UNIT V

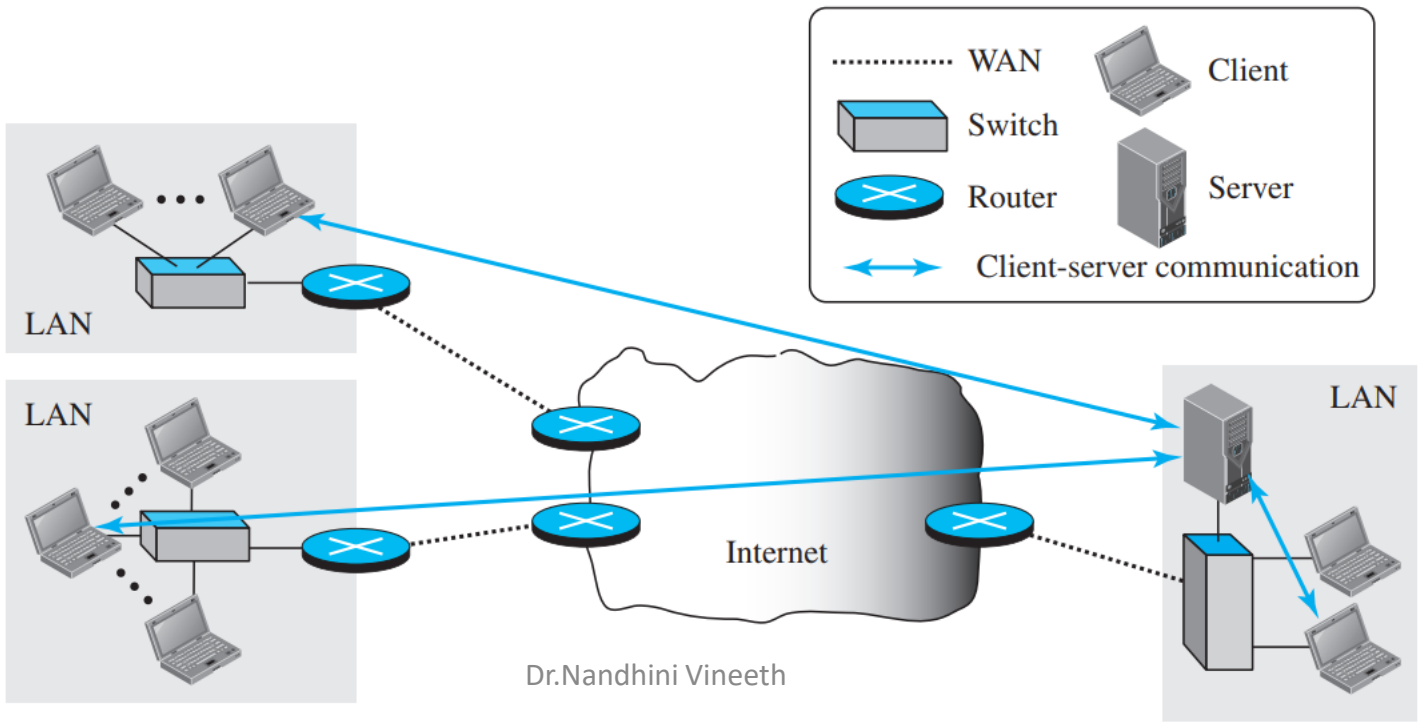
Introduction to Application Layer

INTRODUCTION

- Standard and Nonstandard Protocol
- Standard Protocol:
  - several application-layer protocols that have been standardized and documented by the Internet authority, and we are using them in our daily interaction with the Internet.
  - The study of these protocols enables a network manager to easily solve the problems that may occur when using these protocols
- Non Standard Protocol:
  - If she can write two programs that provide service to the user by interacting with the transport layer.
  - **A private company** can create a new customized application protocol to communicate with all of its offices around the world using the services provided by the first four layers of the TCP/IP protocol suite without using any of the standard application program

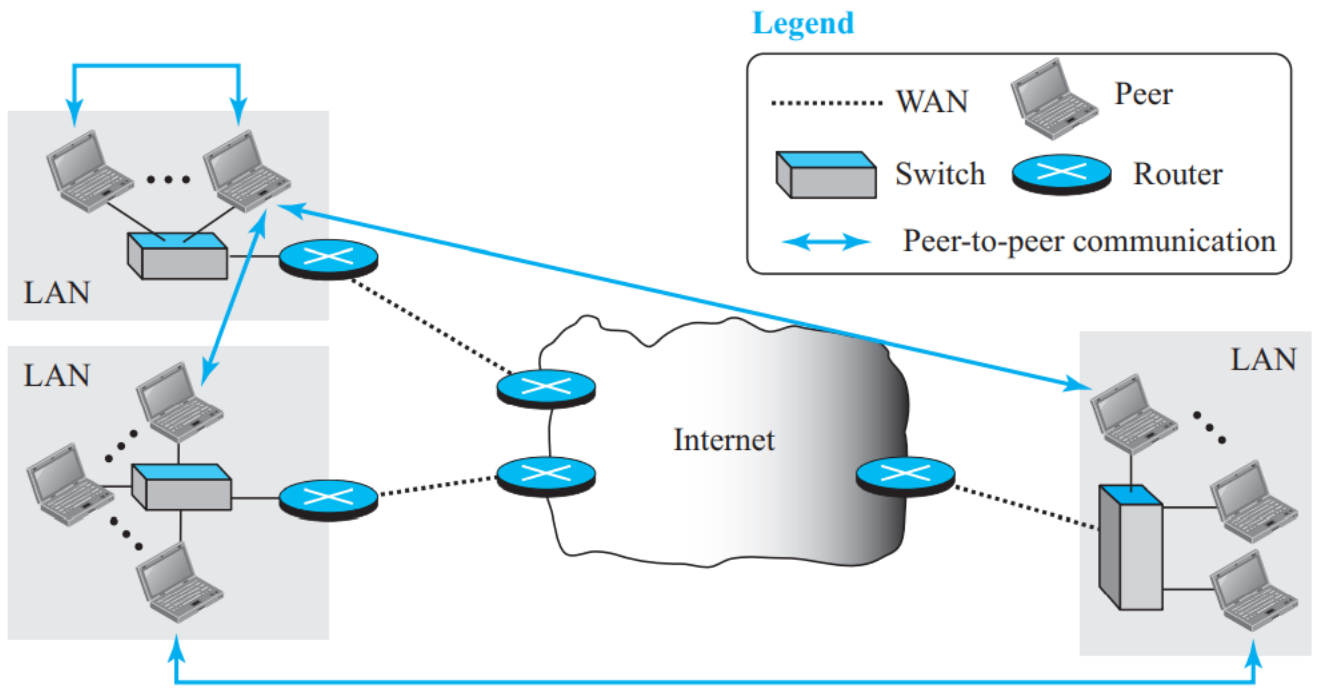
- Application-Layer Paradigms
- Traditional Paradigm: Client-Server

