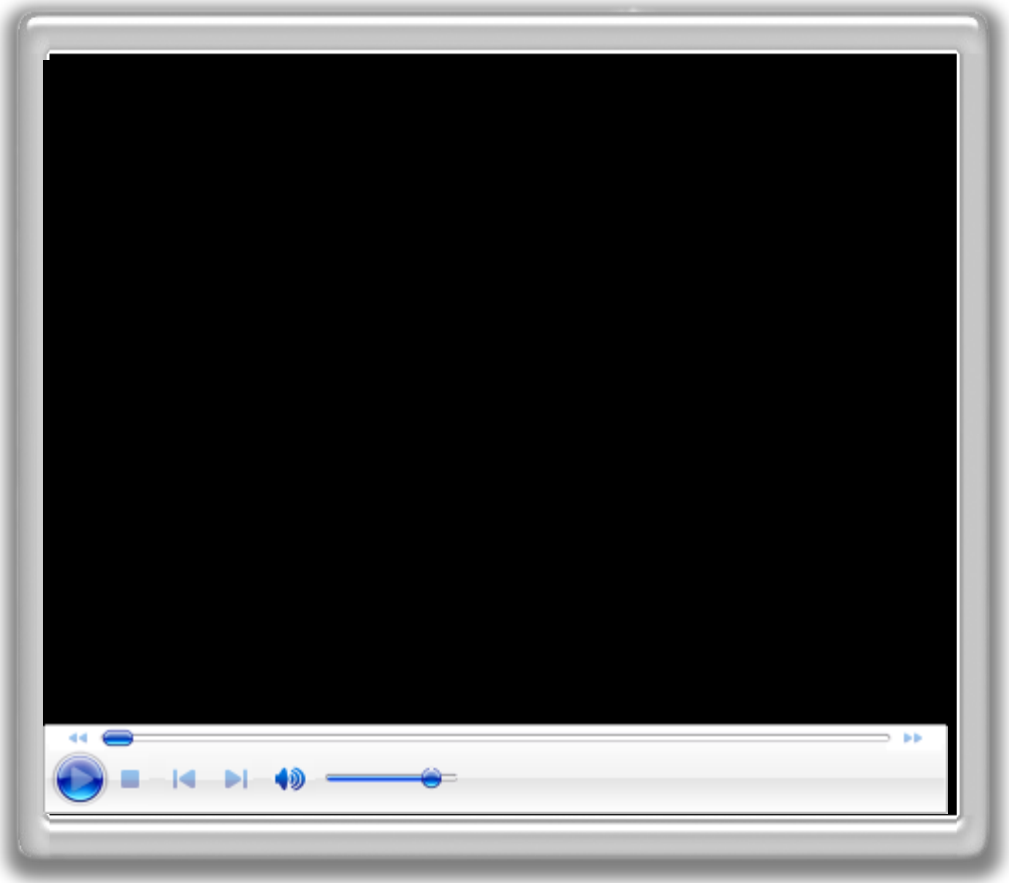
하나의 DEVICE에서만 접속



모바일 웹 PC등 여러 DEVICE에서 접속

2.3 Standard Client-Server Application

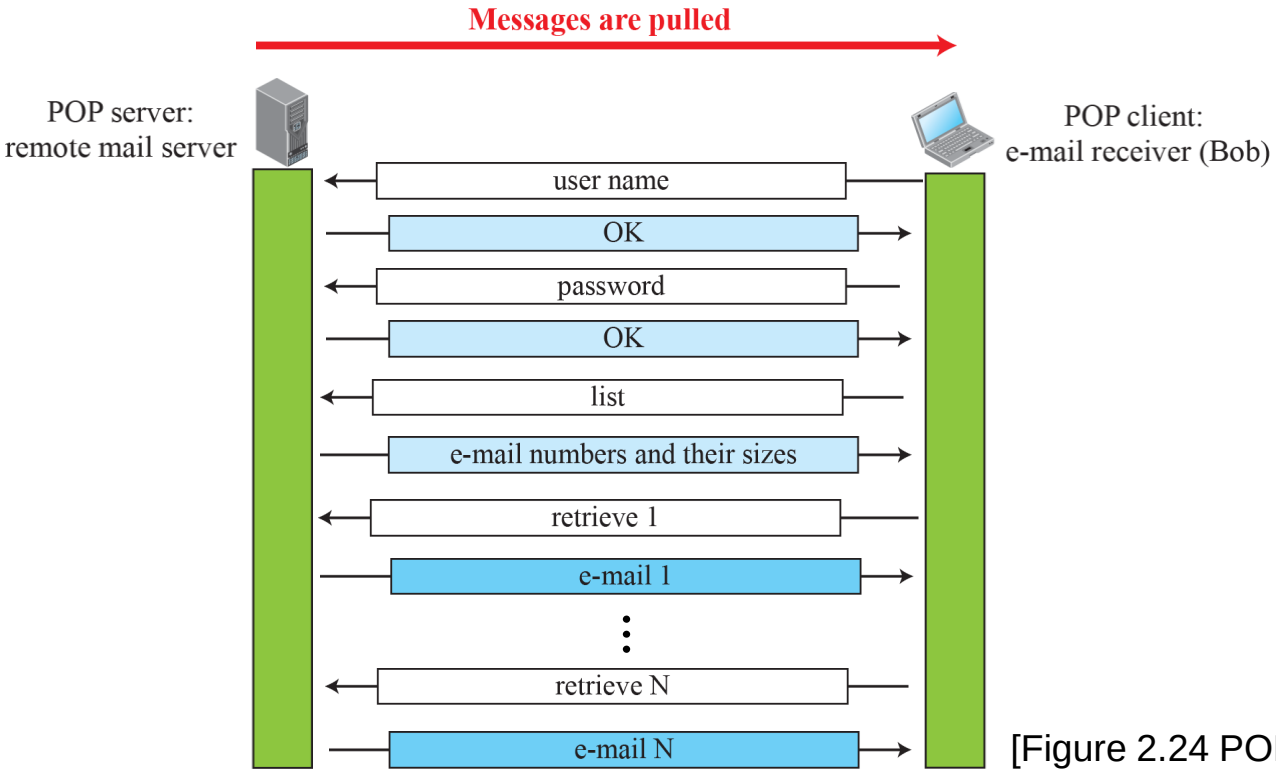
Electronic Mail: POP and IMAP (2/4)



2.3 Standard Client-Server Application

Electronic Mail: POP and IMAP (3/4)

- **Post Office Protocol, version 3 (POP3)**
  - This protocol is simple but limited in functionality
  - The client POP3 software is installed on the recipient computer and the server POP3 software is installed on the mail server



[Figure 2.24 POP3]

2.3 Standard Client-Server Application

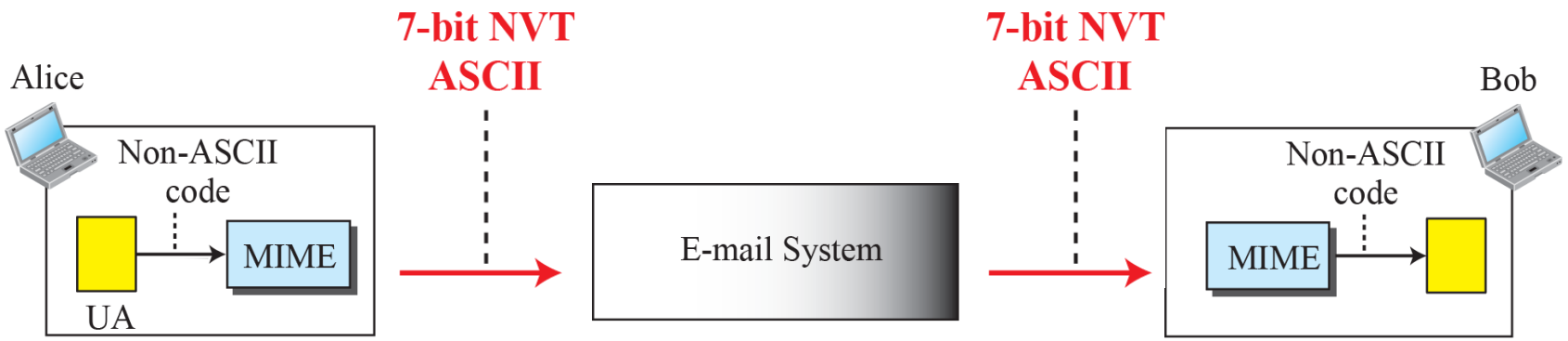
Electronic Mail: POP and IMAP (4/4)

- **Internet Mail Access Protocol, version 4 (IMAP4)**
  - This protocol is similar to POP3, but it has more features, powerful, and complex
  - IMAP 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 and it 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

2.3 Standard Client-Server Application

Electronic Mail: MIME (1/7)

- **Multipurpose Internet Mail Extension (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 **Network Virtual Terminal** (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



[Figure 2.25 MIME]

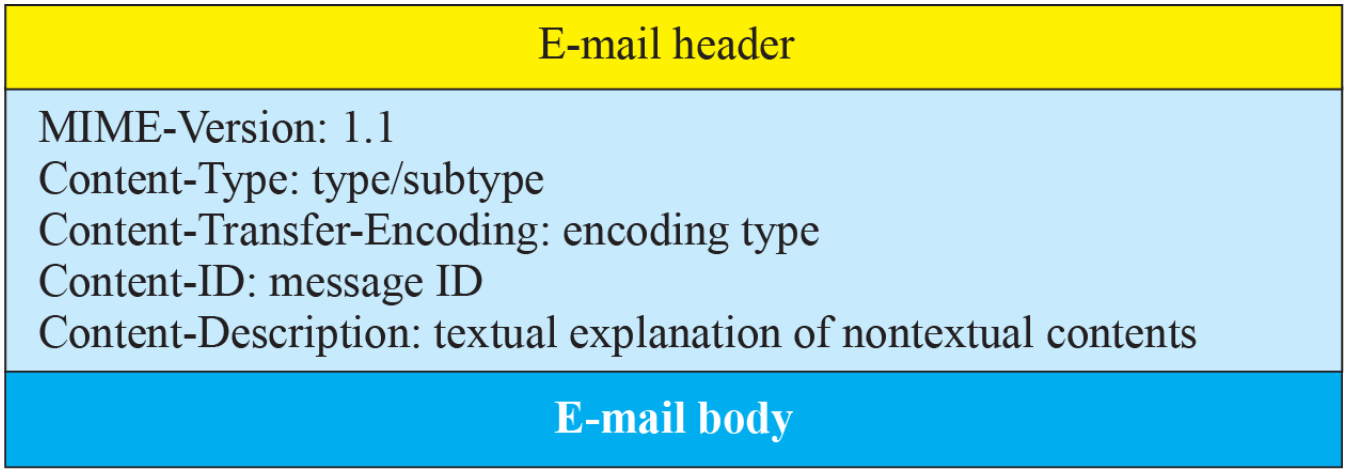
2.3 Standard Client-Server Application

Electronic Mail: MIME (2/7)

■ MIME headers (1/3)

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

MIME headers



[Figure 2.26 MIME header]

2.3 Standard Client-Server Application

Electronic Mail: MIME (3/7)

- **MIME headers (2/3)**
  - **Content-type:** MIME allows 7 different types of data, listed in Table 2.8

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 2.8 Data Types and Subtypes in MIME]

2.3 Standard Client-Server Application

Electronic Mail: MIME (4/7)

■ MIME headers (3/3)

- **Content-transfer-encoding:** defines the method used to encode the message into 0s and 1s for transport
  - ★ The five types of encoding methods are listed in Table 2.9

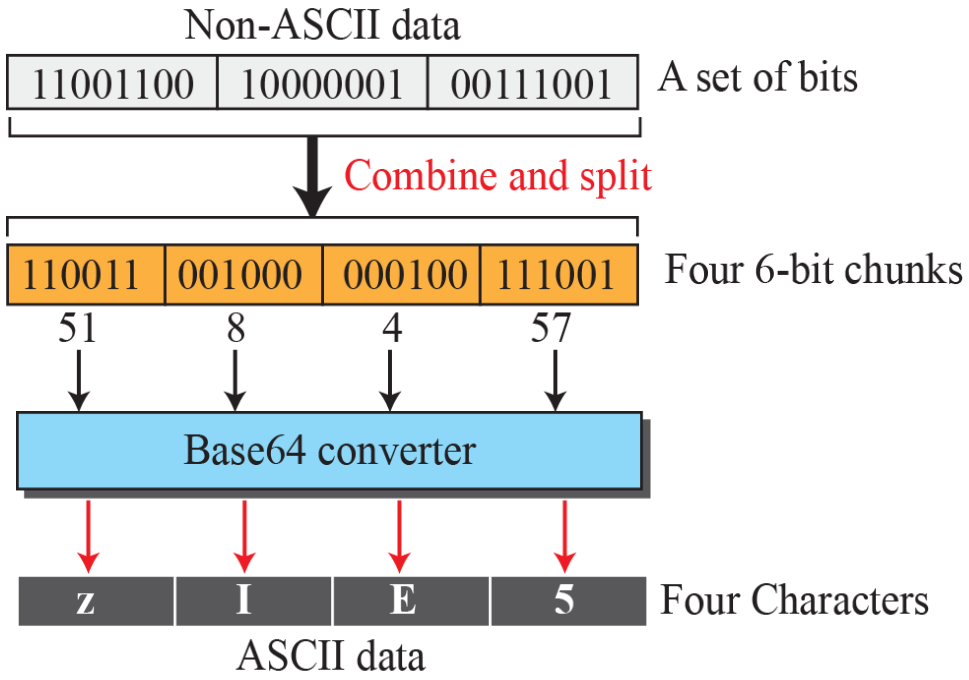
Type	*Description*
7-bit	NVT ASCII characters with each line less than 1000 characters
8-bit	Non-ASCII characters with each line less than 1000 characters
Binary	Non-ASCII characters with unlimited-length lines
Base64	6-bit blocks of data encoded into 8-bit ASCII characters
Quoted-printable	Non-ASCII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

[Table 2.9 Methods for Content-Transfer-Encoding]

2.3 Standard Client-Server Application

Electronic Mail: MIME (5/7)

- **Base64 encoding (1/2)**
  - Data, as a string of bits, is first divided into 6-bit chunks as follows



[Figure 2.27 Base64 conversion]

2.3 Standard Client-Server Application

Electronic Mail: MIME (6/7)

■ Base64 encoding (2/2)

- Each 6-bit section is then converted into an ASCII character according to Table 2.10

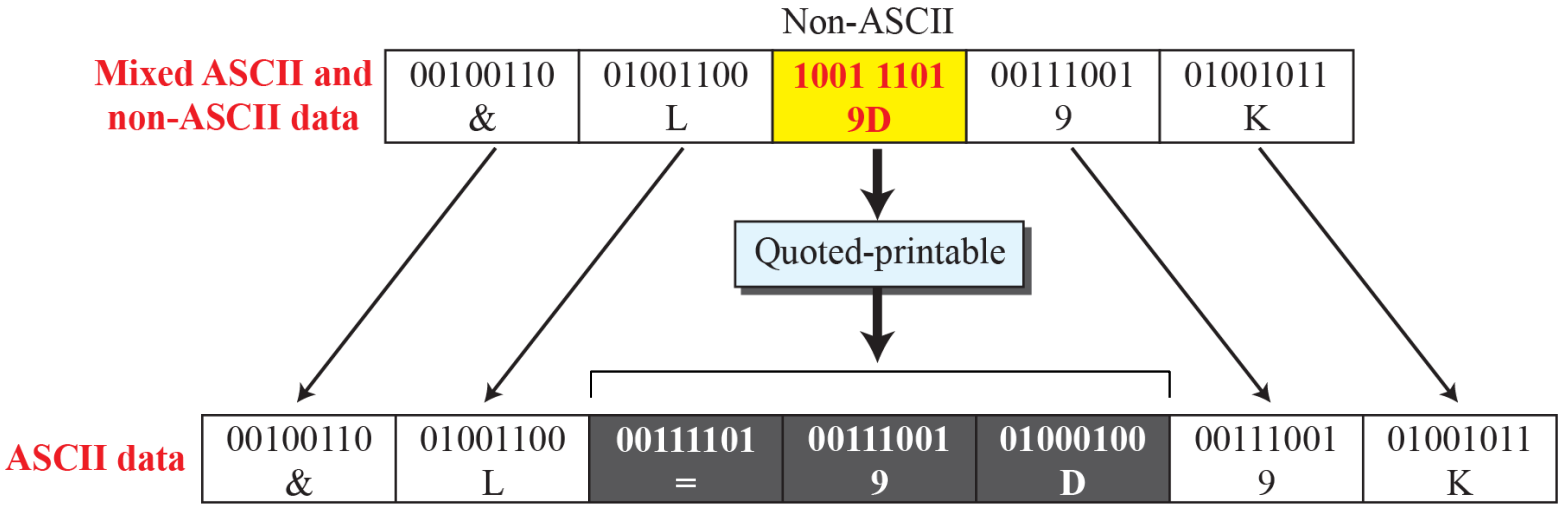
Value	*Code*	*Value*	*Code*	*Value*	*Code*	*Value*	*Code*	*Value*	*Code*	*Value*	*Code*
0	**A**	11	**L**	22	**W**	33	**h**	44	**s**	55	**3**
1	**B**	12	**M**	23	**X**	34	**i**	45	**t**	56	**4**
2	**C**	13	**N**	24	**Y**	35	**j**	46	**u**	57	**5**
3	**D**	14	**O**	25	**Z**	36	**k**	47	**v**	58	**6**
4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

[Table 2.10 Base64 Converting Table]

2.3 Standard Client-Server Application

Electronic Mail: MIME (7/7)

- **Quoted-printable encoding**
  - If a character is ASCII, it is sent as is
  - Otherwise, it is sent as three characters including the equal sign (=), and two last characters in the hexadecimal representation

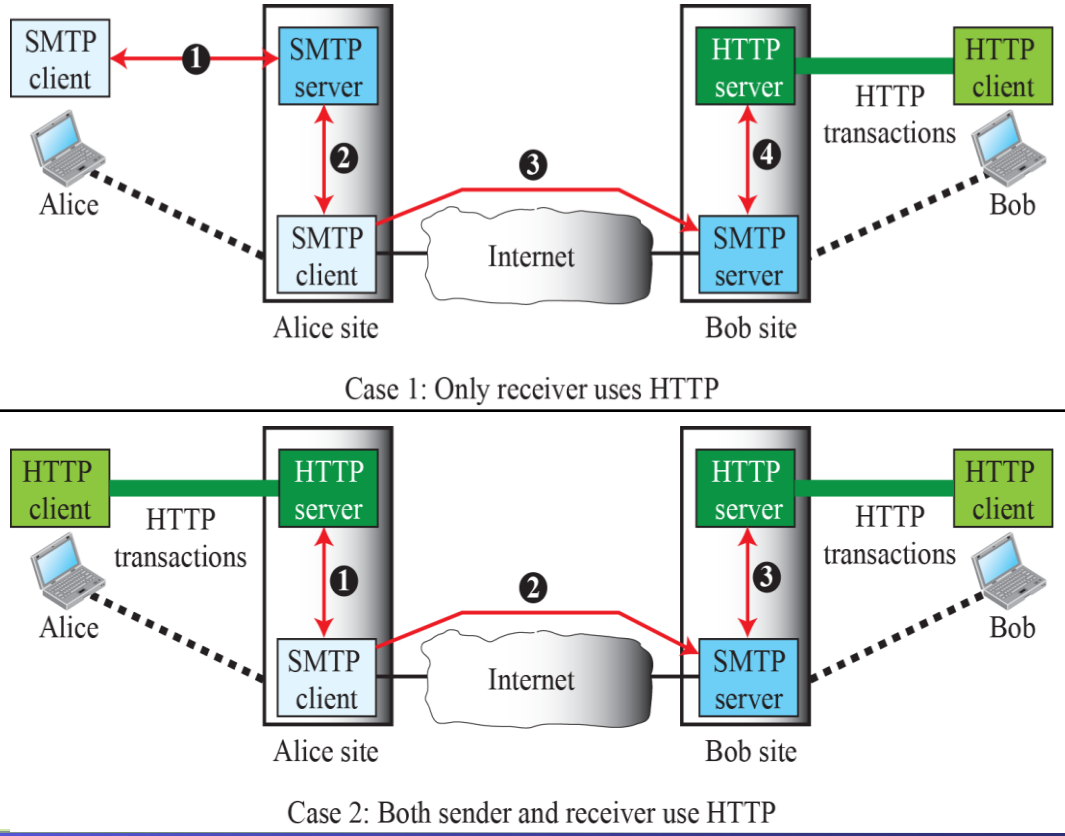


[Figure 2.28 Quoted-printable]

2.3 Standard Client-Server Application

Electronic Mail: 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



[Figure 2.29 Web-based e-mail]

Practice Problems

1. Provide the full name of the acronyms: SMTP, POP3, IMAP.

   Simple Message Transfer Protocol
   Post Office Protocol version 3
   Internet Message Access Protocol

2. Consider SMTP, POP3 and IMAP. Are these stateless protocols? Why or why not?

   All three are stateful protocols
   They maintain store the past information for further uses
   HTTP는 stateless

2.3 Standard Client-Server Application

TELNET

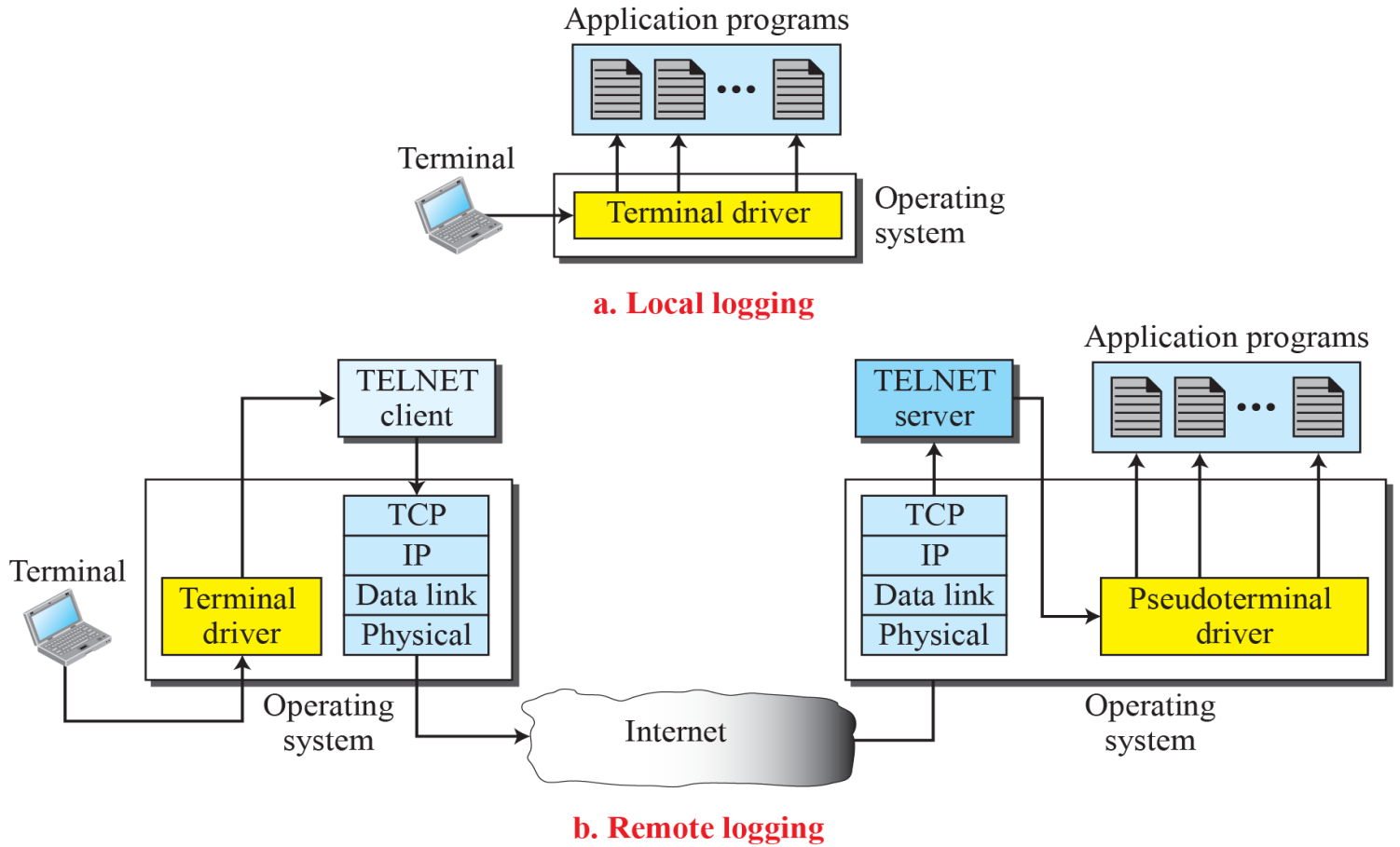
- Besides having a specific client/server program for a set of common scenarios, we also have a demand of some generic client/server programs that allow a user on the client site to log into the computer at the server site and use the services available there
  - These generic client/server pairs are referred to as remote logging applications

- Remote logging protocols
  - **TELNET** (TErminaL NETwork): no secure mechanism for data exchanging
  - **Secure SHell** (SSH): data exchanging securely

2.3 Standard Client-Server Application

TELNET: Local versus remote logging

- The concept of local and remote logging

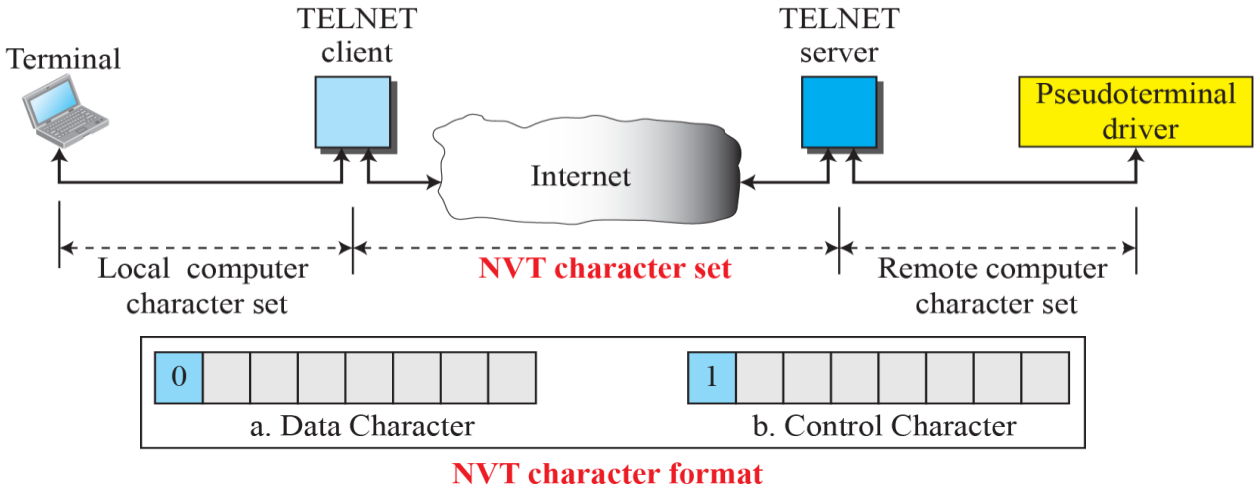


[Figure 2.30 Local versus remote logging]

2.3 Standard Client-Server Application

TELNET: Network Virtual Terminal (NVT)

- Different computer systems uses different mechanisms of character representation
  - To access a particular remote computer, we must first know what type of computer it is, and we must also install the specific terminal emulator used by that computer
- TELNET solves this problem by defining a universal interface called **Network Virtual Terminal** (NVT)



[Figure 2.31 Concept of NVT]

2.3 Standard Client-Server Application

TELNET: User Interface

- The operating system (UNIX, for example) defines an interface with user-friendly commands
- An example of such a set of commands can be found in Table 2.11

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

[Table 2.11 Examples of interface commands]

2.3 Standard Client-Server Application

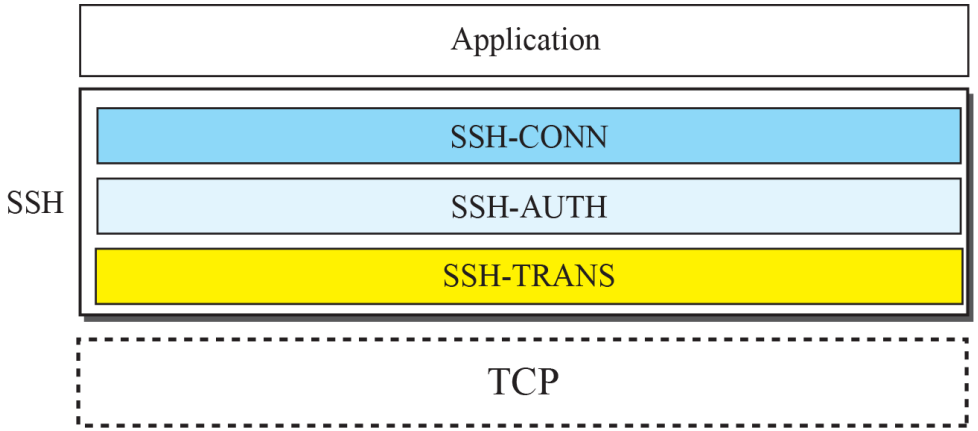
Secure Shell (SSH)

- SSH is a secure application program that can be used today for several purposes such as remote logging and file transfer
- 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 its security flaws

2.3 Standard Client-Server Application

Secure Shell (SSH): Components (1/2)

- SSH is an application-layer protocol with three components
  - **SSH Transport-Layer Protocol (SSH-TRANS)**
    - Privacy or confidentiality of the message exchanged
    - Data integrity: it is guaranteed that the messages exchanged between the client and server are not changed by an intruder
    - Server authentication: the client is now sure that the server is the one that it claims to be
    - Compression of the messages: this improves the efficiency of the system and makes attack more difficult

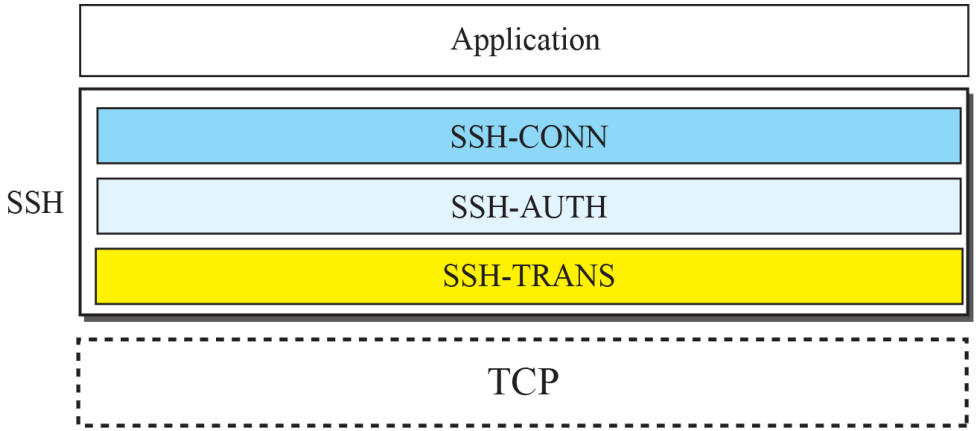


[Figure 2.32 Components of SSH]

2.3 Standard Client-Server Application

Secure Shell (SSH): Components (2/2)

- **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
- **SSH Connection Protocol (SSH-CONN)**
  - After the secured channel is established and both server and client are authenticated, SSH can call a piece of software implementing SSHCONN
  - Client can create multiple logical channels over the secured channel for different purposes, such as remote logging, file transfer, and so on



[Figure 2.32 Components of SSH]

2.3 Standard Client-Server Application

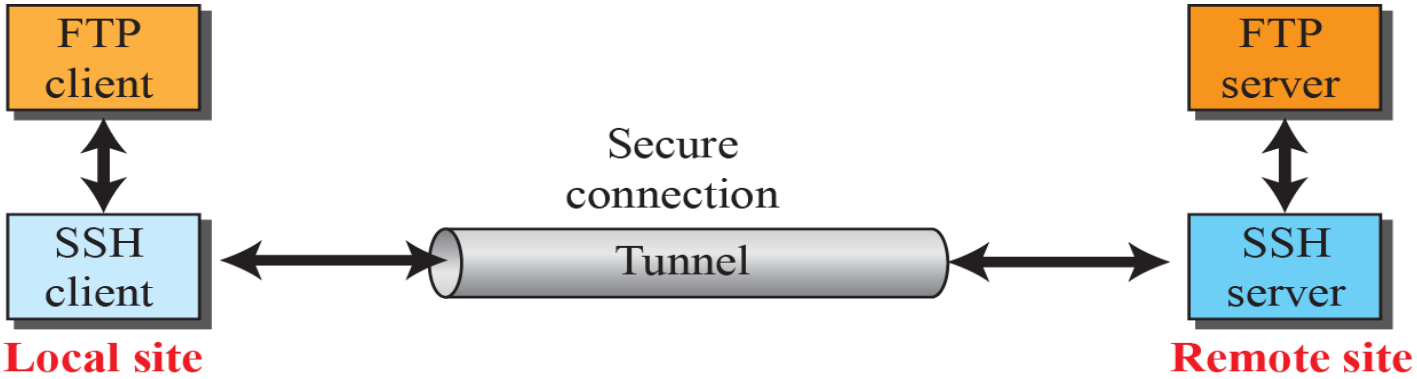
Secure Shell (SSH): Applications

- SSH is a general-purpose protocol that provides a secure connection between a client and a server
- **SSH for Remote Logging**: several free and commercial applications use SSH for remote logging, including *PuTTy* and *Tectia*
- **SSH for File Transfer**: application programs built on top of SSH for file transfer includes *Secure File Transfer Program* (*sftp*) and *Secure Copy* (*scp*)

2.3 Standard Client-Server Application

Secure Shell (SSH): Port Forwarding

- **Port Forwarding:** a service provided by the SSH protocol
- We can use the secured channels available in SSH to access an application program that does not provide security services
  - TELNET, FTP, and Simple Mail Transfer Protocol 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

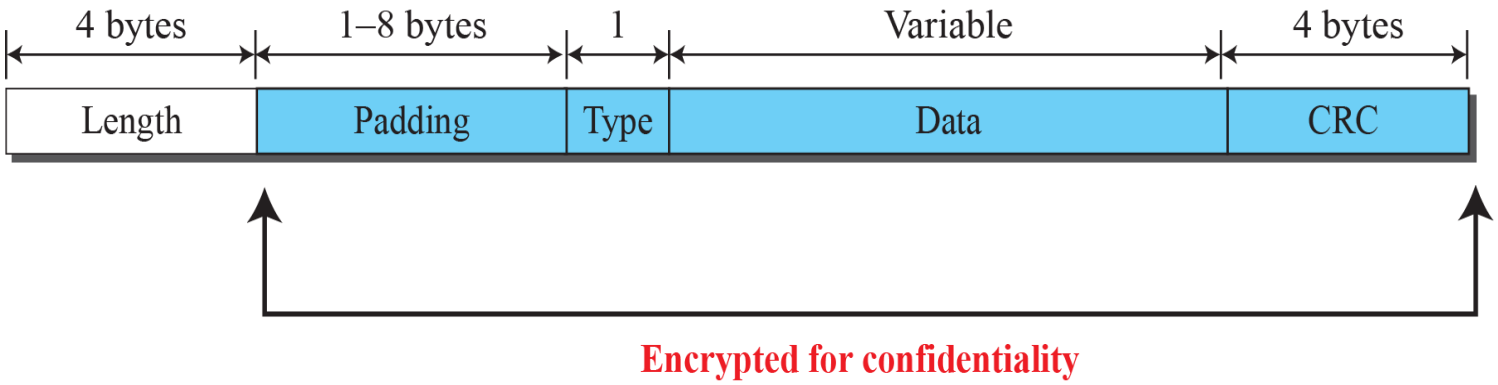


[Figure 2.33 Port Forwarding]

2.3 Standard Client-Server Application

Secure Shell (SSH): Format of the SSH Packets

- Data Fields
  - **Length**: the length of the packet, not include the padding
  - **Padding**: one to eight bytes is added to the packet to make the attack on the security provision more difficult
  - **Cyclic redundancy check (CRC)**: data field for error detection
  - **Type**: type of the packet used in different SSH protocols.
  - **Data**: data transferred by the packet in different protocols

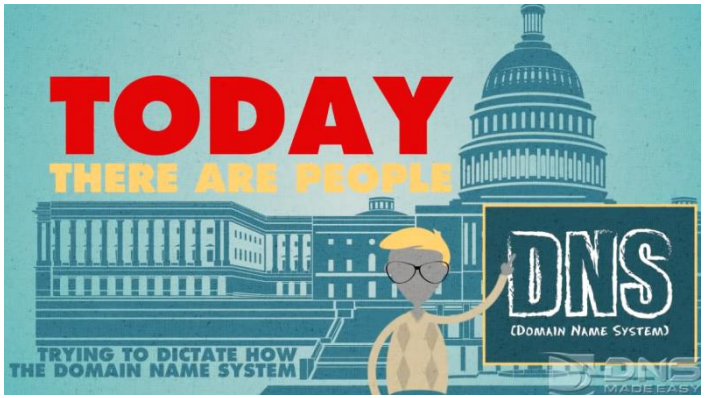


[Figure 2.34 SSH Packet Format]

2.3 Standard Client-Server Application

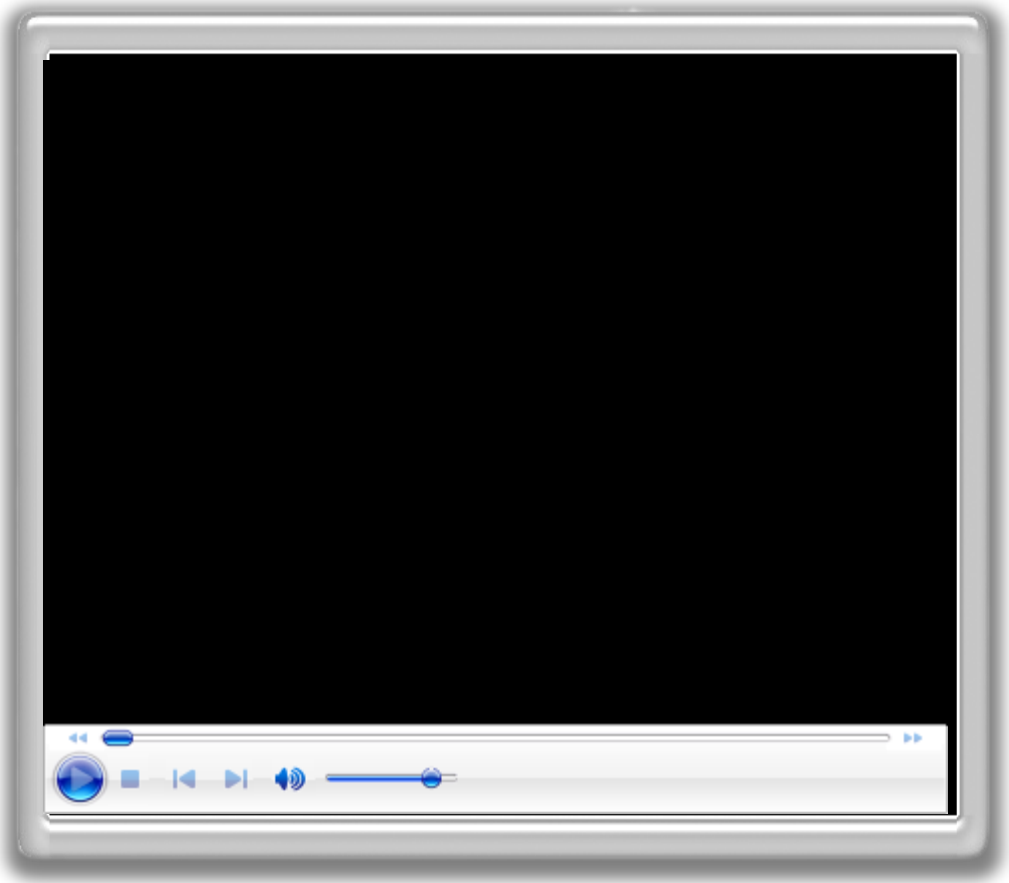
Domain Name System (DNS): Introduction (1/2)

- Video Content
  - An explanation of what DNS is, how it works.
  - The hierarchy of the DNS
  - The process of resolving domain name to IP address using DNS
  - Link: https://www.youtube.com/watch?v=72snZctFFtA



2.3 Standard Client-Server Application

DNS: Introduction (2/4)