Figure 25.2 *Example of a client-server paradigm*



- New Paradigm: Peer-to-Peer

Figure 25.3 *Example of a peer-to-peer paradigm*



Mixed Paradigm is also encouraged

WORLD WIDE WEB AND HTTP

- web pages, hypertext, hypermedia

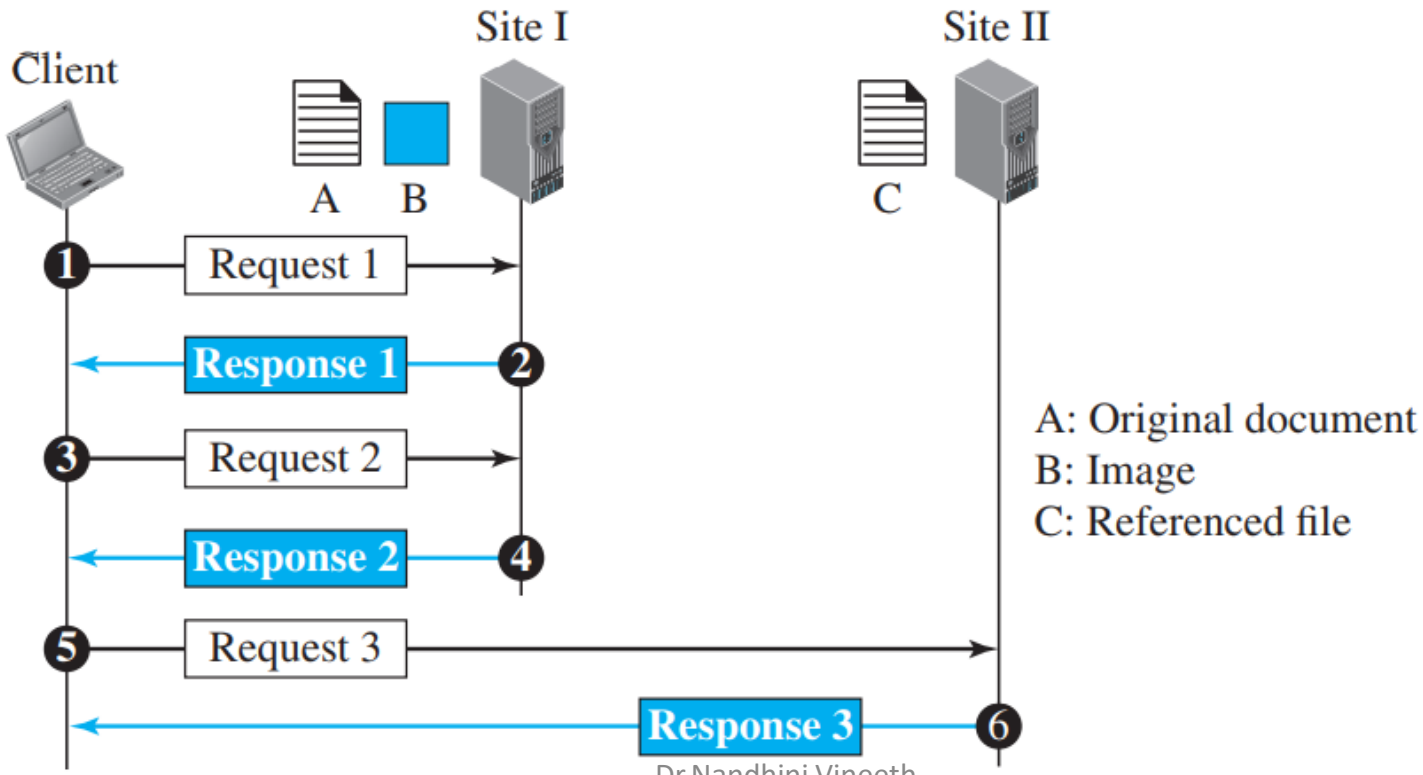
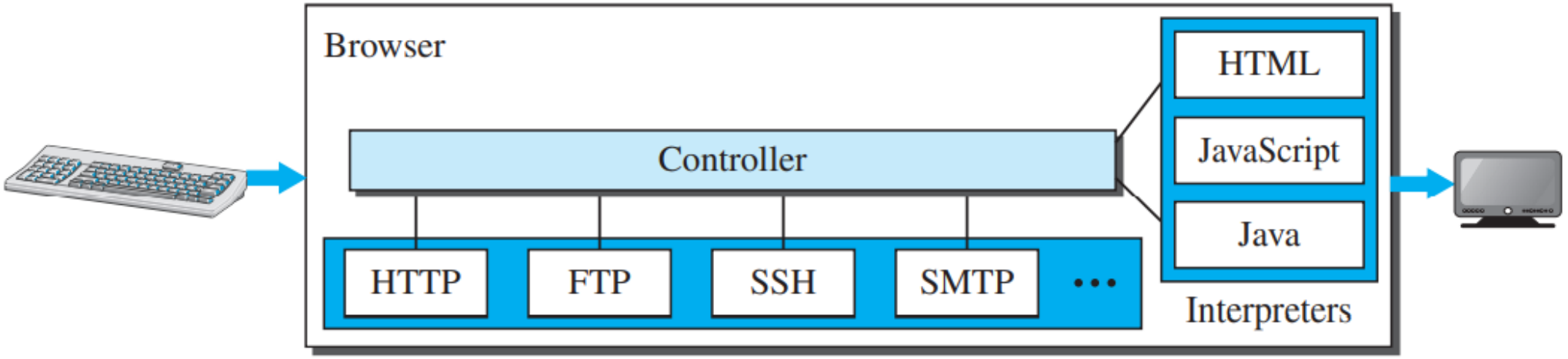


Figure 26.2 *Browser*



- Uniform Resource Locator (URL)
  - Protocol.
  - Host – IP Address
  - Port.
  - Path.
- Web Documents
  - Static
  - Dynamic

HyperText Transfer Protocol (HTTP)

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

Figure 26.3 *Example 26.3*

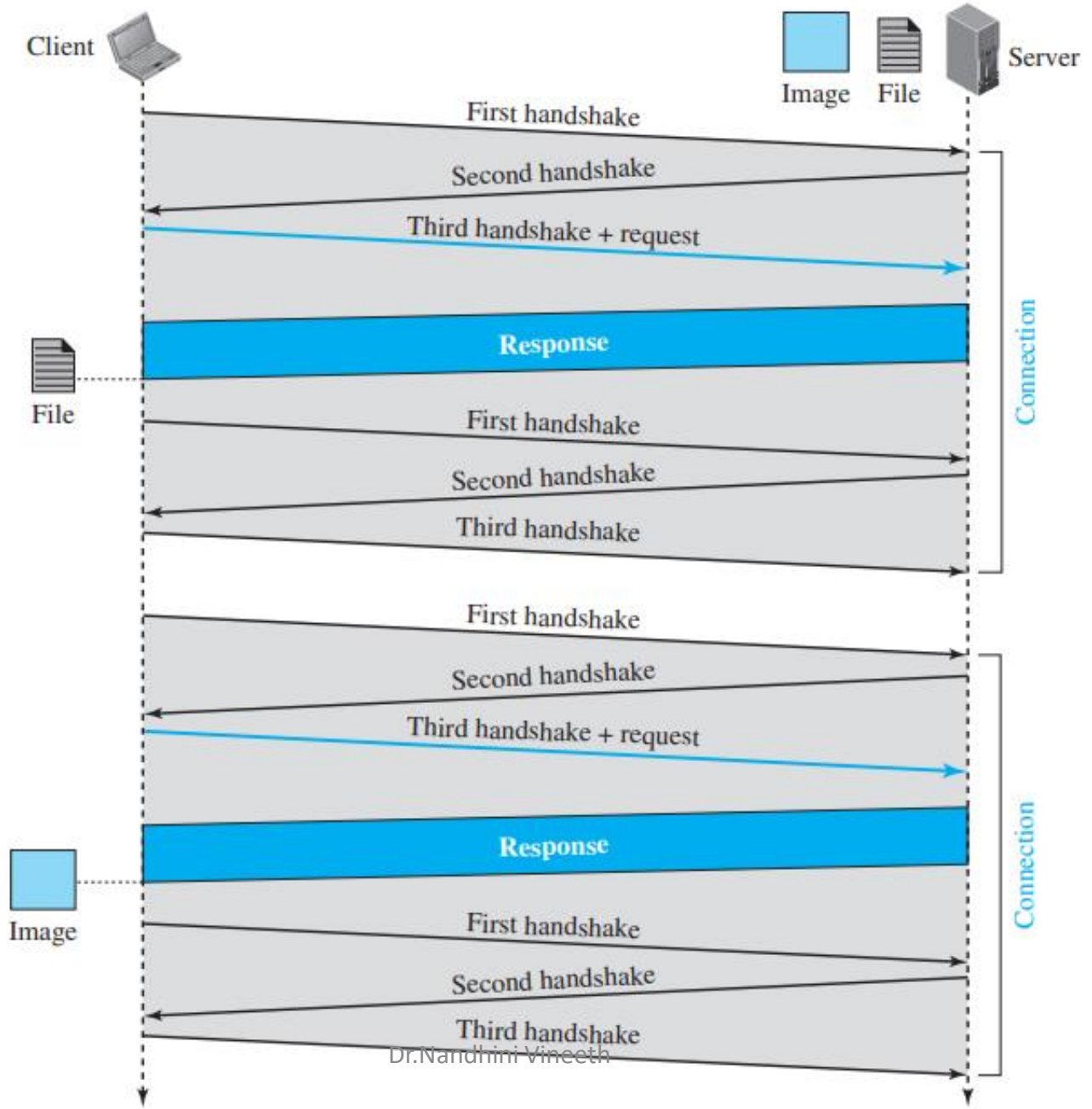


Figure 26.4 *Example 26.4*

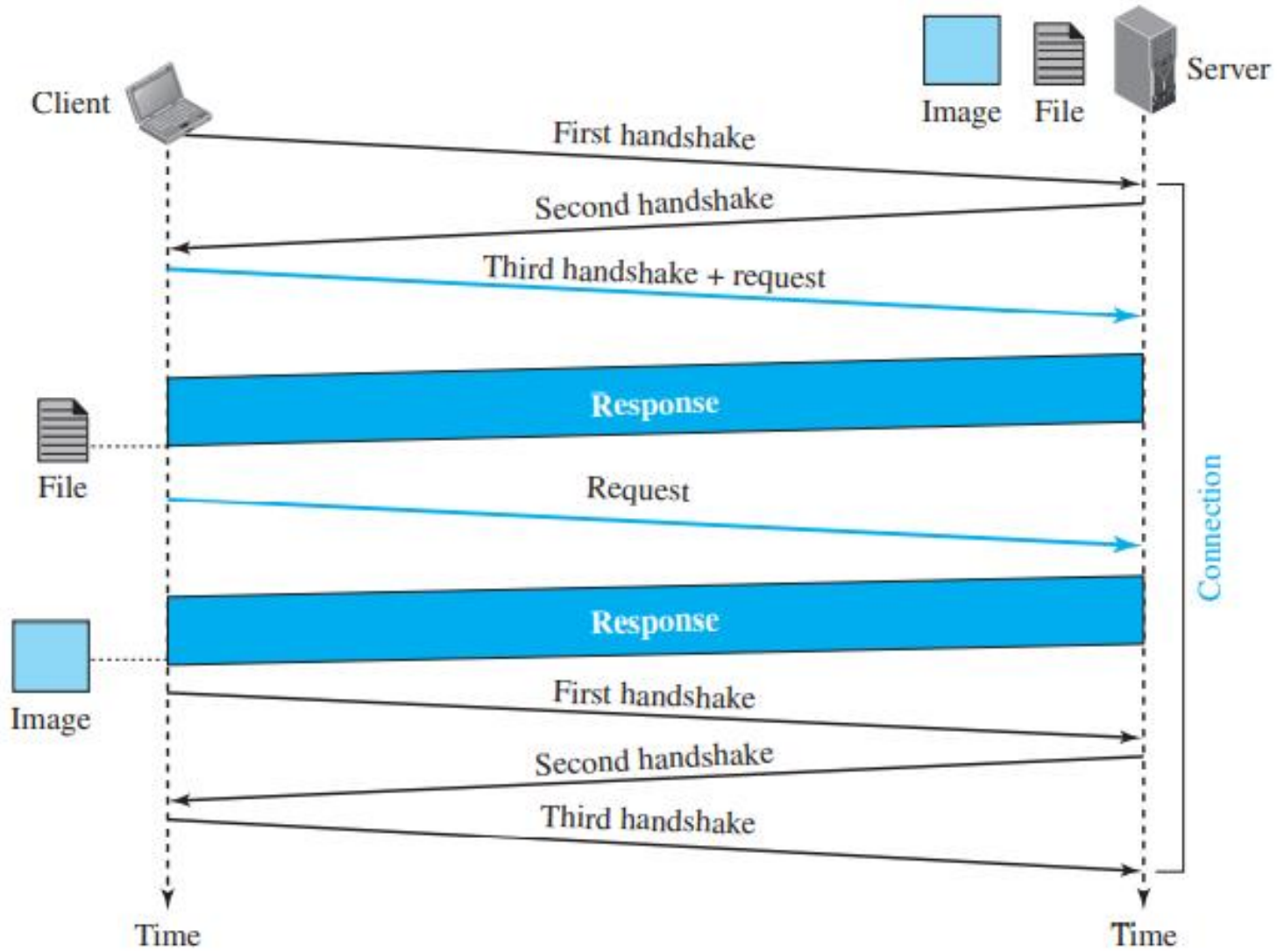


Figure 26.5 *Formats of the request and response messages*