2.3 Standard Client-Server Application

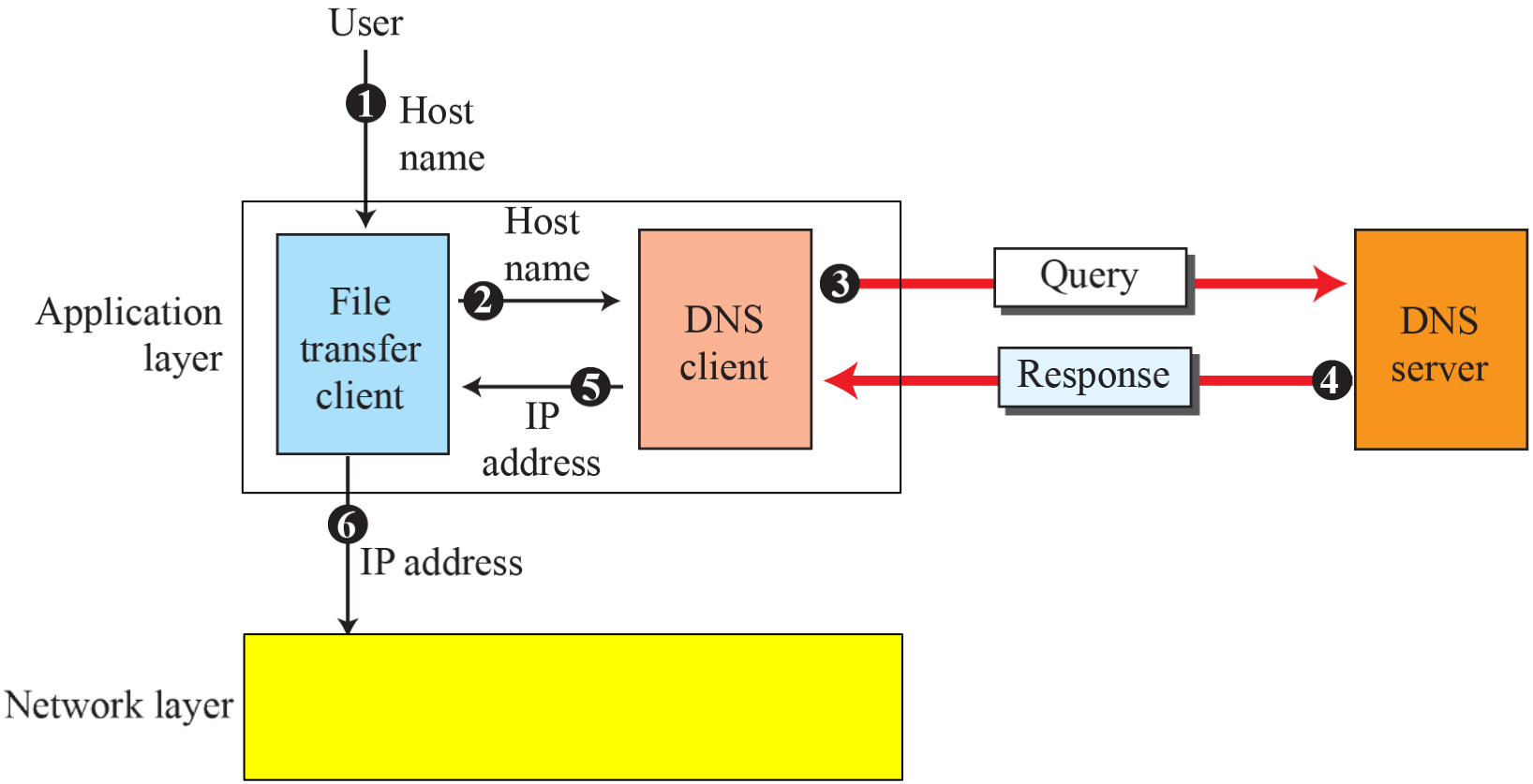
DNS: Introduction (3/4)

- In TCP/IP, IP address is used to identify a host on the Internet
- There is a directory system that maps a name to an IP address of a host for people to access the Internet easily
- Since the Internet is so huge today, a central directory system cannot hold all the mapping; so a better solution is to distribute the information among many computers in the world
  - The host that needs mapping can contact the closest computer holding the needed information
- **Domain Name System (DNS)** utilizes the above method of information distribution

2.3 Standard Client-Server Application

DNS: Introduction (4/4)

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



[Figure 2.35 Purpose of DNS]

2.3 Standard Client-Server Application

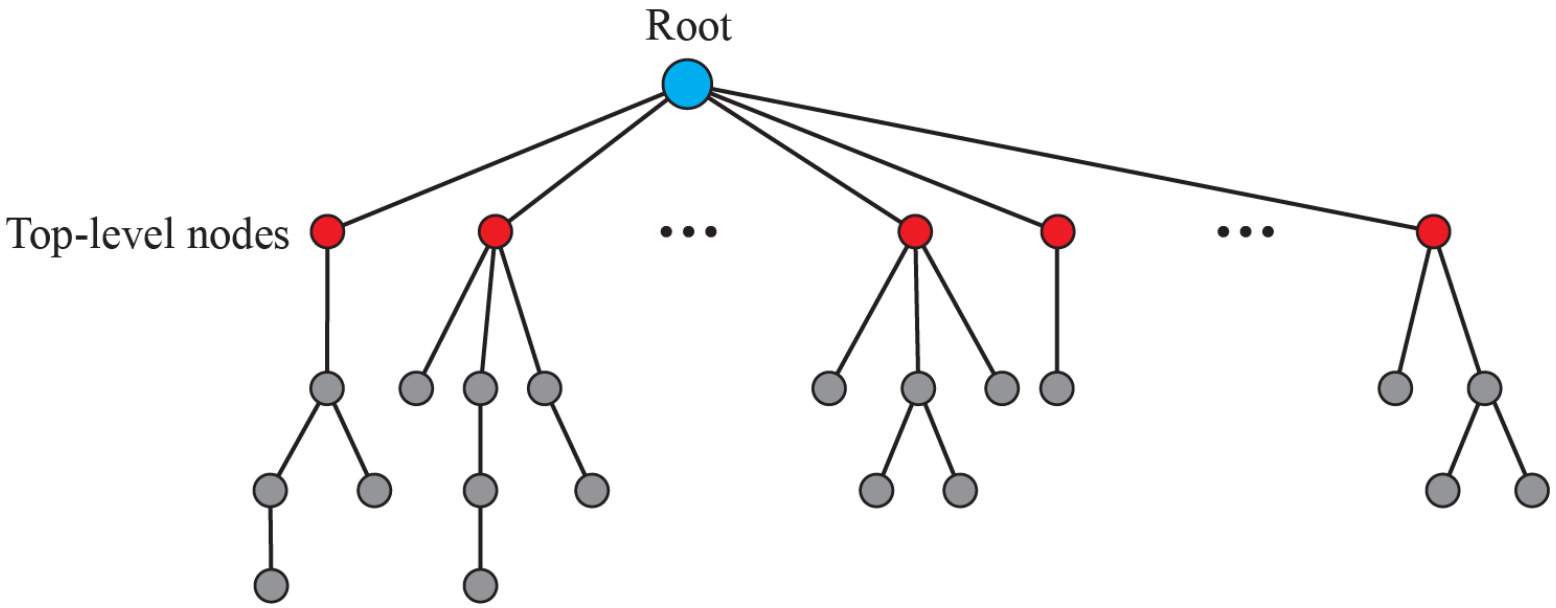
DNS: Name Space (1/2)

- A *name space* that maps each address to a unique name can be organized in two ways: **flat** or **hierarchical**
  - **Flat name space:** a name, with no structure, is assigned to an address
    - 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 representing for organization, department, and so on
    - The authority to assign and control the name spaces can be decentralized

2.3 Standard Client-Server Application

DNS: Name Space (2/2)

- **Domain Name Space:** is designed to support a hierarchical name space
  - 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



[Figure 2.36 Domain name space]

2.3 Standard Client-Server Application

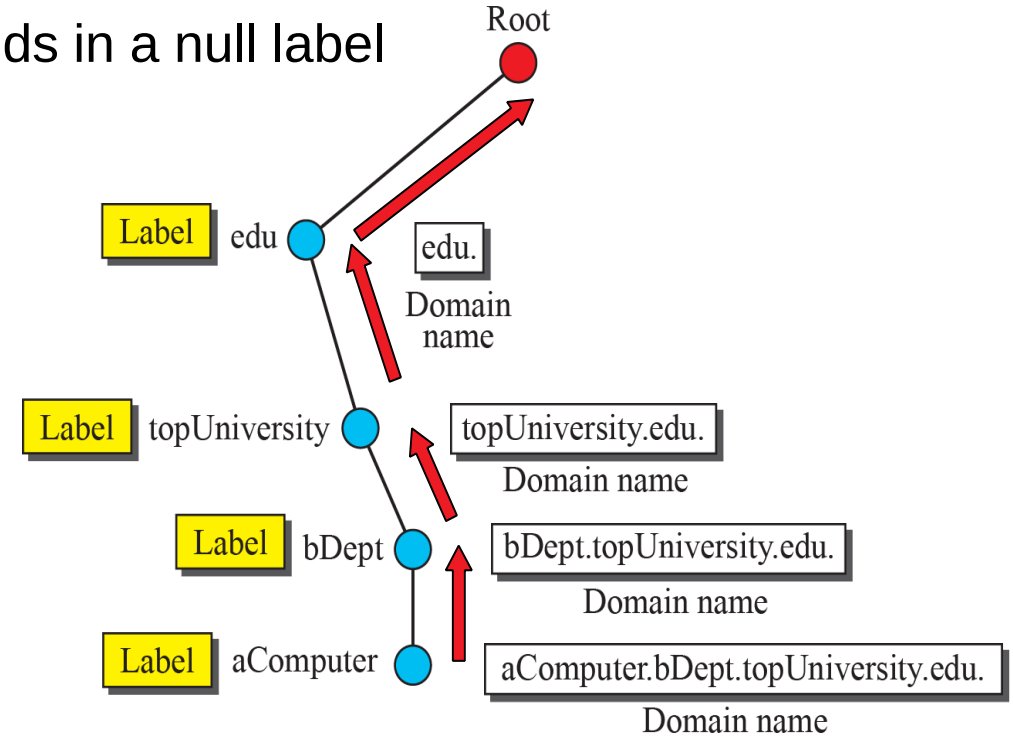
DNS: Label

- 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

2.3 Standard Client-Server Application

DNS: Domain Name (1/2)

- Each node in the tree has a domain name and 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
- A full domain name is always ends in a null label
  - The last character is a dot
- **Fully Qualified Domain Name** (**FQDN**): the complete domain name for a host on the Internet, which consists of the host name and the full domain name

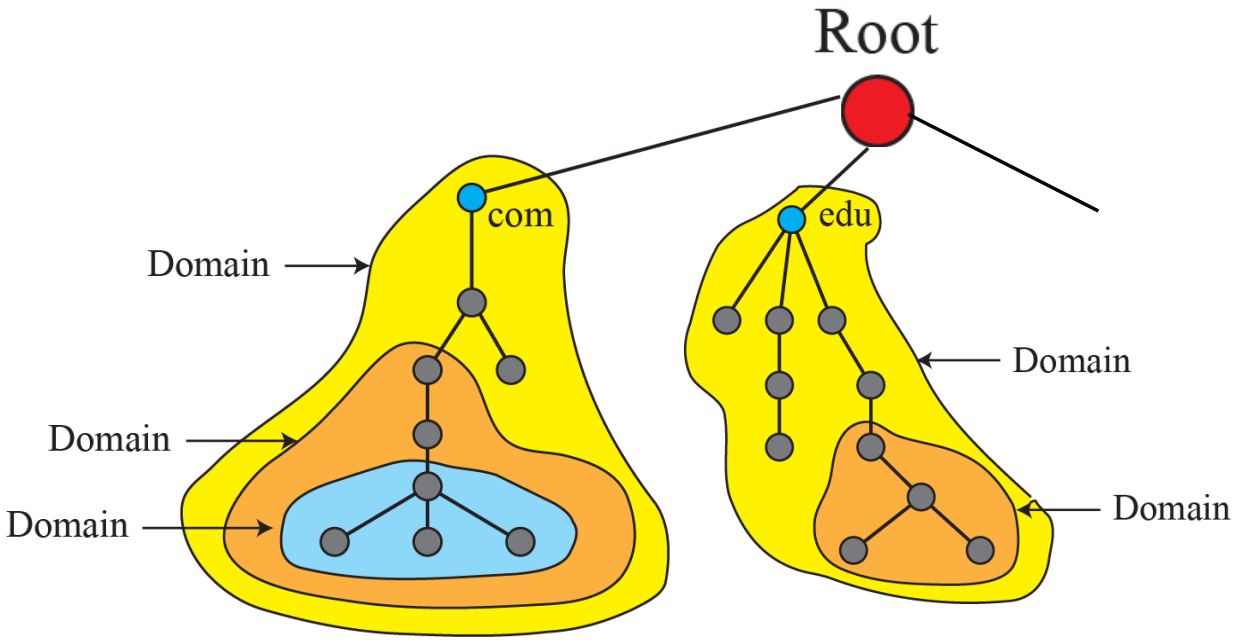


[Figure 2.37 Domain names and labels]

2.3 Standard Client-Server Application

DNS: Domain Name (2/2)

- **Domain:** a subtree of the domain name space
  - The name of the domain is the name of the node at the top of the subtree.

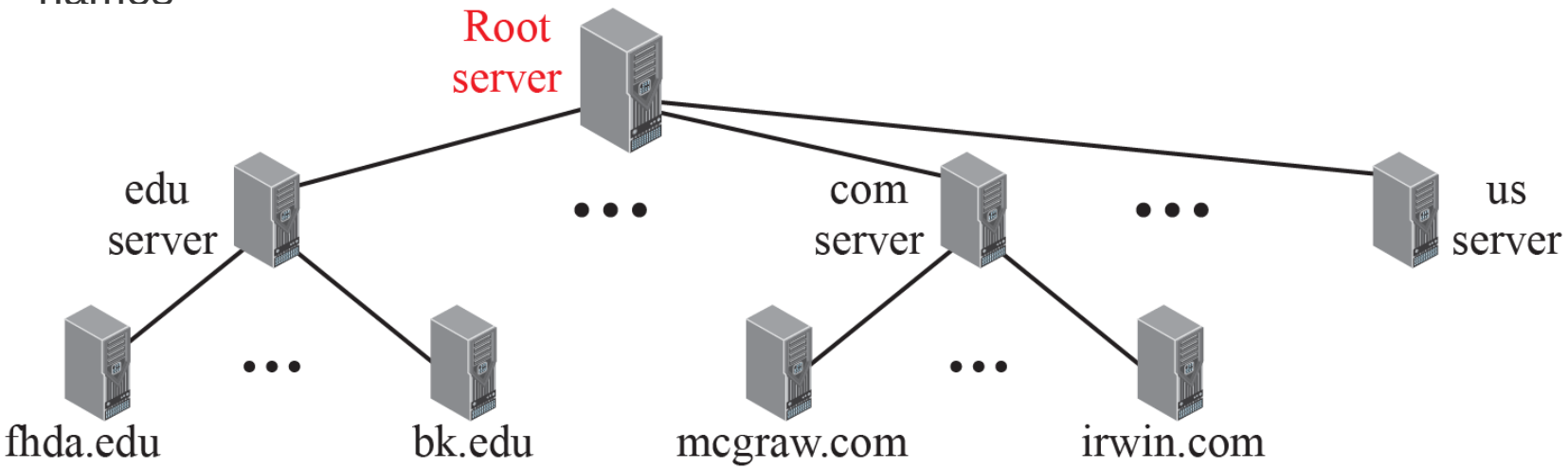


[Figure 2.38 Domains]

2.3 Standard Client-Server Application

DNS: Hierarchy of Name Servers (1/3)

- DNS allows domains to be divided further into smaller domains (subdomains)
- Each DNS server can be responsible (authoritative) for either a large or small domain
  - We have a hierarchy of servers in the same way that we have a hierarchy of names

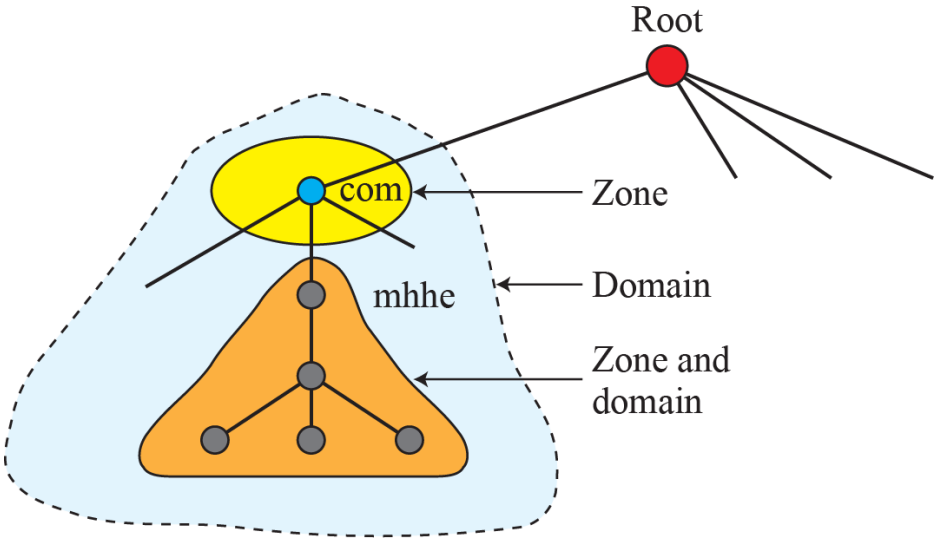


[Figure 2.39 Hierarchy of name servers]

2.3 Standard Client-Server Application

DNS: Hierarchy of Name Servers (2/3)

- **Zone**
  - The complete domain name hierarchy is divided among many servers
  - What a server is responsible for or has authority over is called a **zone**
  - If a server divides its domain into subdomains and delegates part of its authority to other servers, “domain” and “zone” refer to different things



[Figure 2.40 Zone]

2.3 Standard Client-Server Application

DNS: Hierarchy of Name Servers (3/3)

- 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
  - Root servers are distributed all around the world

2.3 Standard Client-Server Application

DNS: 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 and is responsible for creating, maintaining, and updating the zone file
  - A primary server loads all information from the disk file
- **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 나중에 쓸 때는 여기서 그냥 가져오려고 저장해 놓는다
  - A secondary server loads all information from the primary server

2.3 Standard Client-Server Application

DNS: DNS in the Internet (1/4)

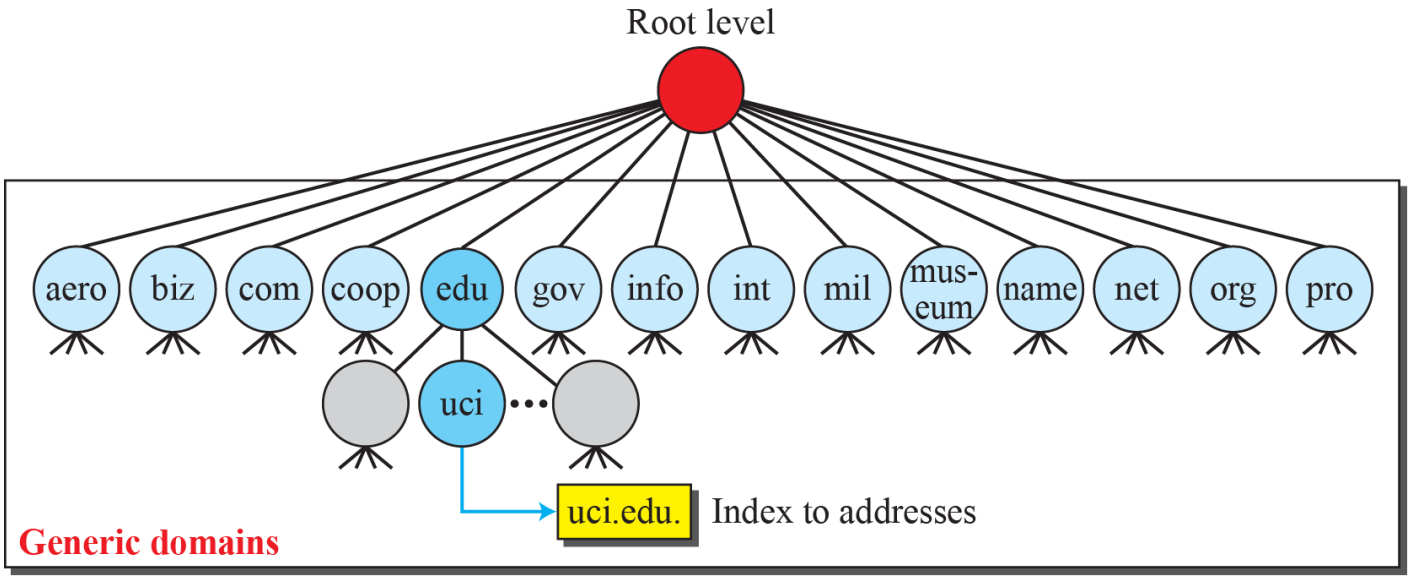
- DNS is a protocol that can be used in different platforms
- In the Internet, the domain name space (tree) was originally divided into three different sections: generic domains, country domains, and the inverse domain
  - The inverse domains are used to find the name of a host when given the IP address
  - The inverse domains are now deprecated (see RFC 3425) since it is extremely difficult to keep track them