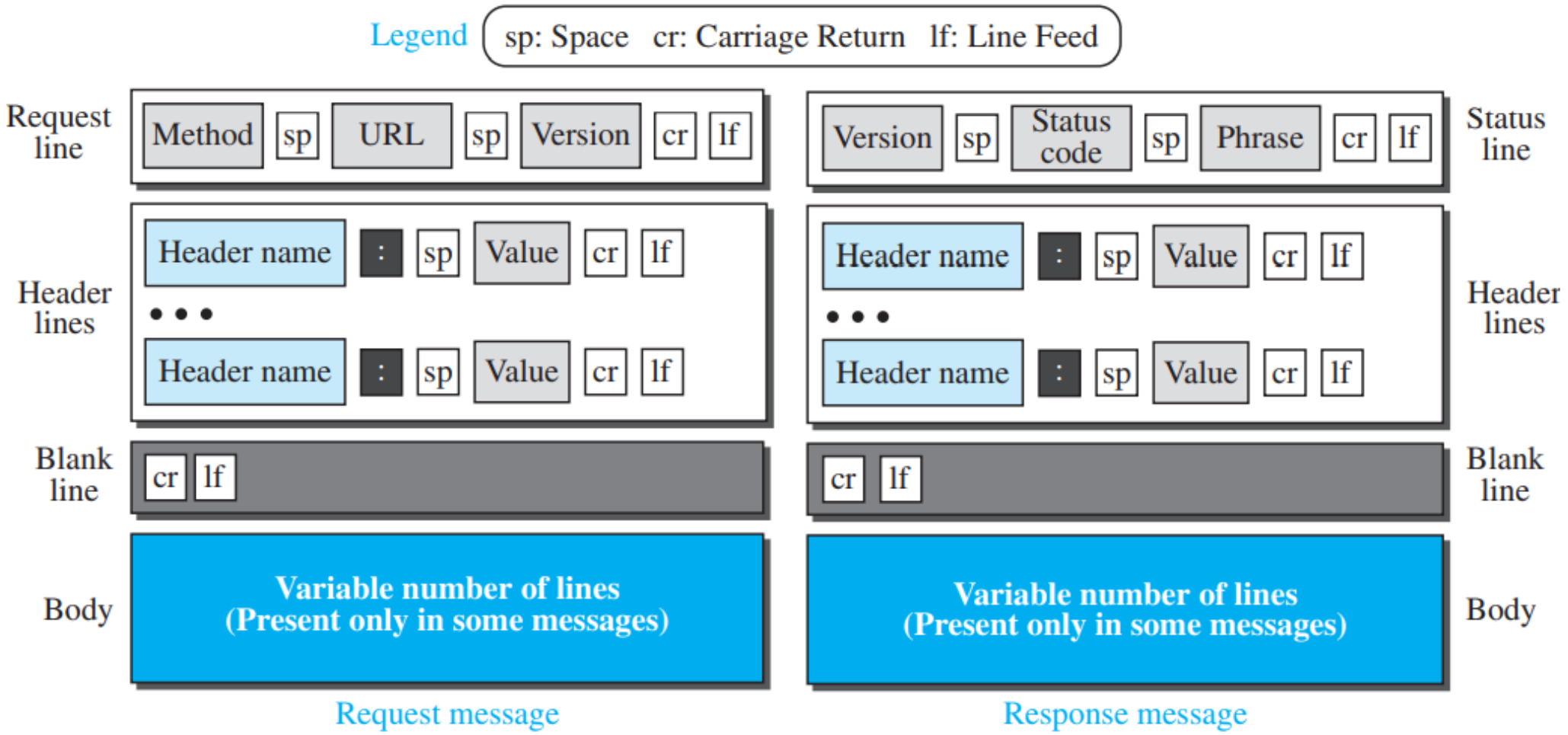


Table 26.1 *Methods*

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

Table 26.2 *Request header names*

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

Table 26.3 *Response header names*

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

Figure 26.6 *Example 26.5*

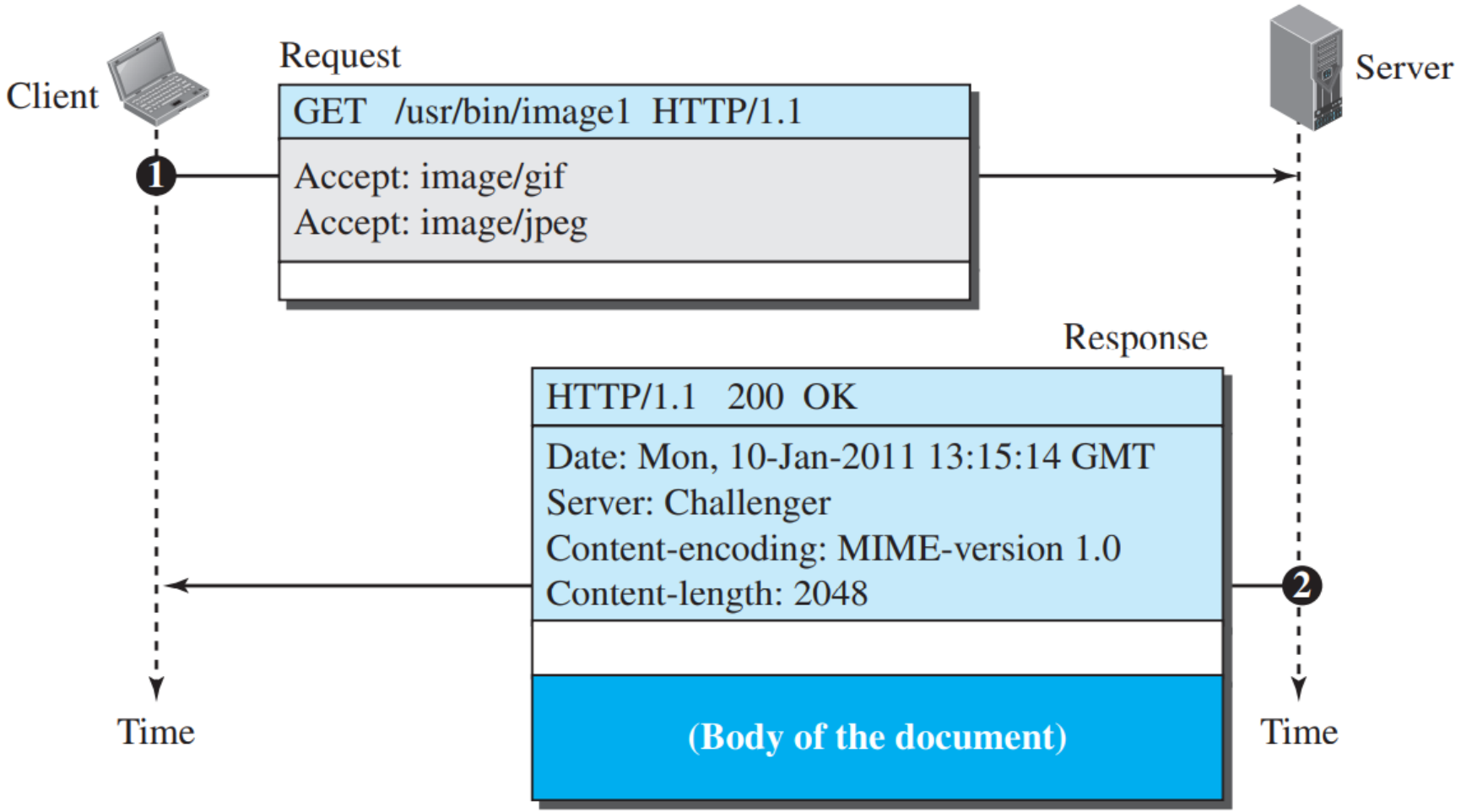
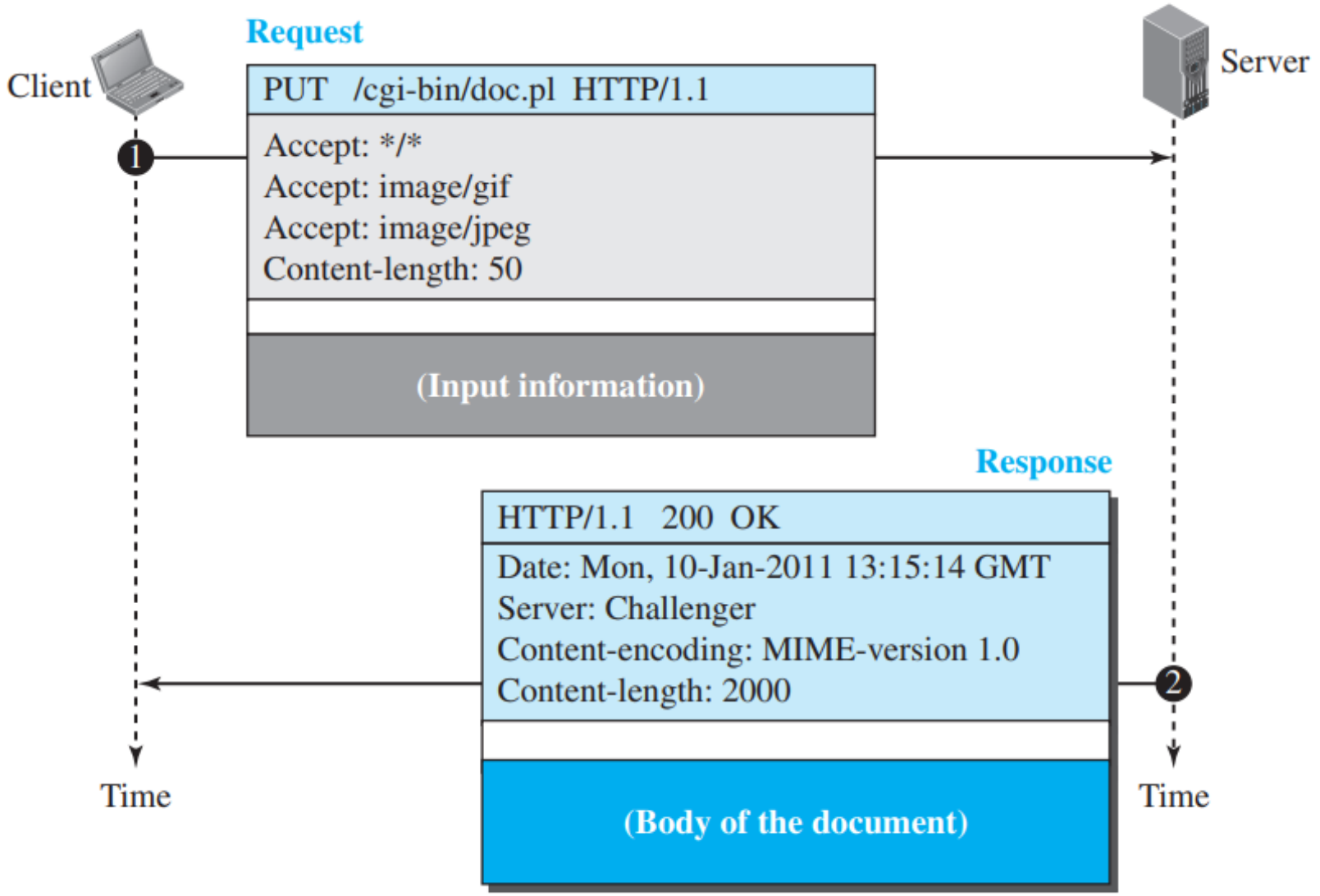


Figure 26.7 *Example 26.6*



Cookies

- electronic store, registered clients, portal(Favourites), advertising
- Creating and Storing Cookies
- Using Cookies