2.3 Standard Client-Server Application

DNS: DNS in the Internet (2/4)

■ Generic Domains (1/2)

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



[Figure 2.41 Generic domains]

2.3 Standard Client-Server Application

DNS: DNS in the Internet (3/4)

■ Generic Domains (2/2)

- The first level in the generic domains section allows 14 possible labels
- These labels describe the organization types as listed in Table 2.12

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

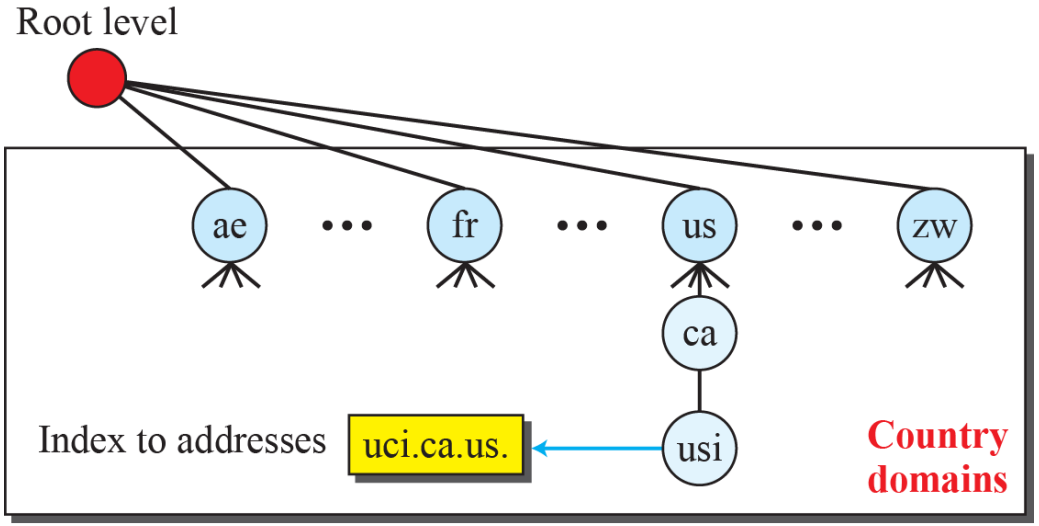
[Table 2.12 Generic domains]

2.3 Standard Client-Server Application

DNS: DNS in the Internet (4/4)

■ Country Domains

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



[Figure 2.42 Country domains]

2.3 Standard Client-Server Application

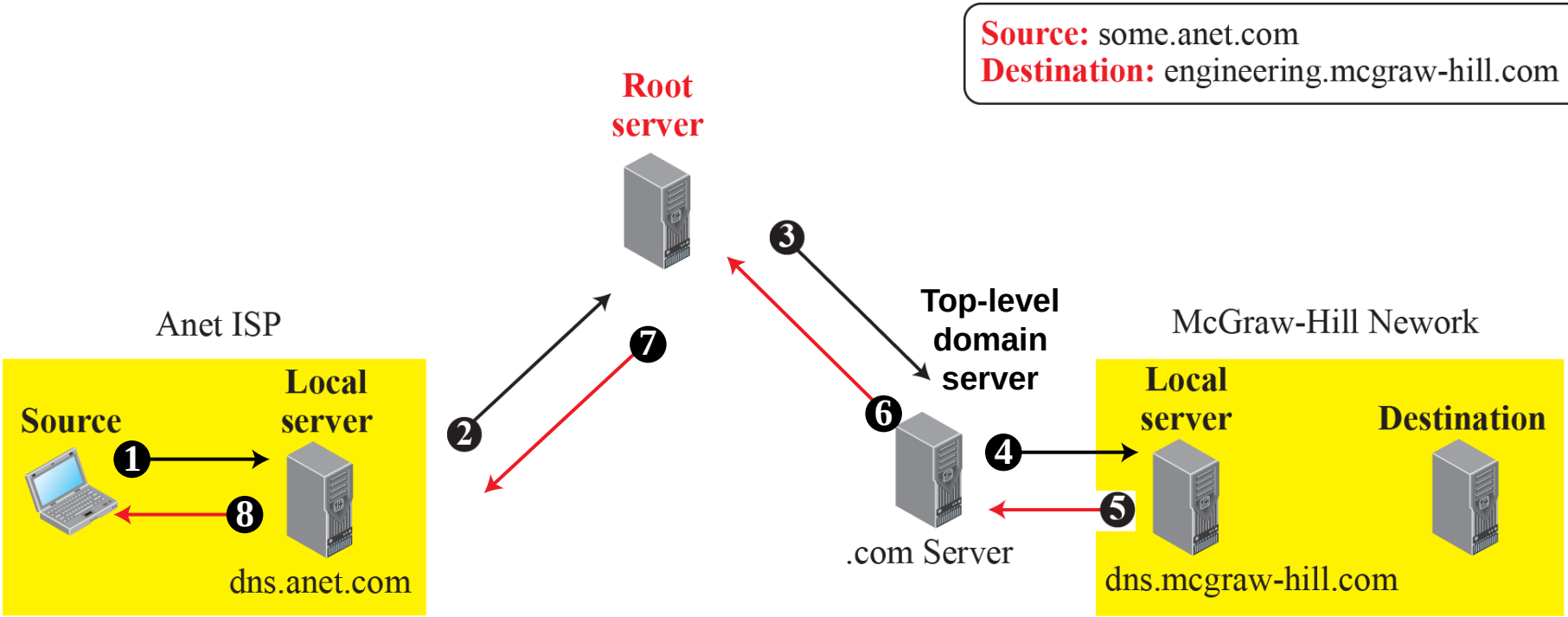
DNS: Resolution (1/4)

- Mapping a name to an address is called *name-address resolution*
- Scenario
  - Assume that an application program running on a host named ***some.anet.com*** needs to find the IP address of another host name ***engineering.mcgraw-hill.com*** to send a message to
  - Also, assume that the local DNS server of anet.com does not know the IP address of the host ***engineering.mcgraw-hill.com***
- **Recursive Resolution**
  - In Recursive Resolution, each server that does not know the mapping takes the responsible for recursively querying the appropriate server to find out the IP address of the destination host
  - Finally, the result is recursively sent back to the client through the chain of servers being queried previously

2.3 Standard Client-Server Application

DNS: Resolution (2/4)

■ **Recursive Resolution**



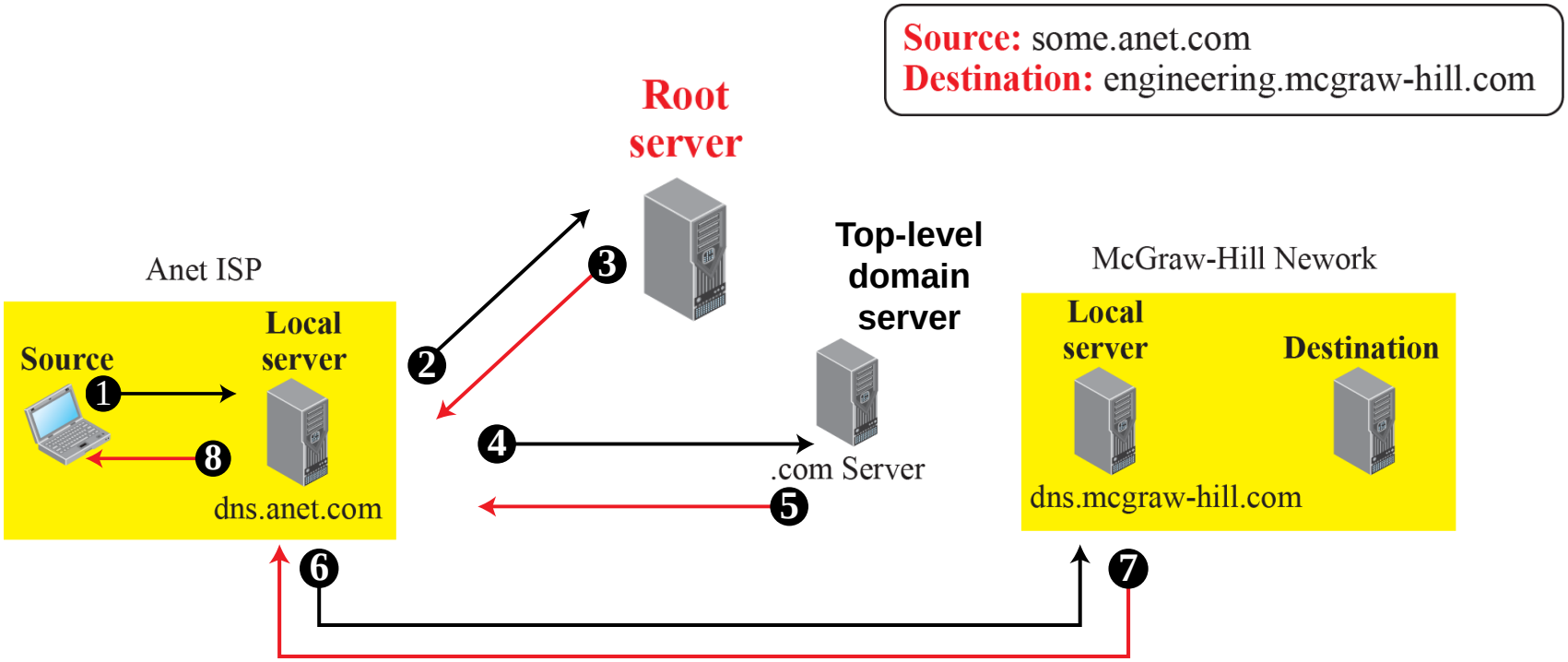
[Figure 2.43 Recursive resolution]

2.3 Standard Client-Server Application

DNS: Resolution (3/4)

■ Iterative Resolution

- In iterative resolution, each server that does not know the mapping sends the IP address of the next server back to the one that requested it



[Figure 2.44 Iterative resolution]

2.3 Standard Client-Server Application

DNS: Resolution (4/4)

- **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
  - This speeds up resolution, but it can also be problematic
    - Server may send an outdated mapping to client
    - To prevent this problem, *time-to-live* (TTL) is used for validating the cached mapping
      - Mapping along with an TTL is returned from the responsible DNS server
      - DNS requires that each server keep a TTL counter for each mapping it caches

cache coherence problem: cache와 sotrage의 정보가 다른 문제

2.3 Standard Client-Server Application

DNS: Resource Records (1/2)

- The zone information associated with a server is implemented as a set of *resource records*
- A *resource record* is a 5-tuple structure, as shown below

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

  - **Domain Name**: identifies the resource record
  - **Value**: information about the domain name
  - **TTL**: number of seconds for which the information is valid
  - **Class**: only one interested value, IN (for Internet)
  - **Type**: how the value should be interpreted

2.3 Standard Client-Server Application

DNS: Resource Records (2/2)

- Table 2.13 lists the common types and how the value is interpreted for each type

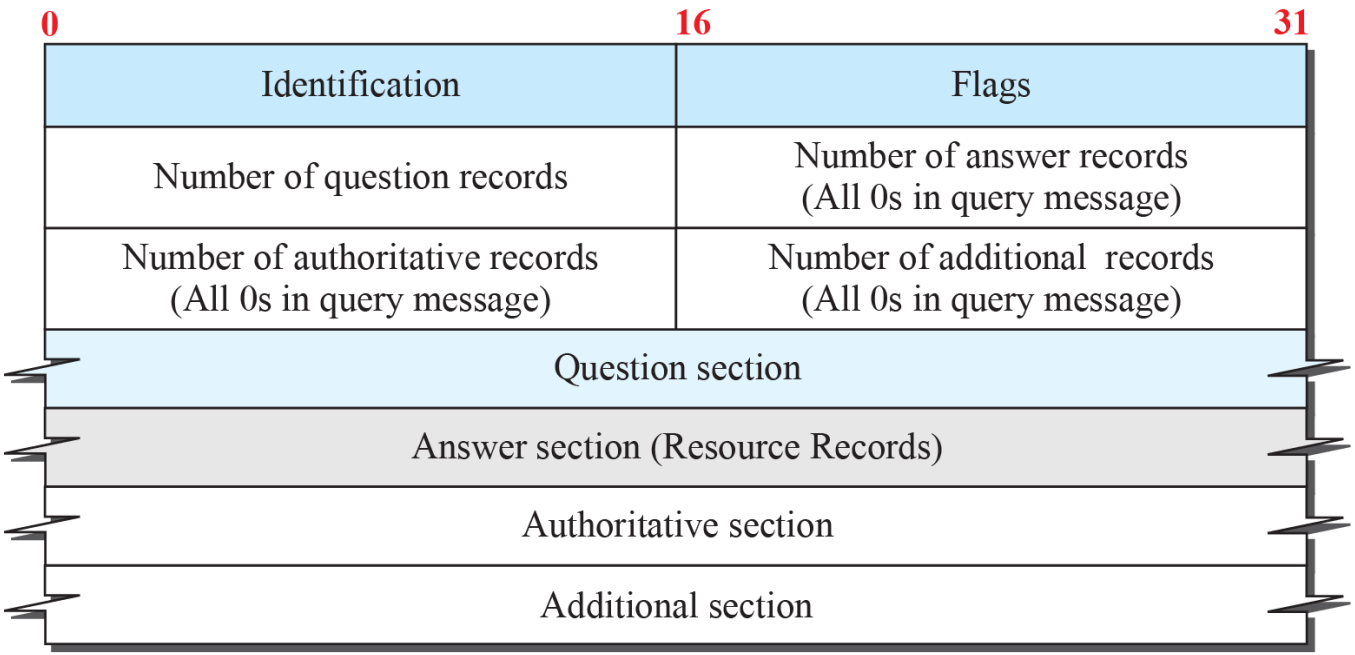
Type	*Interpretation of value*
A	A 32-bit IPv4 address (see Chapter 4)
NS	Identifies the authoritative servers for a zone
CNAME	Defines an alias for the official name of a host
SOA	Marks the beginning of a zone
MX	Redirects mail to a mail server
AAAA	An IPv6 address (see Chapter 4)

[Table 2.13 Types]

2.3 Standard Client-Server Application

DNS: DNS Messages

- DNS uses two types of message: query and response, with the same format



[Table 2.13 Types]

Note:

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

2.3 Standard Client-Server Application

DNS: DNS Resolver

- DNS client (or *resolver*) application ~~are~~ is used to retrieve address/name mapping
  - In UNIX and Windows, we have the utility program *nslookup*

- 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
```

2.3 Standard Client-Server Application

DNS: Encapsulation

- DNS can use either UDP and TCP
  - UDP is used when the size of the response message is less than 512 bytes
  - In case that the response message is more than 512 bytes, a TCP connection is used
    - If resolver has prior knowledge that the size is more than 512 bytes (e.g. in zone transferring from primary server to secondary server)
    - In case of no knowledge in place, if the size of response message exceeds 512 bytes, server informs client to open a TCP connection to get the response
- DNS uses port 53 for both UDP and TCP

2.3 Standard Client-Server Application

DNS: Registrars

- New domains are added to DNS through a registrar which is a commercial entity accredited by ICANN.
  - The requested domain name is first verified for the uniqueness before adding to 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

Domain name: ws.wonderful.com	**IP address:** 200.200.200.5

2.3 Standard Client-Server Application
DNS: 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 zone data file
  - This leads to a lot of manual updating

- **Dynamic Domain Name System** (DDNS) was devised to get the DNS master file updated dynamically
  - When a binding between a name and an address is determined, the information is sent, usually by DHCP (Chapter 4) to a primary DNS server
  - The primary server updates the zone and then notify the secondary server to do a replication

2.3 Standard Client-Server Application

DNS: Security of DNS

- DNS can be attacked in several ways including:
  - The attacker may read the response of DNS server to find the nature or names of sites the user mostly accesses; then can construct the user's profile for different purposes
  - The attacker may intercept the response of a DNS server and change it or create a totally new bogus response to direct the user to the faked site or domain
  - The attacker may flood DNS server to overwhelm it or eventually crash it

- To protect DNS, IETF has devised DNS Security (DNSSEC) that provides message origin authentication and message integrity using a security service called *digital signature* (chapter 10)

Practice Problems

1. When DNS use UDP / TCP?

512 bytes보다 짧으면 UDP 길면 TCP

2. Each Internet host will have at least one local name server and one authoritative name server. What role does each of these servers have in DNS?

official translation of a hostname to an IP: authoritative
proxy: local

2.4 P2P Paradigm

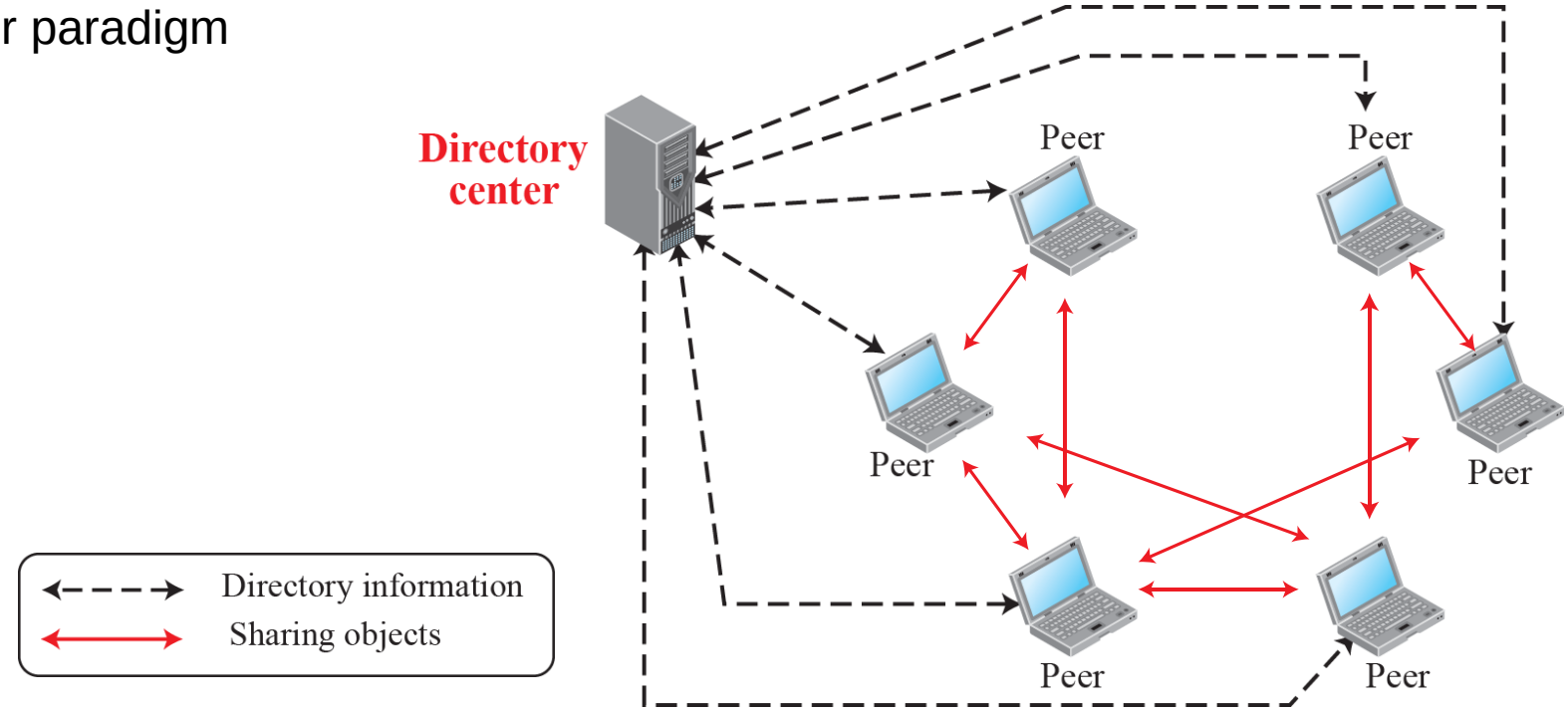
P2P Networks

- A peer-to-peer network is the one in which each computer can act as a client or server for the other computers
- When a peer in the network has a file (e.g., an audio or video file) to share, it makes it available to the rest of the peers
- P2P networks can be divided into two categories: centralized and decentralized networks

2.4 P2P Paradigm

P2P Networks: Centralized Networks

- In a centralized P2P network,
  - The directory system – listing of the peers and what they offer – uses the client-server paradigm
  - Only the storing and downloading of the file are done using the peer-to-peer paradigm



[Figure 2.46 Centralized network]

2.4 P2P Paradigm

P2P Networks: Decentralized Networks (1/3)

- A decentralized P2P network does not depend on a centralized directory system

  physically seperated network, logically a network

- In this model, peers arrange themselves into an overlay network (logical network made on top of the physical one)

- A decentralized P2P network is classified as either unstructured or structured networks

2.4 P2P Paradigm

P2P Networks: Decentralized Networks (2/3)

- Unstructured Networks
  - The nodes are linked randomly
  - A search in an unstructured P2P is not very efficient because a query to find a file must be flooded through the network
  - This network type produces significant traffic and still the query may not be resolved
  - Two example of this type of network are **Gnutella** and **Freenet** (for file sharing)

2.4 P2P Paradigm

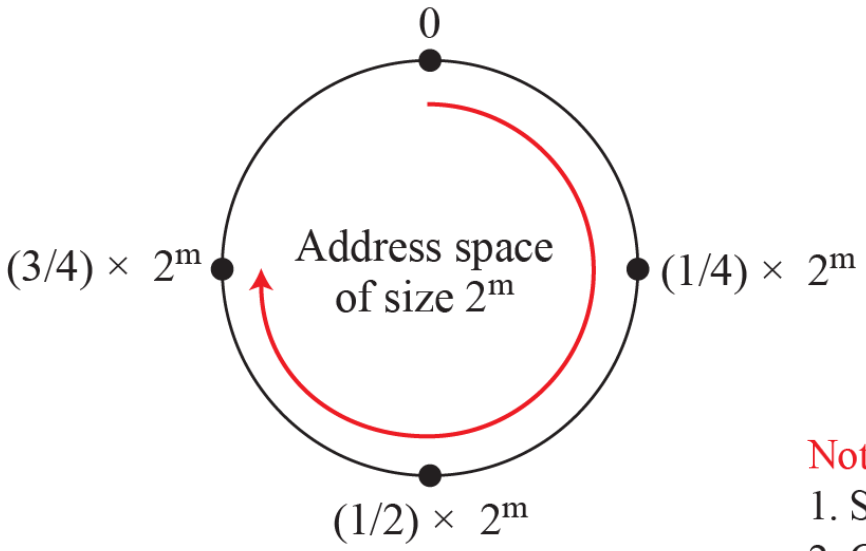
P2P Networks: Decentralized Networks (3/3)

- Structured Networks
  - A structured network uses a predefined set of rules to link nodes so that a query can be effectively and efficiently resolved
  - The most common technique used for this purpose is the Distributed Hash Table (DHT)
  - One popular P2P file sharing protocol that uses the DHT is BitTorrent

2.4 P2P Paradigm

Distributed Hash Table (1/5)

- A Distributed Hash Table (DHT) distributes data among a set of nodes according to some predefined rules
- In a DHT-based network, each data item and the peer is mapped to a point in a large address space of size 2^m
- Most of the DHT implementations use m=160



[Figure 2.47 Address space]

Torus wrapping around

Note:
1. Space range is 0 to $2^m - 1$.
2. Calculation is done modulo 2^m.

2.4 P2P Paradigm

Distributed Hash Table (2/5)

- Hashing Peer Identifier
  - The fist step in creating the DHT system is to place all peers on the address space ring
  - This is normally done by using a *hash* function that hashes the peer identifier, normally its IP address, to an m-bit integer, called a ***node ID***

    node ID = hash (Peer IP address)

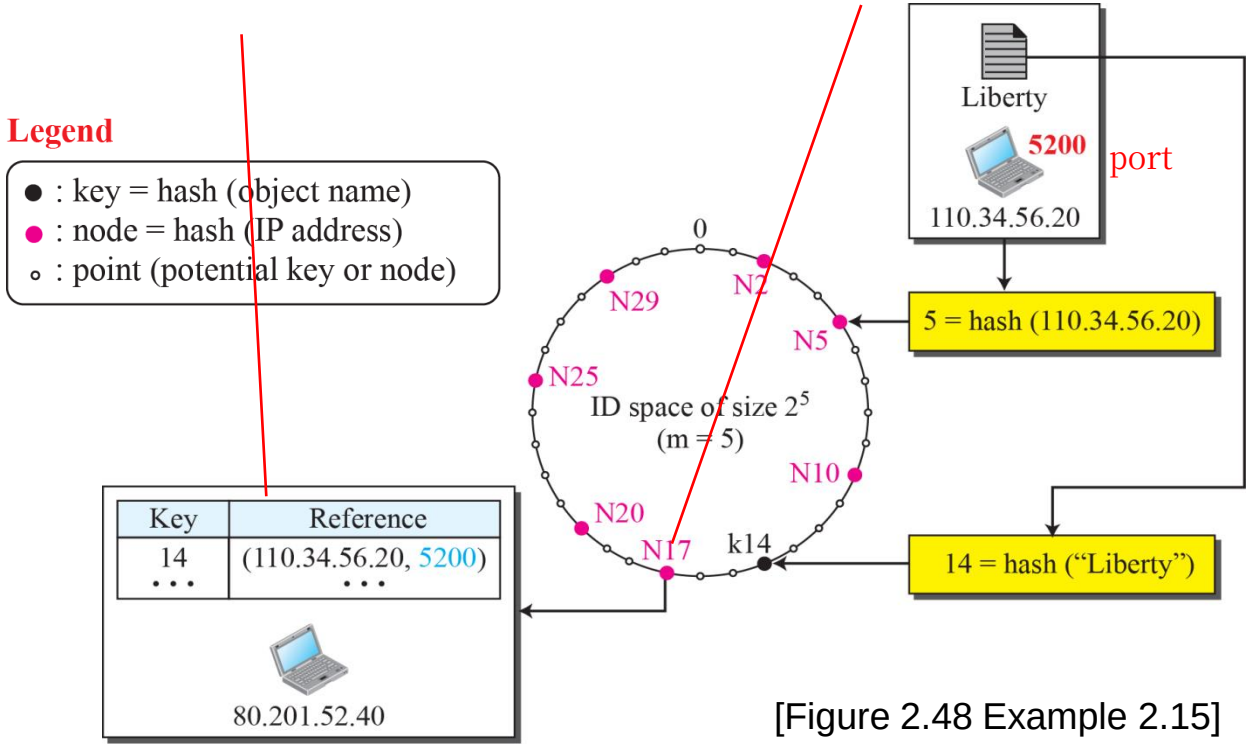
- Hashing Object Identifier
  - The name of the object (e.g., a file) to be shared is also hashed to an m-bit integer in the same address space
  - The result in DHT notation is called **a *key***

    key = hash(Object name)

2.4 P2P Paradigm

Distributed Hash Table (3/5)

- Storing the Object
  - Assume that *m*=5 and several peers have already joined the group
  - The node **N5** has a file named *Liberty* that wants to share with its peers
    - Reference to this file is stored at the closest node to the hashed key of the file



[Figure 2.48 Example 2.15]

2.4 P2P Paradigm

Distributed Hash Table (4/5)

- Routing
  - Each node needs to have a partial knowledge about the ring to route a query to a particular node which is responsible for storing the reference to an object
- Arrival and Departure of Nodes
  - When a computer launches the DHT software, it joins the network; when it is turned off or the peer closes the software, it leaves the network
  - The arrival or departure of the nodes need to be handled by a clear and efficient strategy

2.4 P2P Paradigm

Distributed Hash Table (5/5)

- There are several protocols that implement DHT systems
- Three of these protocols, Chord, Pastry, and Kademlia, will be discussed for some reasons
  - Chord protocol: due to its simplicity and elegant approach to routing queries
  - Pastry protocol: different approach than Chord, but close to Kademlia protocol in routing strategy
  - Kademlia protocol: used in the most popular file-sharing network, BitTorrent

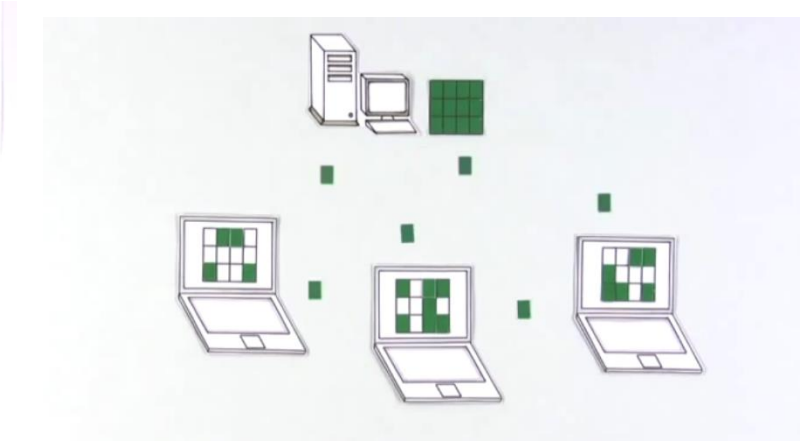
2.4 P2P Paradigm

BitTorrent: Introduction (1/2)

- Video Content
  - What BitTorrent is and Why BitTorrent?
  - An explanation of sharing huge contents on the Internet
  - How BitTorrent works?
  - Link: https://www.youtube.com/watch?v=6PWUCFmOQwQ

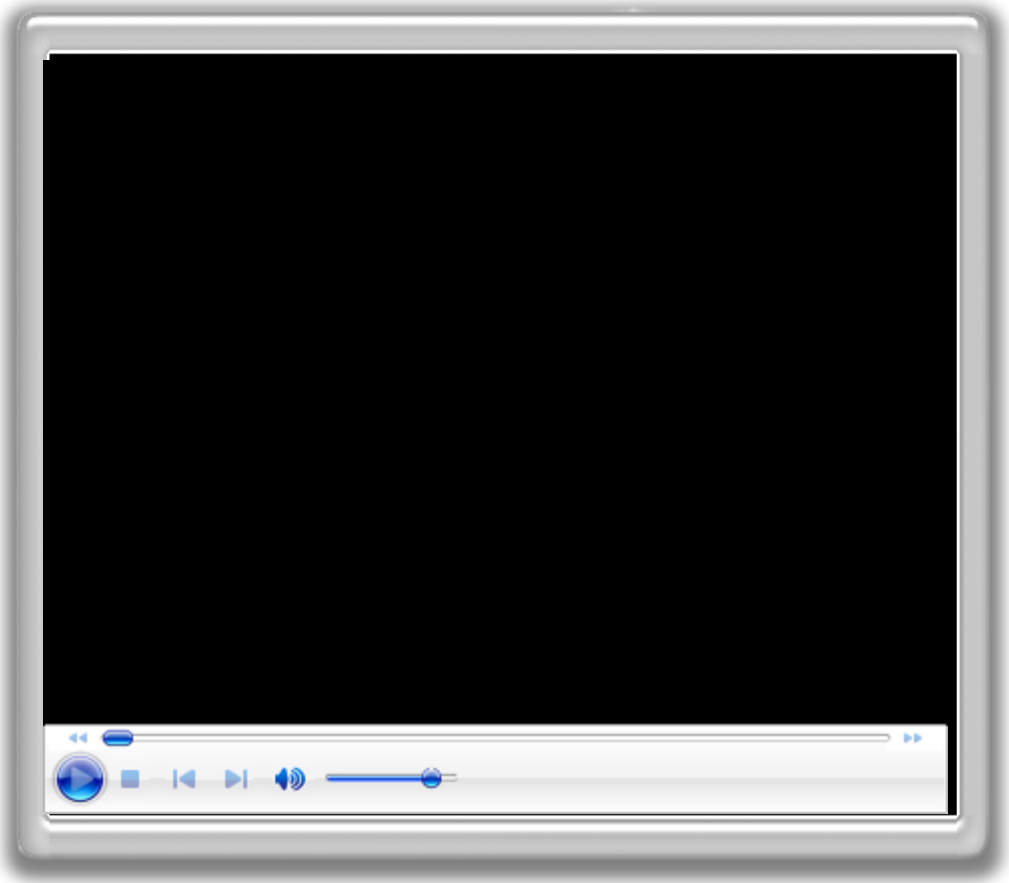
BitTorrent™

www.bittorrent.com



2.3 Standard Client-Server Application

BitTorrent: Introduction (2/2)



2.4 P2P Paradigm

Chord: Identifier Space

- Data items and nodes in Chord create an identifier space of size 2^m points distributed in a circle in the clockwise direction
  - We refer to the identifier of a data item as k (for ***key***) and the identifier of a peer as N (for ***node***)
- Arithmetic in the space is done modulo 2^m, which means that the identifiers are wrapped from $\mathbf{2^m - 1}$ back to **0**
- The closest peer with N ≥ k (key), called the successor of k, hosts the value (k, v), where v is information about the peer server that has the object
  - The peer that stores the data item and the peer that holds the pair (k, v) are not necessarily the same

2.4 P2P Paradigm

Chord: Finger Table (1/2)

- A node in the Chord algorithm should be able to resolve a query: given a key, the node should be able to find the node identifier responsible for that key or forward the query to another node
- Each node needs to have a routing table, called a finger table by Chord
- The finger table of a node contains entries for *m Successors*
  - i^{th} entry contains a reference to the first node that succeeds this node by at least $2^{(i-1)}$ on the identifier circle
- Each node also stores one *Predecessor*

i	*Target Key*	*Successor of Target Key*	*Information about Successor*
1	N + 1	Successor of(N + 1)	IP address and port of successor
2	N + 2	Successor of(N + 2)	IP address and port of successor
⋮	⋮	⋮	⋮
m	$N + 2^{m-1}$	Successor of($N + 2^{m-1}$)	IP address and port of successor

[Table 2.14 Finger table]

2.4 P2P Paradigm

Chord: Finger Table (2/2)

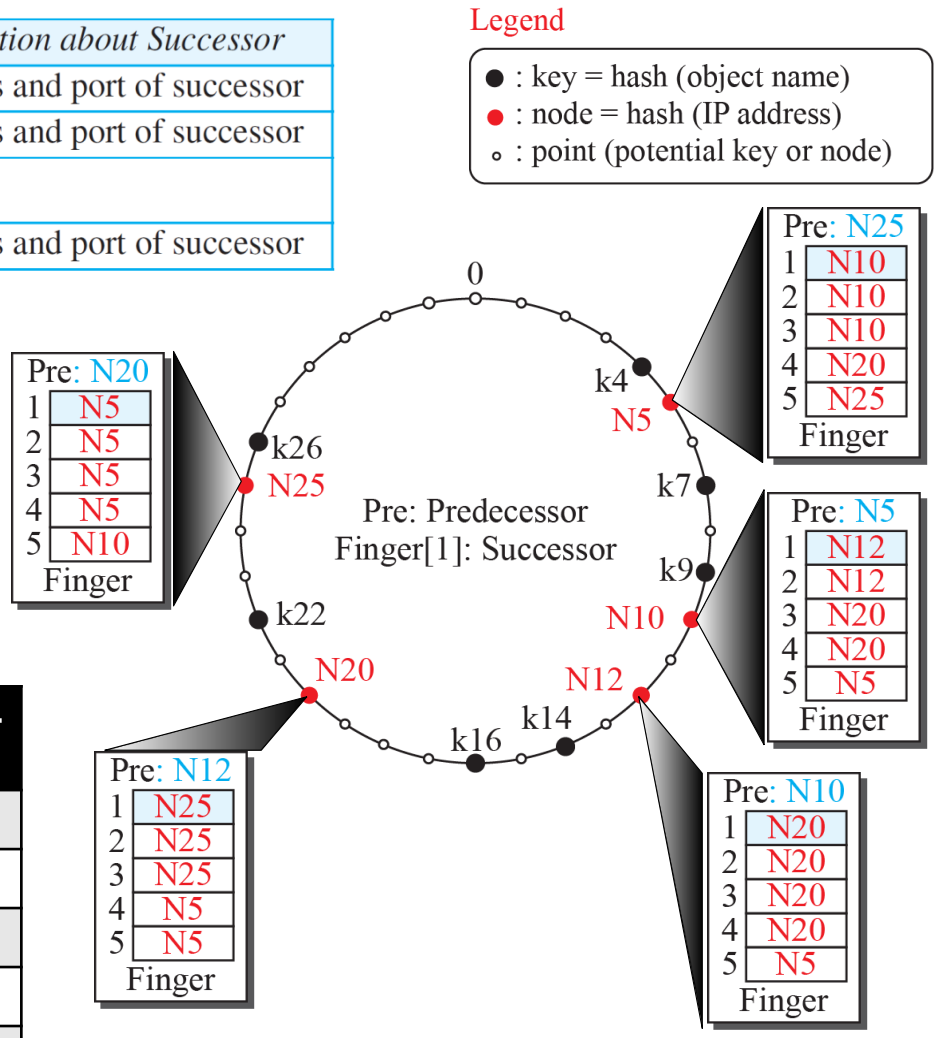
i	*Target Key*	*Successor of Target Key*	*Information about Successor*
1	N + 1	Successor of(N + 1)	IP address and port of successor
2	N + 2	Successor of(N + 2)	IP address and port of successor
⋮	⋮	⋮	⋮
m	$N + 2^{m-1}$	Successor of($N + 2^{m-1}$)	IP address and port of successor

[Table 2.14 Finger table]

- The successor of k hosts the information about the peer server that has the object
- The finger table of **N5**, **N10**

i	Taget Key	Succe-ssor
1	6(5+1)	**N10**
2	7(5+2)	**N10**
3	9(5+4)	**N10**
4	13(5+8)	**N20**
5	21(5+16)	**N25**

i	Target Key	Succe-ssor
1	11(10+1)	**N12**
2	12(10+2)	**N12**
3	14(10+4)	**N20**
4	18(10+8)	**N20**
5	26(10+16)	**N5**



[Figure 2.49 An example of a ring in Chord]

2.4 P2P Paradigm

Chord: Interface (1/7)

- Lookup
  - The mostly used operation in Chord