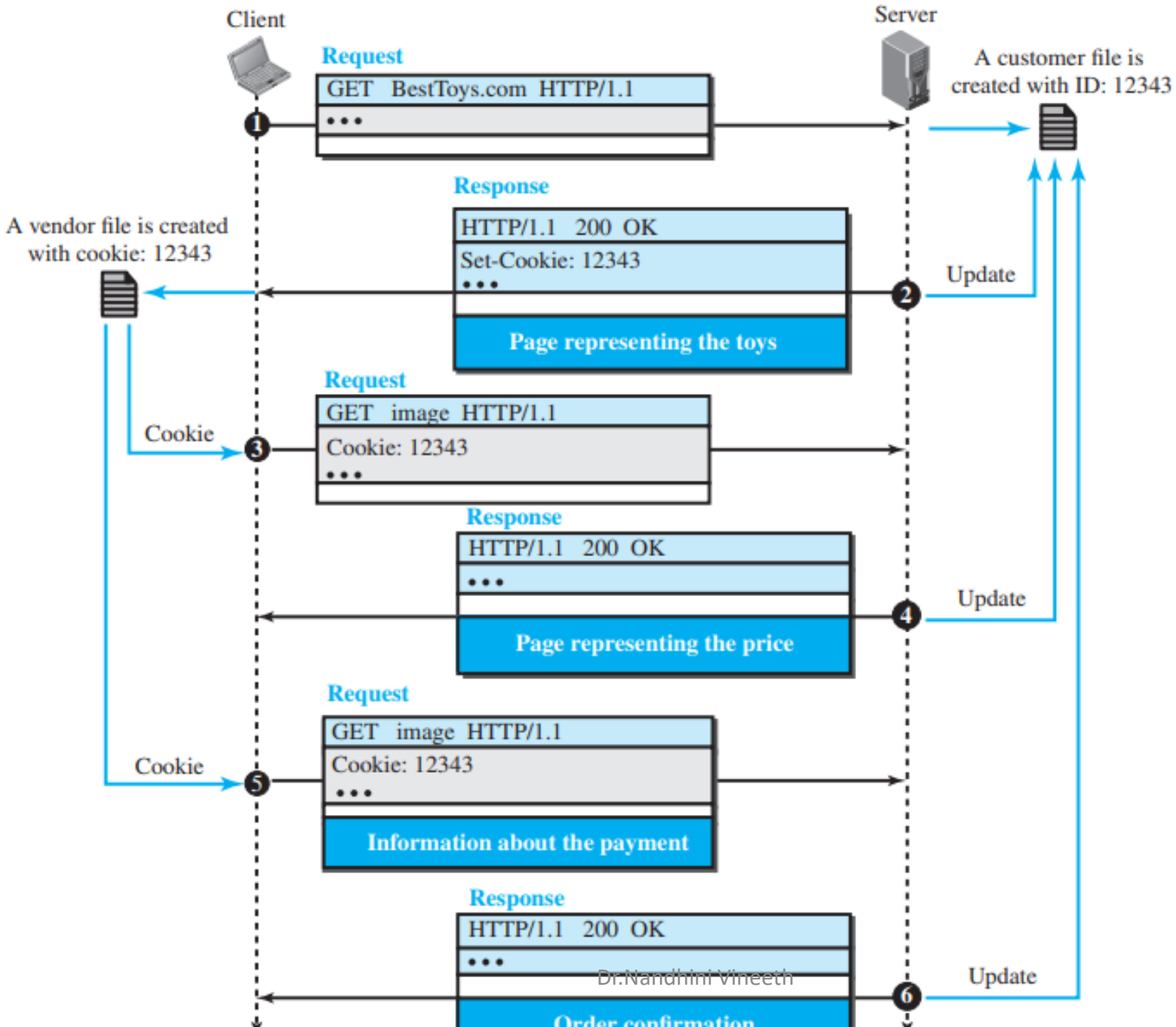
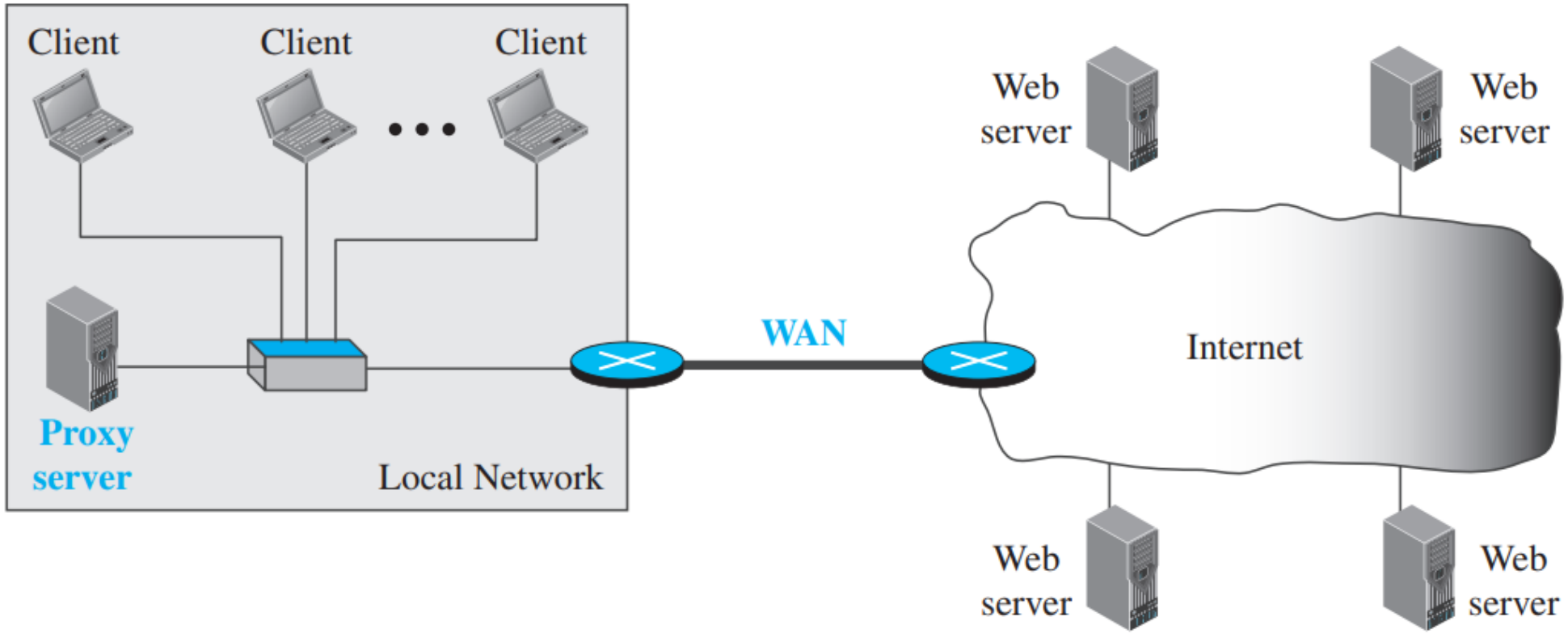
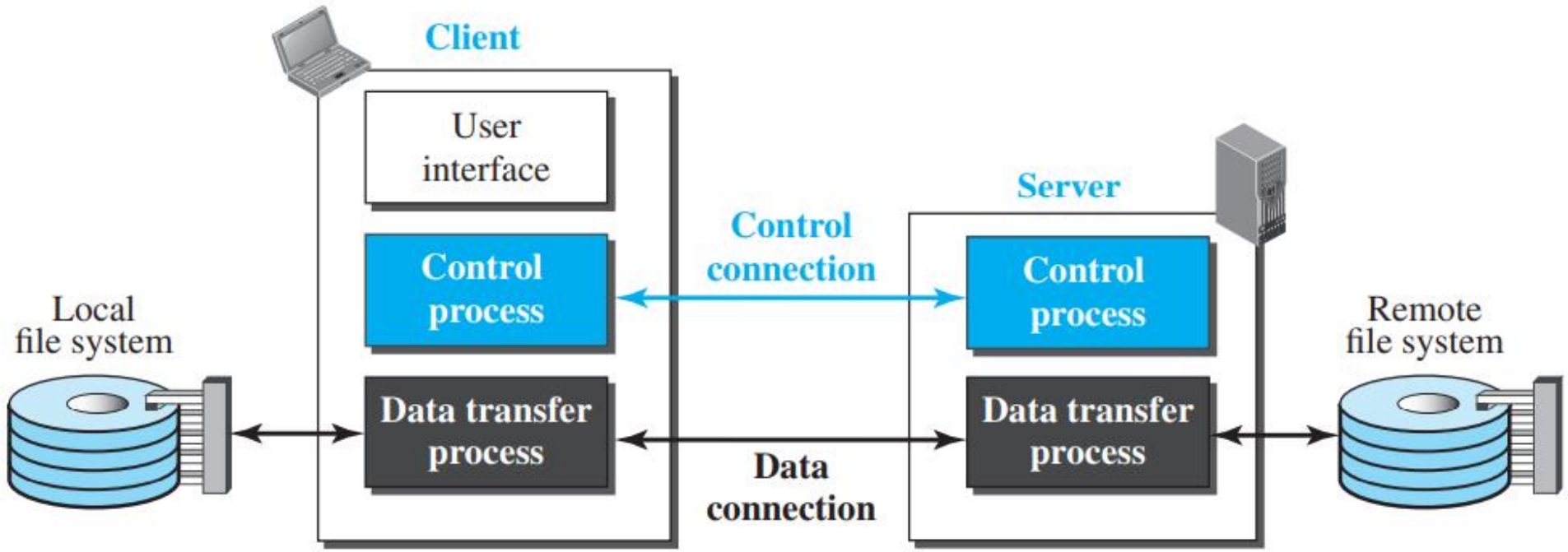