```
Lookup (key)
{
        if (node is responsible for the key)
                return (node's ID)
        else
                return find_succesor (key)
}

find_successor (id)
{
        x = find_ predecessor (id)
        return x.finger[1]
}
```

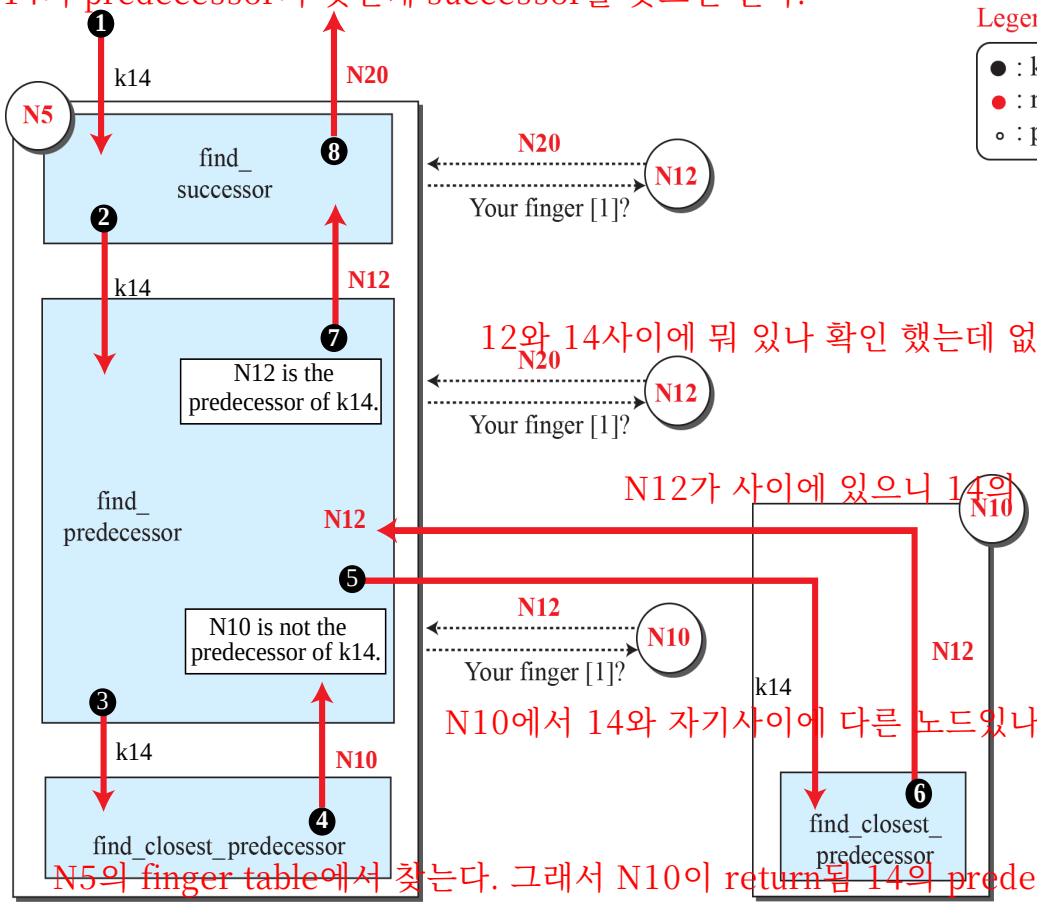
```
find_predecessor (id)
{
        x = N          // N is the current node
        while (id ∉ (x, x.finger[1]]
        {
                x = x.find_closest_predecessor (id)
        }
        return x
}

find_closest_predecessor (id)
{
        for (i = m downto 1)
        {
                if (finger [i] ∈ (N, id))
                        return (finger [i])
        }
        return N
}
```

2.4 P2P Paradigm

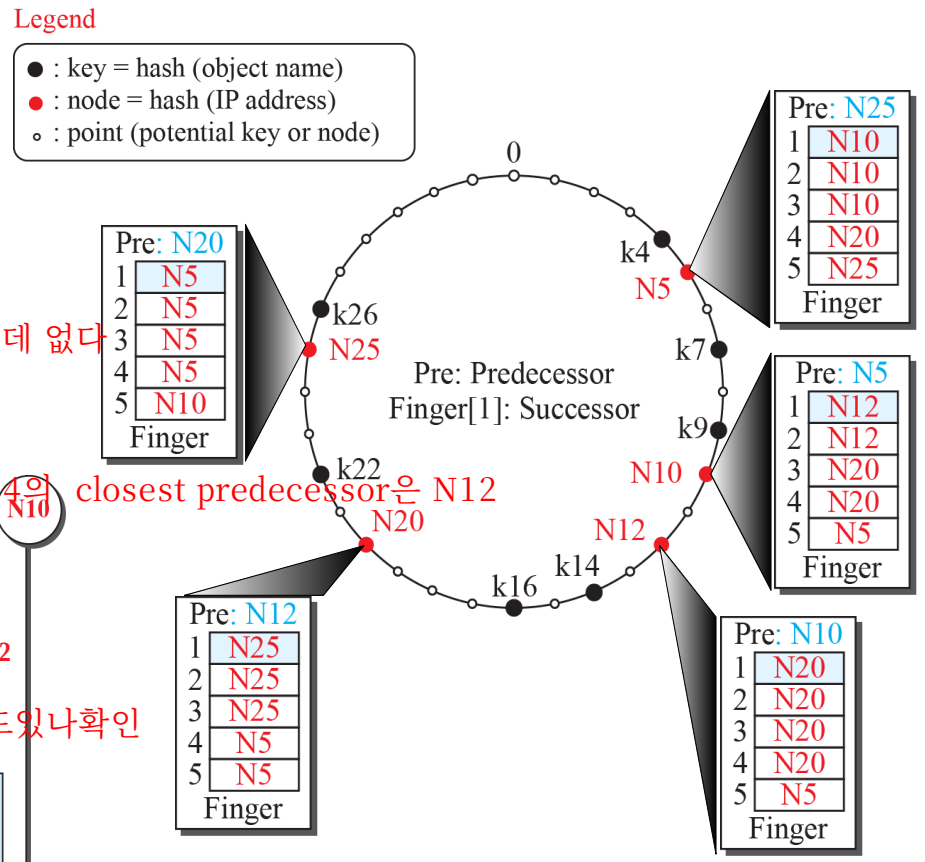
Chord: Interface (2/7)

■ Assume node N5 needs to find the responsible node for key k14

14의 predecessor의 첫번째 successor를 찾으면 된다.



[Figure 2.50 Example 2.16]



[Figure 2.49 An example of a ring in Chord]

2.4 P2P Paradigm

Chord: Interface (3/7)

- Stabilize
  - Each node in the ring periodically uses this operation to validate its information about its successor and let the successor validate its information about its predecessor

```
Stabilize ()
{
        P = finger[1].Pre                          //Ask the successor to return its predecessor
        if(P∈ (N, finger[1]))   finger[1] = P      // P is the possible successor of N
        finger[1].notify (N)                       // Notify P to change its predecessor
}
Notify (x)
{
            if (Pre = null or x ∈ (Pre, N))        Pre = x
}
```

2.4 P2P Paradigm

Chord: Interface (4/7)

- Fix_Finger
  - Each node in the ring must periodically call this operation to maintain its finger table update
  - One finger is chosen randomly to be updated in each call

```
Fix_Finger ()
{
        Generate (i ∈ (1, m])          //Randomly generate i such as 1< i ≤ m
        finger[i] = find_successor (N + 2^(i-1))          // Find value of finger[i]
}
```

2.4 P2P Paradigm

Chord: Interface (5/7)

- Join
  - When a peer joins the ring, it uses the join operation and known ID of another peer to find its successor and set its predecessor to null
  - Stabilize function is called then to validate its successor
  - Move-key function is called to ask the successor to transfer the keys that this node is responsible for

```
Join (x)
{       // N is the current node
        Initialize (x)
        finger[1].Move_Keys (N)
}
```

```
Initialize (x)
{
        Pre = null
        if (x = null) finger[1] = N
        else finger[1] = x. Find_Successor (N)
}

Move_Keys (x)
{
        for (each key k)
        {
                if (x ∈ [k, N)) move (k to node x)
        }
}
```

2.4 P2P Paradigm

Chord: Interface (6/7)

- Assume that node **N17** joins the ring with the help of **N5**

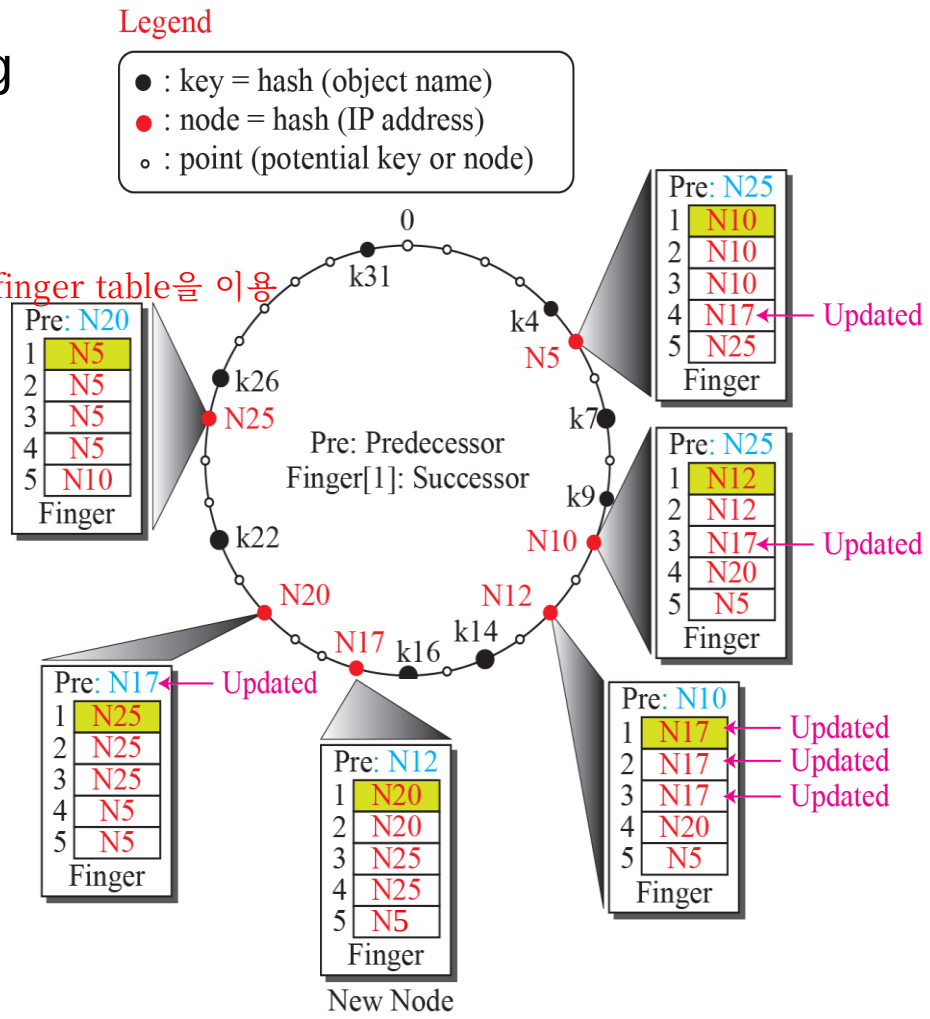
1. N17 set its predecessor to null and its successor to N20 by *Initialize*(5) N5의 finger table을 이용

2. N17 then asks N20 to send k14 and k16 to N17 by *Move_Keys*(17)

3. N17 validates its own successor by *Stabilize* and asks N20 to change its predecessor to N17

4. The predecessor of N17 is updated to N12 when N12 uses *Stabilize*

5. The finger table of nodes N17, N10, N5, and N12 is changed by *Fix_Finger*



[Figure 2.51: Example 2.17]

2.4 P2P Paradigm

Chord: Interface (7/7)

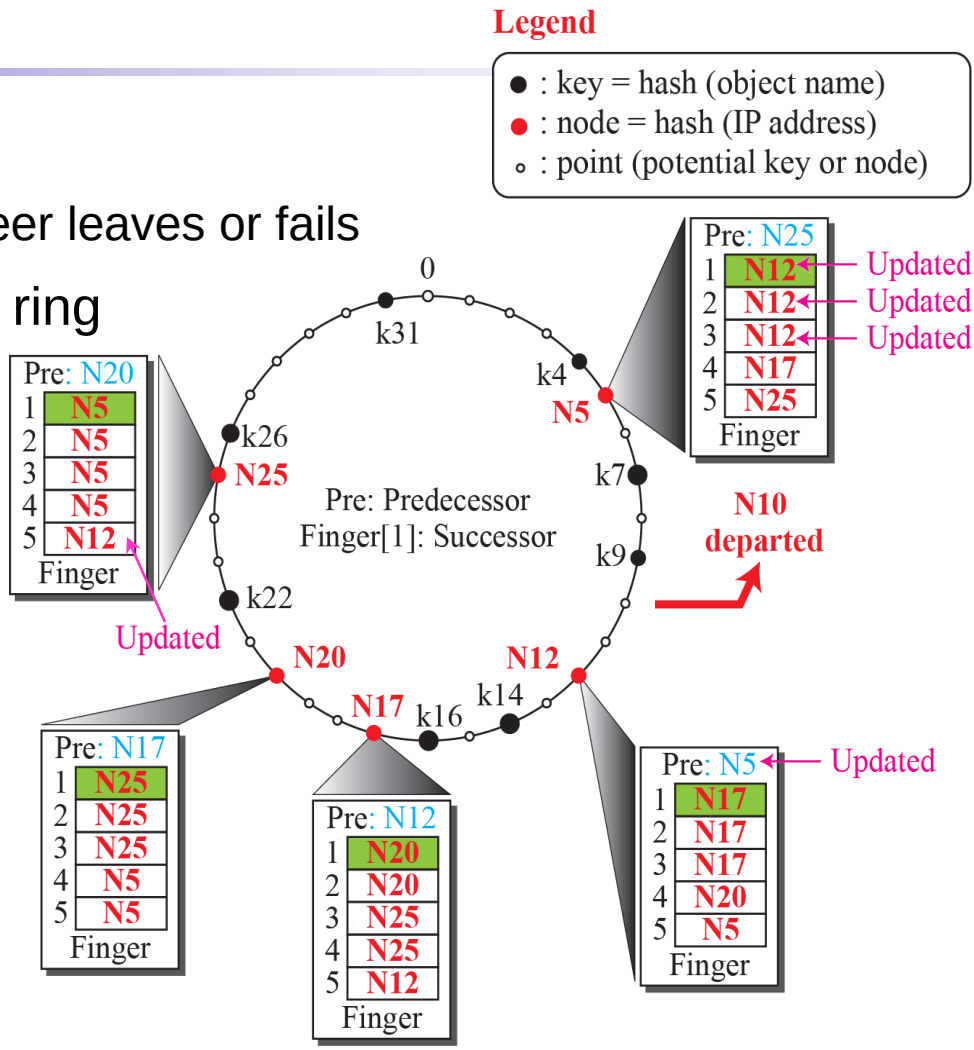
- Leave or Fail
  - The ring must be stabilized if a peer leaves or fails
- Assume that node **N10** leaves the ring

1. Node N5 detects N10's departure, then changes its successor to N12 (the second in the list of successors)

2. Node N5 launches the *Stabilize* function and asks N12 to change its predecessor to N5

3. Hopefully, k7 and k9, which were under the responsibility of N10, have been duplicated in N12 before the departure of N10

4. After a few calls of *Fix_Finger*, nodes N5 and N25 update their finger tables



[Figure 2.52: Example 2.18]

2.4 P2P Paradigm

Pastry: Identifier Space

- In Pastry, like Chord, nodes and data items are ***m*-bit identifiers** that create an identifier space of $\mathbf{2^m}$ points distributed uniformly on a circle **in the clock wise direction**
  - An identifier is seen as an n-digit string in base 2^b
  - Common value for m is 128 and for b is 4; in this case an identifier is a 32-digit number in base 16 (hexadecimal)
- A key is stored in the node whose identifier is numerically closest to the key
  - A key may be stored in its successor and/or predecessor node

2.4 P2P Paradigm

Pastry: Routing (1/2)

- Each node contains two entities: *routing table* and *leaf set*
- Routing table
  - Pastry requires that each node keeps a routing table with n rows and (2^b) columns
    - When m=128 and b=4, we have **32** (m/b) rows and **16** columns ($2^{128} = 16^{32}$)
  - For node N, the cell at row i and column j, Table [i,j], gives the identifier of a node (if exists):
    - Shares the i left most digits with the identifier for N
    - Its $(i+1)^{th}$ digit has a value of j
- Leaf set: set of 2^b identifiers
  - 2^{b-1} nodes before the current node in the ring
  - 2^{b-1} nodes after the current node in the ring

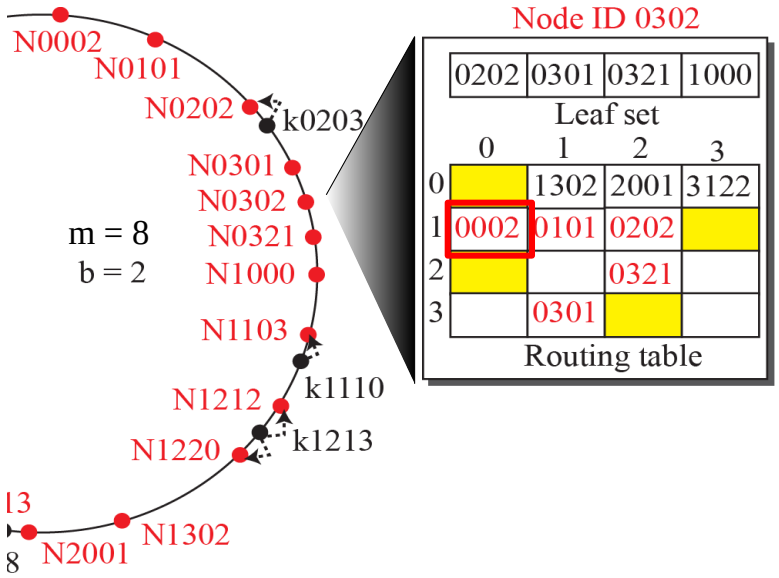
Common prefix length	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*A*	*B*	*C*	*D*	*E*	*F*
0																
1																
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
31																

[Table 2.19: Routing table for a node in Pastry]

2.4 P2P Paradigm

Pastry: Routing (2/2)

- Assume that m=8 bits and b=2, we have up to 2^m = **256 identifiers**, and each identifier has m/b = **4 digits** in base 2^b=4
  - The key k1213 is stored in two nodes because it is equidistant from them
  - For node N=0302, the value of Table [1,0] can be the identifier of a node with ID=***00xx*** (e.g. ***0002***)
  - Yellow cell corresponds to the digits of ID of N
  - White cell means no live node in the network satisfies the condition



[Figure 2.53: An example of a Pastry ring]

2.4 P2P Paradigm

Pastry: Lookup (1/3)

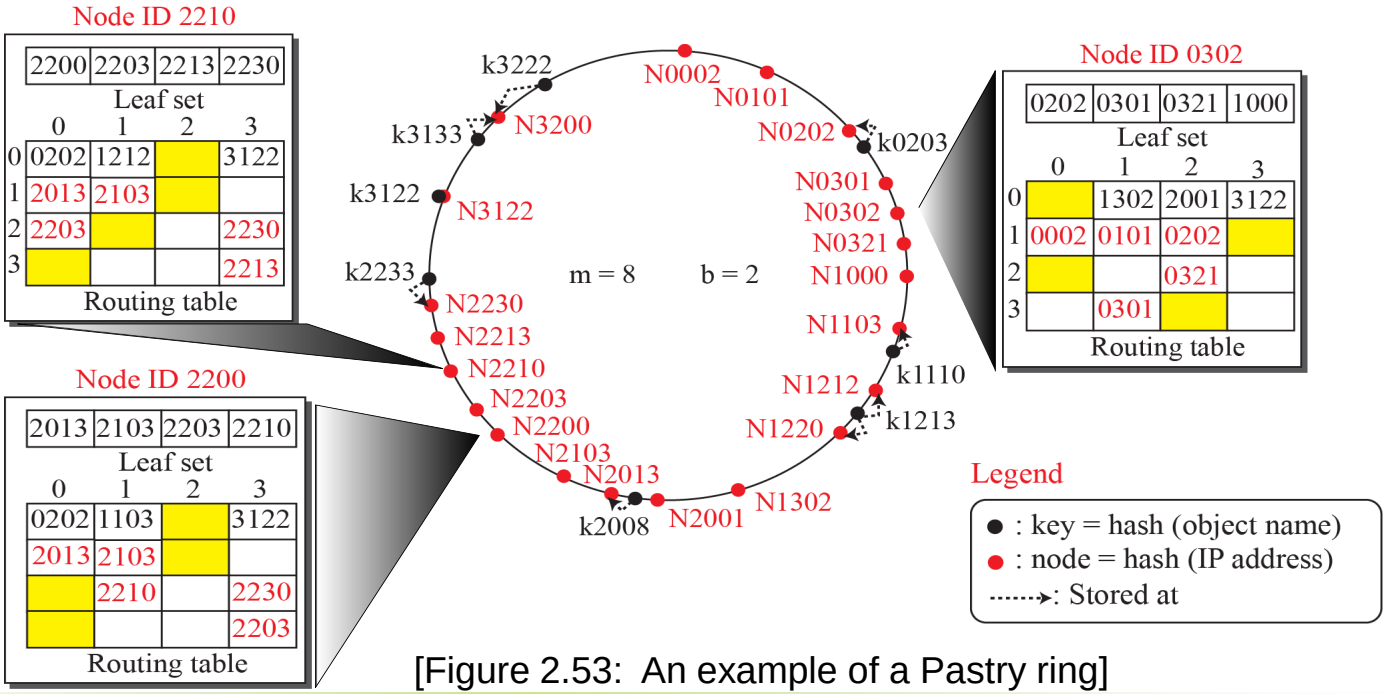
- Pseudo code of the lookup operation

```
Lookup (key)
{
        if (key is in the range of N's leaf set)
                forward the message to the closest node in the leaf set
        else
                route (key, Table)
}
route (key, Table)
{
        p = length of shared prefix between key and N
        v = value of the digit at position p of the key          // Position starts from 0
        if (Table [p, v] exists)
                forward the message to the node in Table [p, v]
        else
                forward the message to a node sharing a prefix as long as the current node, but
                numerically closer to the key.
}
```

2.4 P2P Paradigm

Pastry: Lookup (2/3)

- Assume that node **N0302** receives a query to find the node responsible for **key 0203**
- This node is not responsible for this key, but the key is in the range of its leaf set
- The closest node in this set is the node **N0202**; so the query is sent to this node

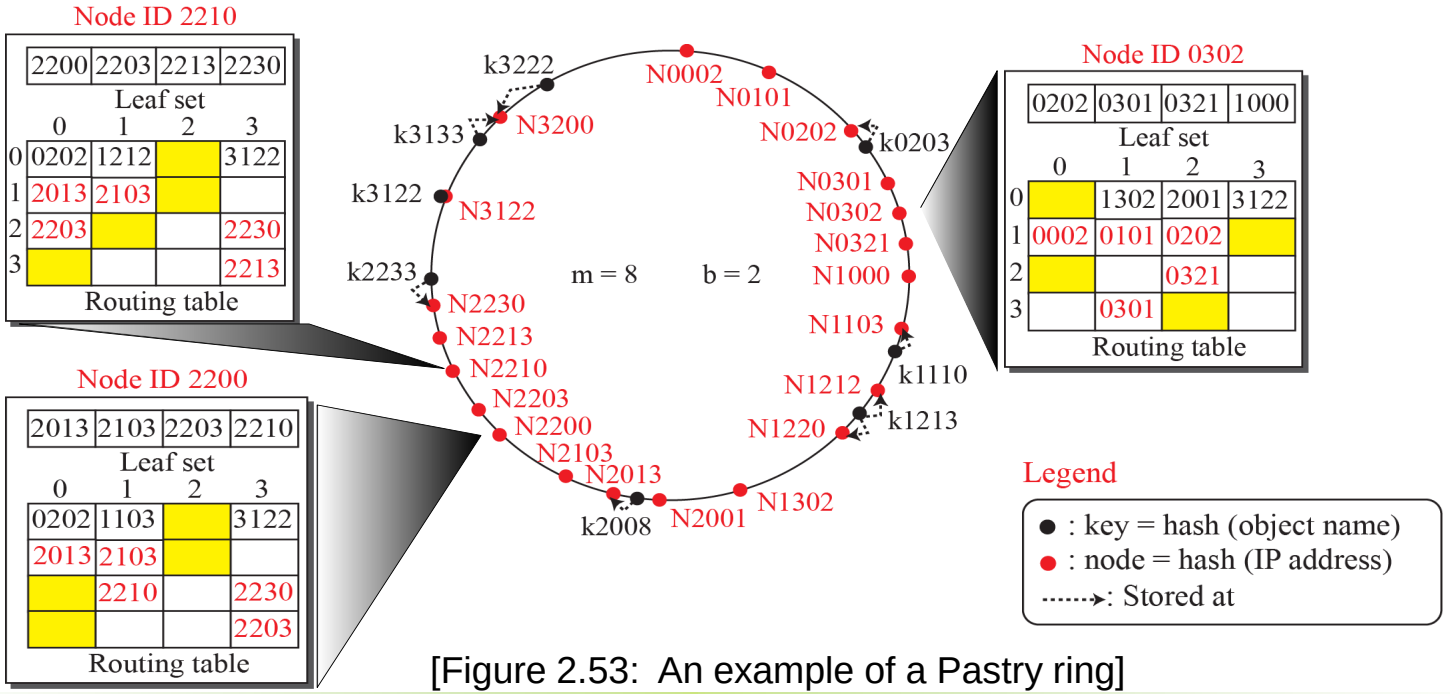


[Figure 2.53: An example of a Pastry ring]

2.4 P2P Paradigm

Pastry: Lookup (3/3)

- Assume that node **N2210** receives a query to find the node responsible for **key 2008**
- This node is not responsible for this key, and the key is not in the range of its leaf set
- The query is sent to node N2013 (value of Table [1, 0], where p=1, v=0)

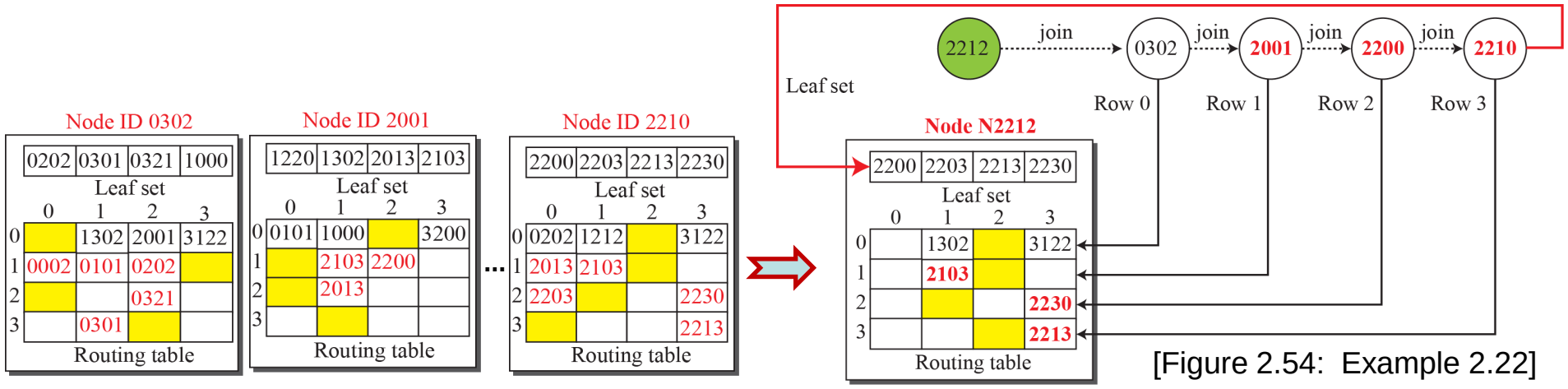


[Figure 2.53: An example of a Pastry ring]

2.4 P2P Paradigm

Pastry: Join

- Assume that node N2212 joins the network and it knows node N0302
  - The process of building the routing table for N2212 is similar as querying data, but with the *join* message (key=k2212)
    - Each node receiving *join* message replies N2212 with the corresponding row in its routing table
    - N2212 uses the appropriate parts of this information to build its corresponding row
  - Node N2210 sends its leaf set to N2212; then N2212 exchanges information with nodes in its routing table and leaf set to improve its routing information and allows those nodes to update theirs



[Figure 2.54: Example 2.22]

2.4 P2P Paradigm

Pastry: Leave or Fail

- Probe messages are exchanged periodically for testing the liveliness of nodes in the leaf set and routing table
  - If a node is not responding to the probe message, it is assumed to be failed or departed
- If a node in leaf set fails or departs, the local node repair its leaf set using information in the leaf set of the live node in its leaf set with the largest identifier
- If a node in Table [i, j] fails or departs, the local node replaces this one by the identifier in Table [i, j] of a live node in the same row

2.4 P2P Paradigm

Kademlia

- Kademlia, like Pastry, routes messages based on the distance between nodes

non - real

- The distance between the two identifiers (nodes or keys) is measured as the bitwise exclusive-or (XOR) between them

- In other words, if x and y are two identifiers:

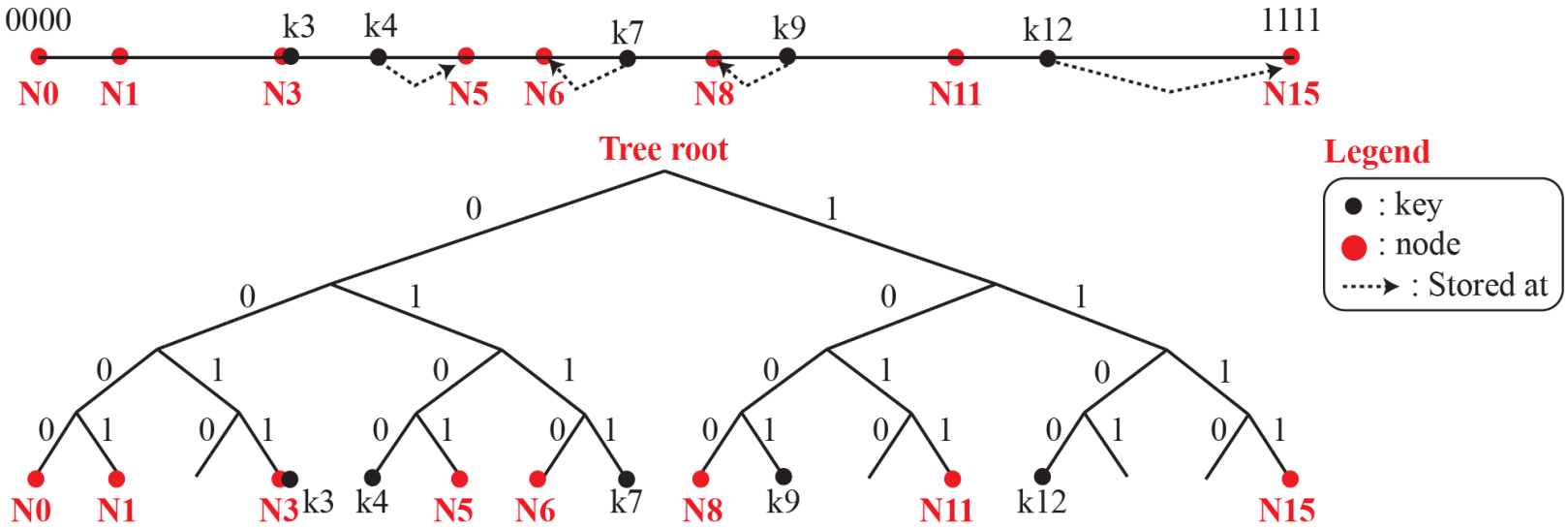
$$\text{distance}(x, y) = x \oplus y$$

- For example, 12 $(1100)_2 \oplus$ 11 $(1011)_2$ = **7 $(0111)_2$**

2.4 P2P Paradigm

Kademlia: Identifier Space

- In Kademlia, nodes and data items are m-bit identifiers that create an identifier space of 2^m points distributed on the leaves of a binary tree
  - Assume that m=4, we have 16 (2^4) identifiers in the identifier space
  - The key **k3** is stored in **N3** because $3 \oplus 3 = 0$
  - Although the key **k7** looks numerically equidistant from N6 and N8, it is stored only in **N6** because $6 \oplus 7 = 1$ but $6 \oplus 8 = 14$



[Figure 2.55: Example 2.23]

2.4 P2P Paradigm

Kademlia: Routing Table (1/3)

- Each node has m subtrees corresponding to m rows in the routing table, but only one column
- Subtree *i* includes nodes that share *i* leftmost bit (common prefix) with the corresponding node
  - Assume that each row holds the identifier of one of the nodes in the corresponding subtree

Common prefix length	*Identifiers*
0	Closest node(s) in subtree with common prefix of length 0
1	Closest node(s) in subtree with common prefix of length 1
⋮	⋮
$m-1$	Closest node(s) in subtree with common prefix of length $m-1$

공통된 prefix가 없는 subtree에서 closest 찾는다

공통된 prefix가 1인 subtree에서 closest 찾는다

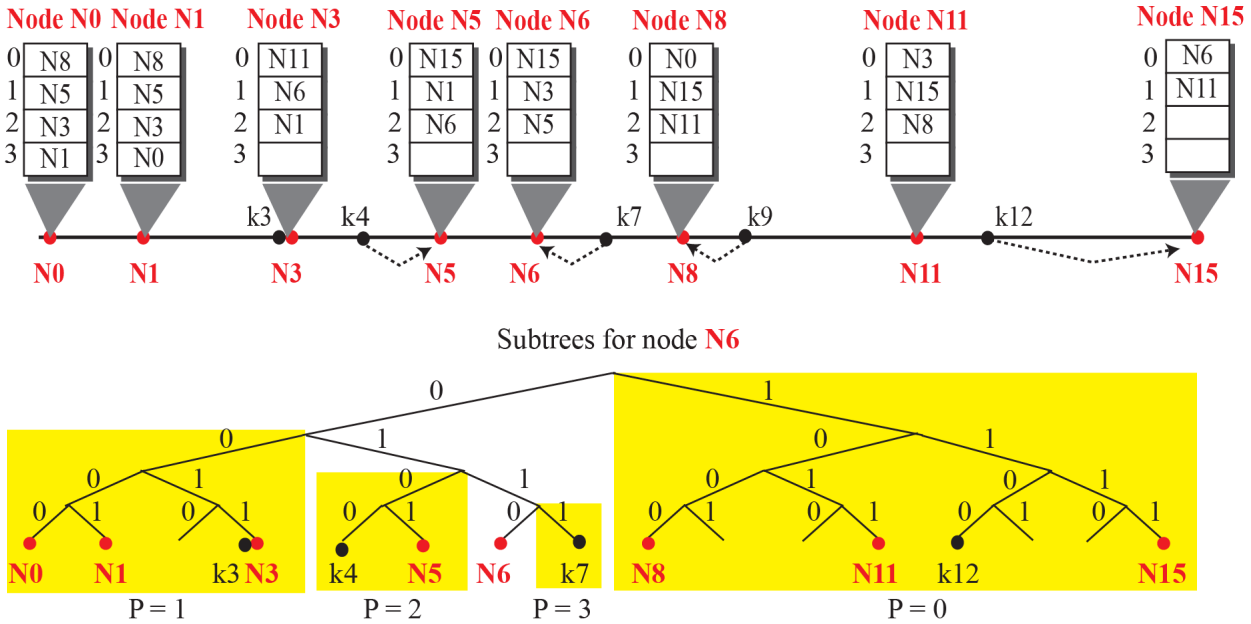
[Table 2.21: Routing table for a node in Kademlia]

N6면 0110인데 common prefix 0은 1~~~인 애들에서 closest
common prefix 1은 0~~~인 애들에서 closest

2.4 P2P Paradigm

Kademlia: Routing Table (2/3)

- With m=4, each node has four subtrees corresponding to 4 rows in the routing table
- In case of **row 0** (common prefix p=0) in **N6**, N15 inserted in row 0 because N15 is the closest to N6 (N6⊕N15 = 9, N6⊕N8 = 14, N6⊕N11 = 13)

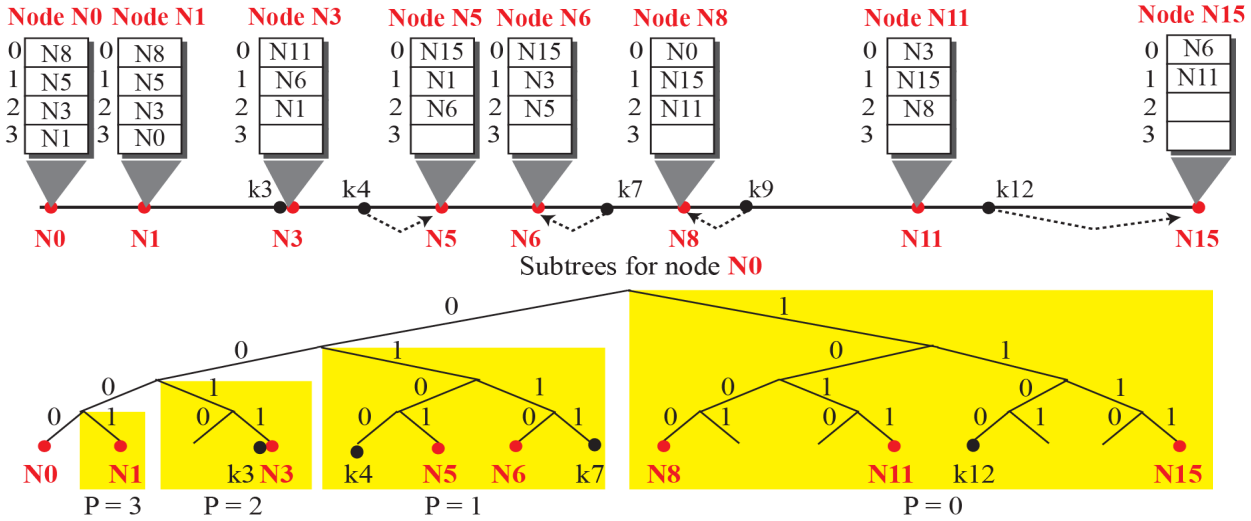


[Figure 2.56: Example 2.24]

2.4 P2P Paradigm

Kademlia: Routing Table (3/3)

- Assume that node **N0** receives a lookup message to find the node responsible for **k12**
- The length of the common prefix between **N0**, **k12** is 0; so node N0 sends the message to the node in row 0 of its routing table, node **N8**
- The length of the common prefix between **N8** and **k12** is 1; so node N8 sends the message to the node in row 1, node **N15**, which is responsible for k12



[Figure 2.56: Example 2.24]

2.4 P2P Paradigm

Kademlia: k-Buckets

- For more efficiency, Kademlia requires that each row keeps at least up to k (around 20) nodes from the corresponding subtree
  - Each row in the routing table is referred to as a k-bucket
  - Redundancy is for tolerance of the leaving or failure of nodes
- Parallel Query
  - Kademlia allows sending α parallel queries to α nodes at the top of the k-bucket for reducing the delay in case some node fails
- Concurrent Updating
  - Whenever a node receives a query or a response, it updates its k-bucket
  - If multiple queries to a node receive no response, the sender of the query removes the destination node from the corresponding k-bucket