Client
Server
Request
GET BestToys.com HTTP/1.1
A customer file is created with ID: 12343
Response
HTTP/1.1 200 OK
Set-Cookie: 12343
A vendor file is created with cookie: 12343
Update
Page representing the toys
Request
GET image HTTP/1.1
Cookie: 12343
Cookie
Response
HTTP/1.1 200 OK
Update
Page representing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

Figure 26.9 *Example of a proxy server*



- Cache update

Figure 26.10 *FTP*



FTP can transfer a file across the data connection using one of the following interpretations of the structure of the data: file structure, record structure, or page structure.

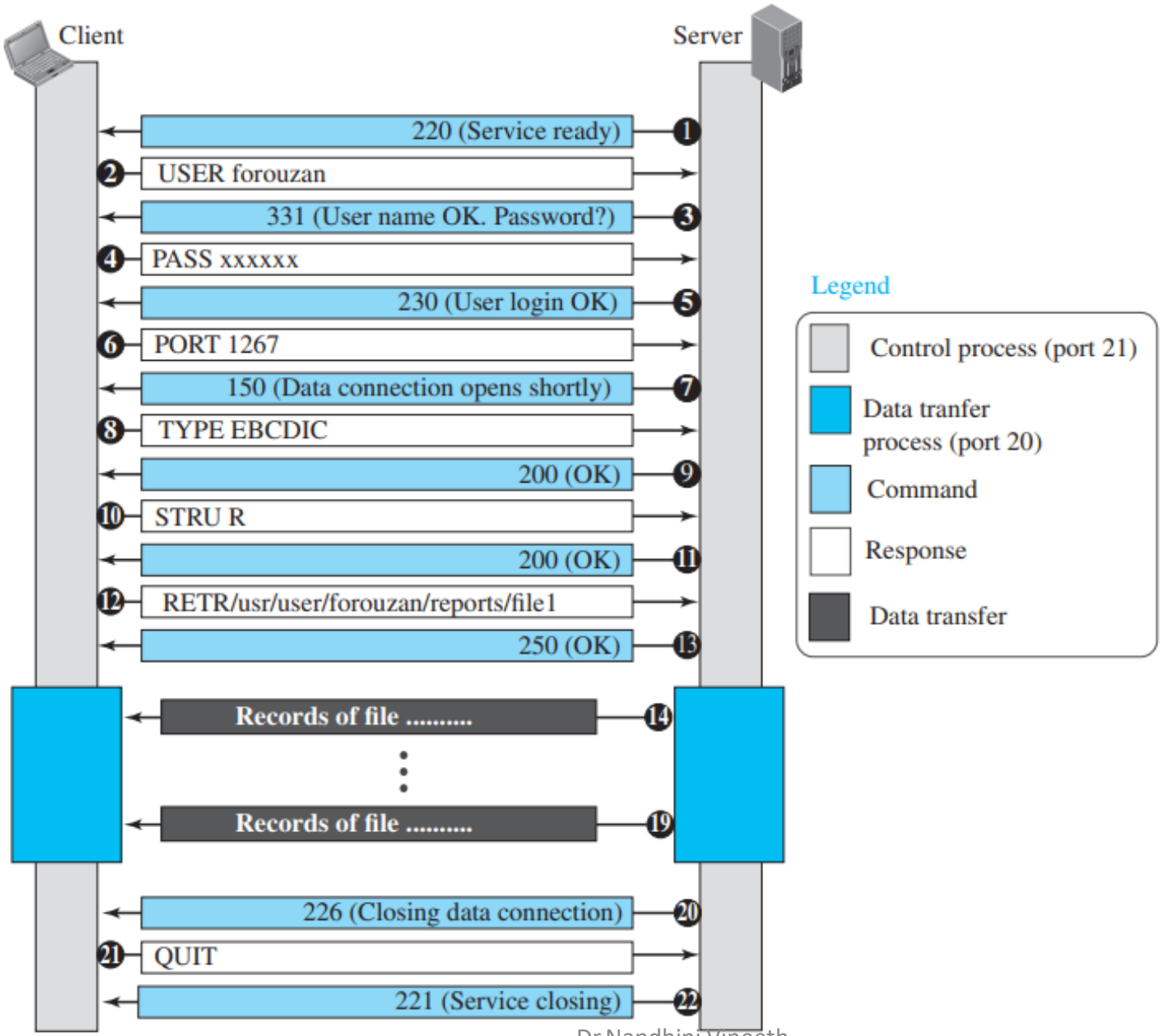
Transmission Mode FTP can transfer a file across the data connection using one of the following three transmission modes: **stream mode, block mode, or compressed mode**. The stream mode is the default mode; data are delivered from FTP to TCP as a continuous stream of bytes. In the block mode, data can be delivered from FTP to TCP in blocks. In this case, each block is preceded by a 3-byte header. The first byte is called the block descriptor; the next two bytes define the size of the block in bytes.