2.4 P2P Paradigm

Kademlia: Join / Leave or Fail

- Node joins the network
  - It needs to know at least one other node
  - It sends its identifier to the known node as if it is a key to be found; and the response allows it to create its k-bucket

- Node leaves or fails
  - When a node leaves the network or fails, other nodes update their k-buckets using the aforementioned concurrent process

2.4 P2P Paradigm

조각조각을 다운받아서 서로 공유함으로서 원래 source에 traffic이 몰리지 않게 한다.

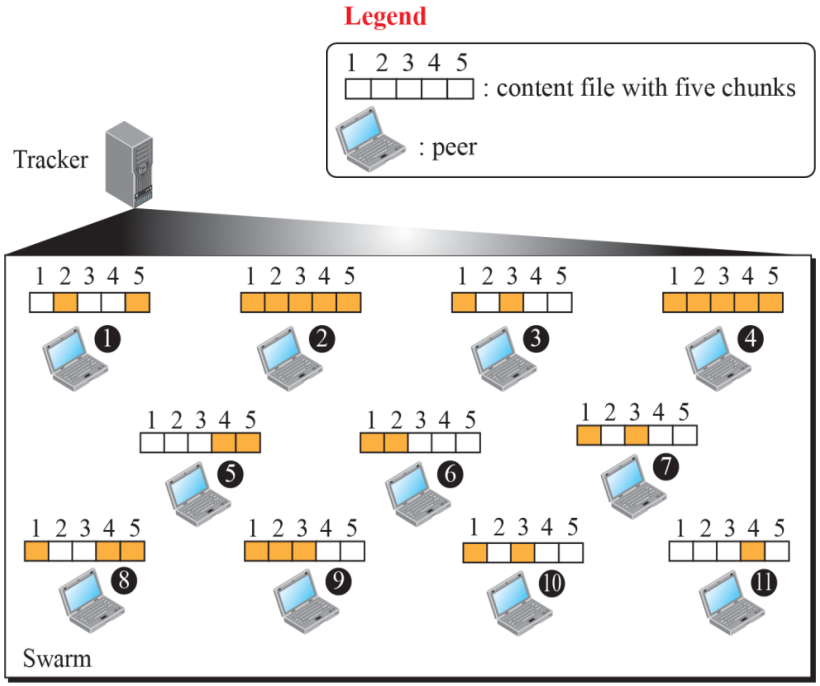
A Popular P2P Network: BitTorrent (1/3)

- **BitTorrent** is a P2P protocol for sharing a large file among a set of peers
- File sharing is done in a collaborating process called a torrent
- Each peer participating in a torrent downloads chunks of the large file from another peer (that has them) and uploads chunks of that file to other peers
- Set of all peers in a torrent is referred to as a swarm
  - A peer that has the complete content file is called a seed
  - A peer that has only part of the file and wants to download the rest is called a leech 거머리
- BitTorrent has gone through several versions and implementations
  - Original one using a central node, called tracker
  - New versions without tracker

2.4 P2P Paradigm

A Popular P2P Network: BitTorrent (2/3)

- BitTorrent with a Tracker
  - A *Tracker* is a server that assists in the communication between peers using BitTorrent protocol
  - To download a data file, a peer
    - Must have a torrent file, from BitTorrent server, containing information of pieces in the content file and address of the tracker handling that specific torrent
    - Contacts the Tracker to get information of seeds and leeches in the torrent
    - Starts downloading data chunks from seeds and leeches



Note: Peers 2 and 4 are seeds; others are leeches.

[Figure 2.57 Example of a torrent with Tracker]

2.4 P2P Paradigm

A Popular P2P Network: BitTorrent (3/3)

- Trackerless BitTorrent
  - The job of tracking is distributed among some nodes in the network
    - Some nodes play the role of trackers
  - Kademlia DHT can be used for the implementation
    - *Key*: metadata file that defines the torrent
    - *Value*: list of peers in the torrent (seeds and leeches)
    - The Kademlia protocol finds the node responsible for a given key, and then returns the list of peers in the corresponding torrent
      - A joining peer can use the BitTorrent protocol to share/download the content file with/from peers in the list

Practice Problems

1. In a DHT-based network, what is the size of peer-identifier space? And what is the range of values in this space?

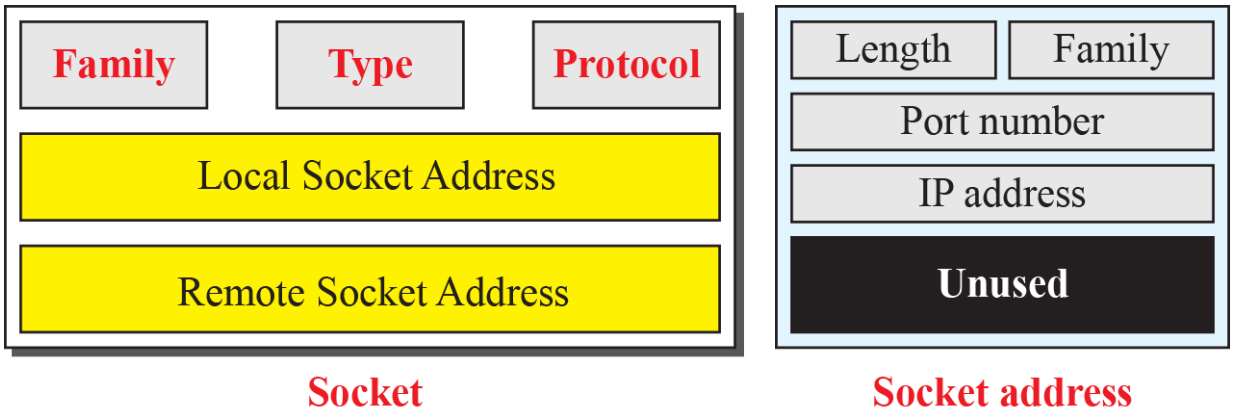
   2^m (each identifier is m - bit log)
   0 ~ (2^m - 1)

2. In Kademlia, assume *m*=4 and active nodes are N4, N7, and N12. Where is the key k3 stored in this system?

2.5 Socket-Interface Programming

Socket Interface in C: Data Structure for Socket (1/2)

- The socket structure is made of five fields
  - **Family**: This field defines the family protocol such as PF_INET (IPv4 internet protocol), PF_INET6 (IPv6 internet protocol)
  - **Type**: This field defines four types of socket such as SOCK_STREAM (for TCP), SOCK_DGRAM (for UDP), SOCK_SEQPACKET (for SCTP), and IP (for applications that directly use the services of IP)

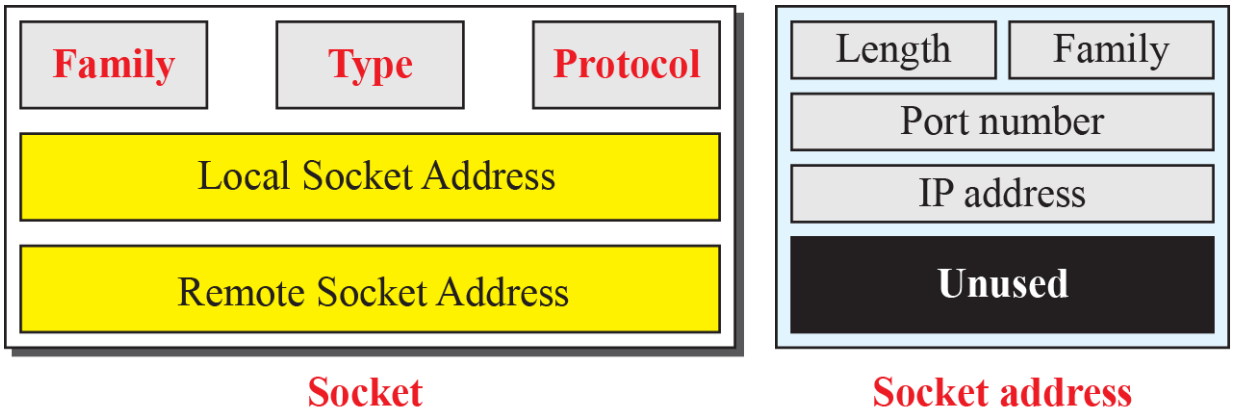


[Figure 2.58 Socket data structure]

2.5 Socket-Interface Programming

Socket Interface in C: Data Structure for Socket (2/2)

- **Protocol**: This field defines the specific protocol in the family
  - It is set to 0 for TCP/IP protocol suite
- **Local Socket Address**: it includes the *length* field, the *family* field (AF_INET), the *port number* field, and the *IP address* field
- **Remote Socket Address**: Its structure is the same as the local socket address

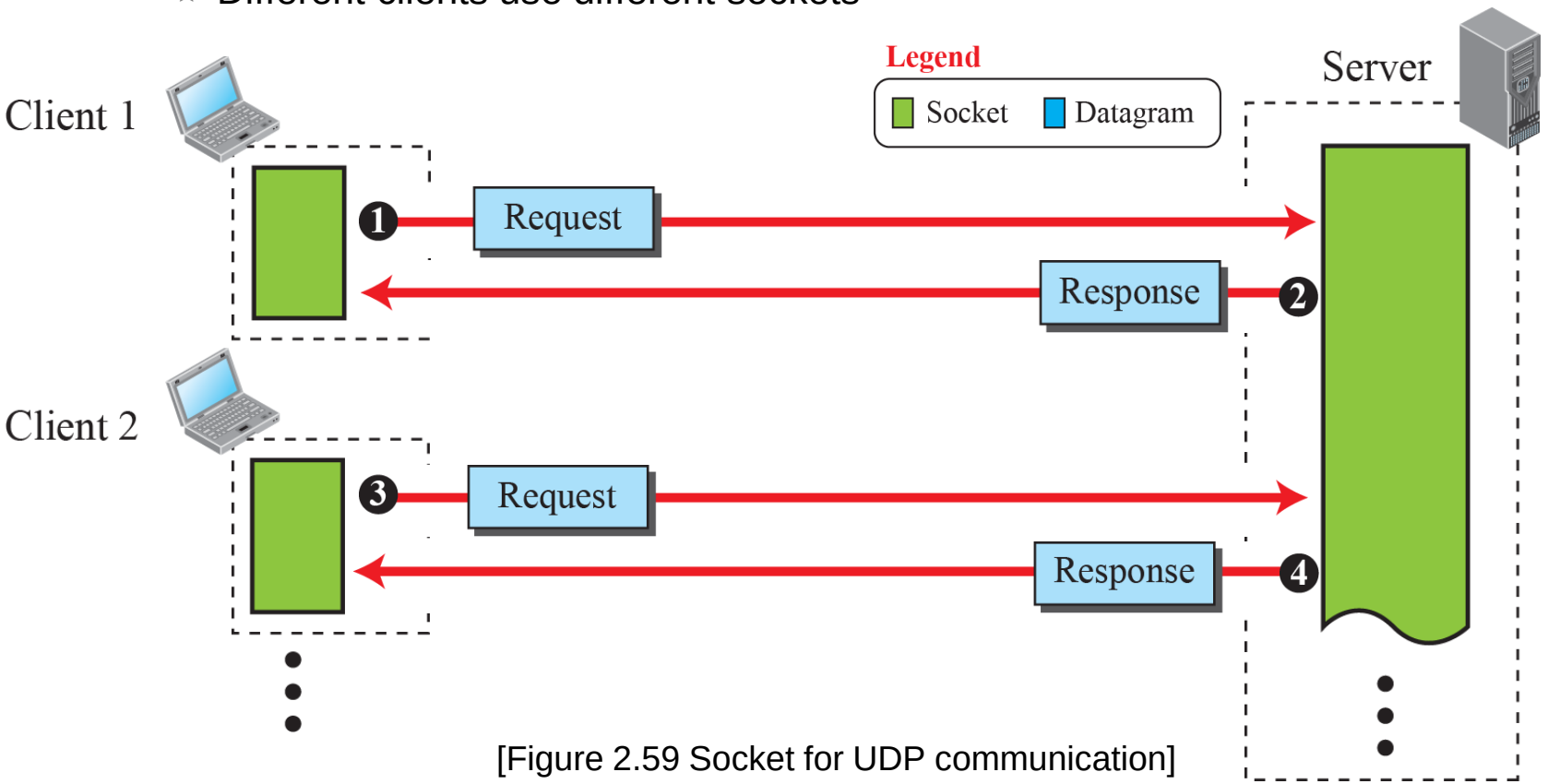


[Figure 2.58 Socket data structure]

2.5 Socket-Interface Programming

Socket Interface in C: Iterative Communication Using UDP (1/2)

- Sockets used for UDP
  - In UDP communication, the client and server use only one socket each
    - Different clients use different sockets

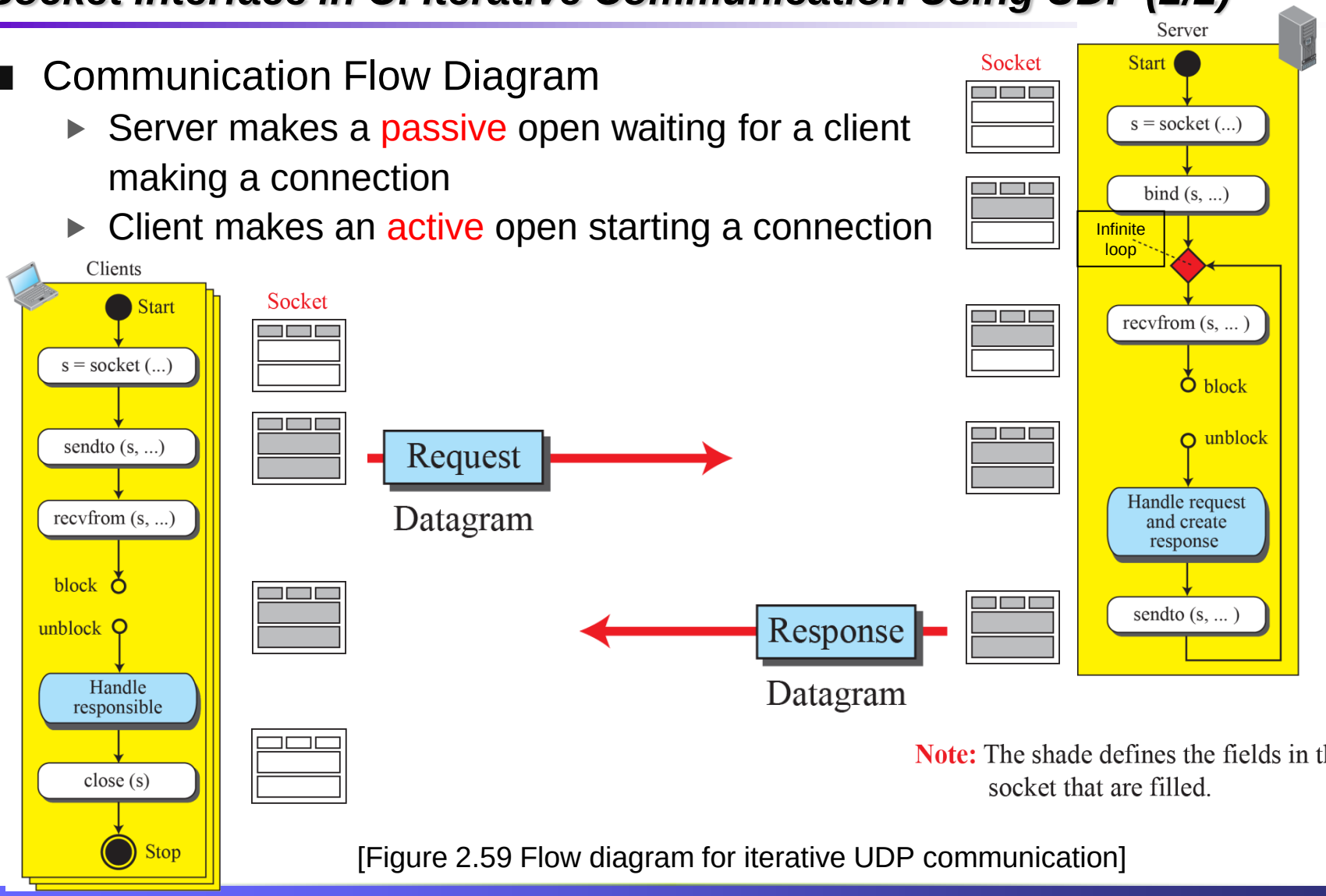


[Figure 2.59 Socket for UDP communication]

2.5 Socket-Interface Programming

Socket Interface in C: Iterative Communication Using UDP (2/2)

- Communication Flow Diagram
  - Server makes a passive open waiting for a client making a connection
  - Client makes an active open starting a connection



[Figure 2.59 Flow diagram for iterative UDP communication]

2.5 Socket-Interface Programming

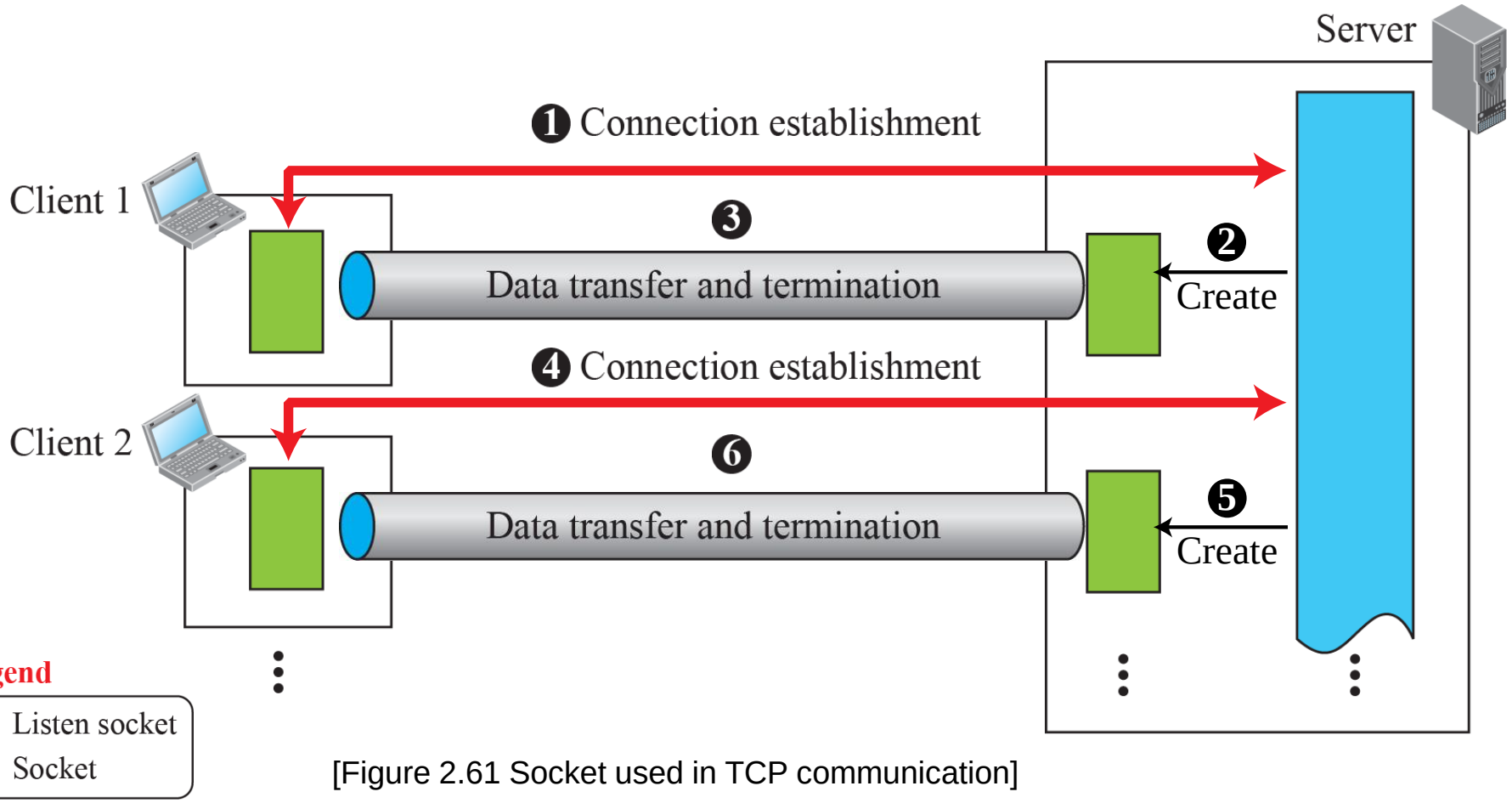
Socket Interface in C: Communication Using TCP (1/3)

- TCP is a connection-oriented protocol
  - Before sending data, a connection needs to be established between the client and server
- TCP communication can be iterative (serving a client at a time) or concurrent (serving several clients at a time)

- Sockets used in TCP
  - TCP server uses two different sockets
    - Listen socket: for listening and establishing connection from client
    - Socket: for exchanging data with the client
      - This socket is created after the connected established

2.5 Socket-Interface Programming

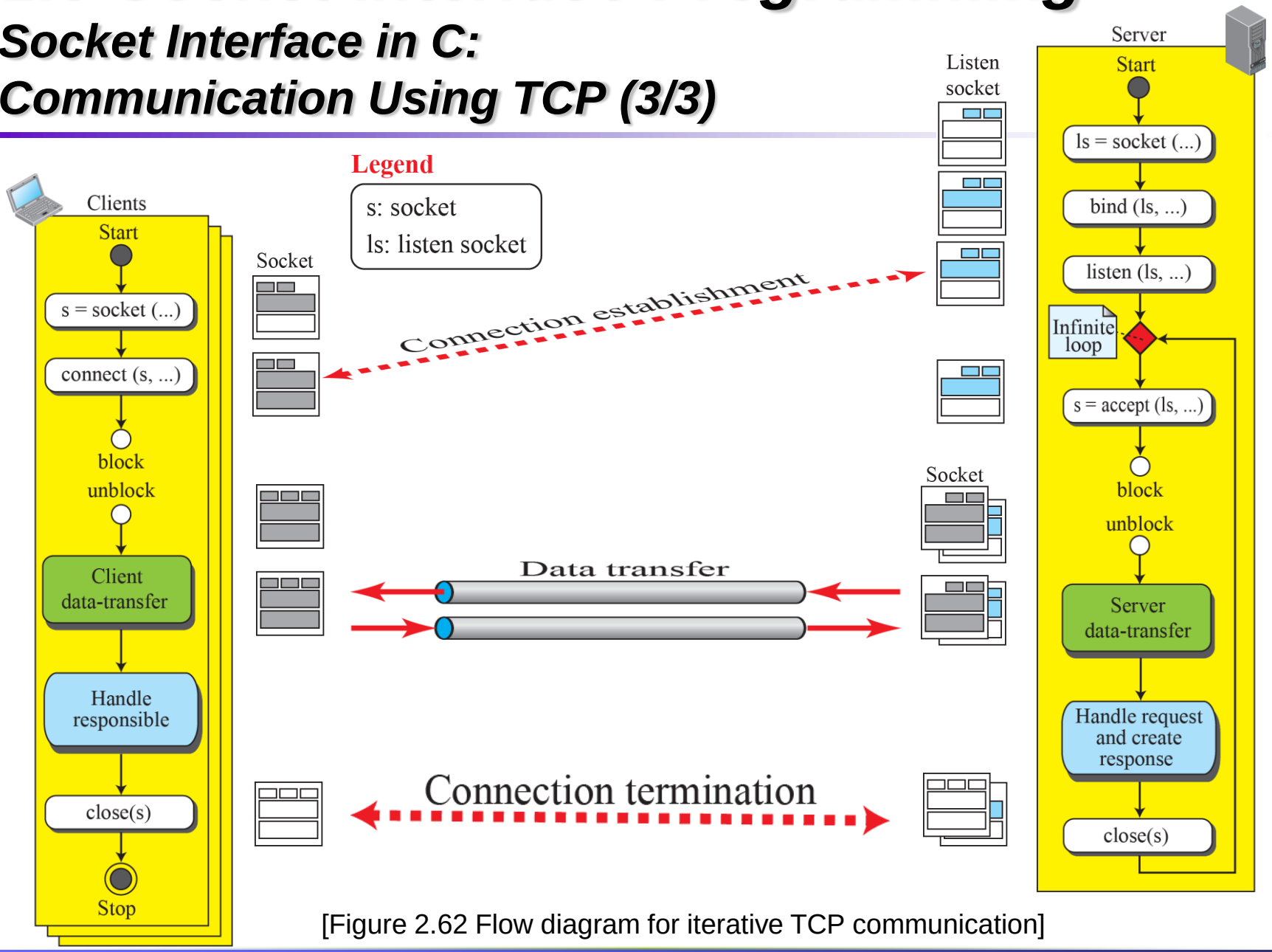
Socket Interface in C: Communication Using TCP (2/3)



[Figure 2.61 Socket used in TCP communication]

2.5 Socket-Interface Programming

Socket Interface in C: Communication Using TCP (3/3)



[Figure 2.62 Flow diagram for iterative TCP communication]

Practice Problems

1. What is the purpose of Listen Socket in TCP?

2. The UDP server usually only needed one socket, whereas the TCP server needed at least two sockets. Why? If the TCP server were to support *n* simultaneous connections, each from a different client host, how many sockets would the TCP server need?

UDP는 하나로 다