Table 26.4 *Some FTP commands*

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

Table 26.5 *Some responses in FTP*

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


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

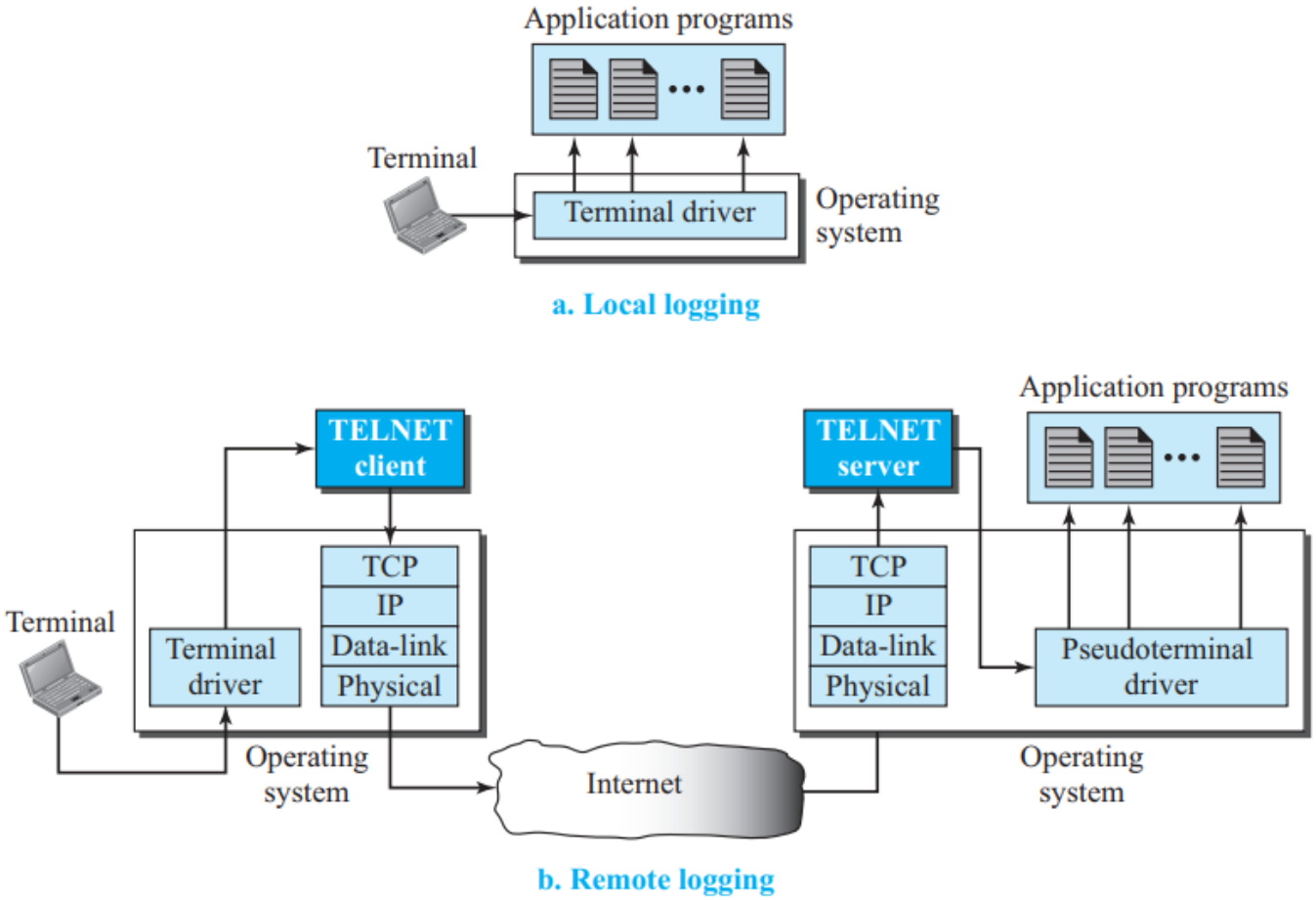
```
$ ftp voyager.deanza.fhda.edu
Connected to voyager.deanza.fhda.edu.
220 (vsFTPd 1.2.1)
530 Please login with USER and PASS.
Name (voyager.deanza.fhda.edu:forouzan): forouzan
331 Please specify the password.
Password:*********
230 Login successful.
Remote system type is UNIX.
Using binary mode to 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.
```

TELNET

- one of the **original remote logging protocols** is TELNET, which is an abbreviation for TErminaL NETwork.
- Although TELNET requires a logging name and password, it is vulnerable to hacking because it sends all data including the password in plaintext (not encrypted). A hacker can eavesdrop and obtain the logging name and password.
- Although TELNET is almost replaced by SSH, we briefly discuss TELNET here for two reasons:
- 1. The simple plaintext architecture of TELNET allows us to explain the issues and challenges related to the concept of remote logging, which is also used in SSH when it serves as a remote logging protocol.
- 2. Network administrators often use TELNET for diagnostic and debugging purposes.

TELNET -Local versus Remote Logging

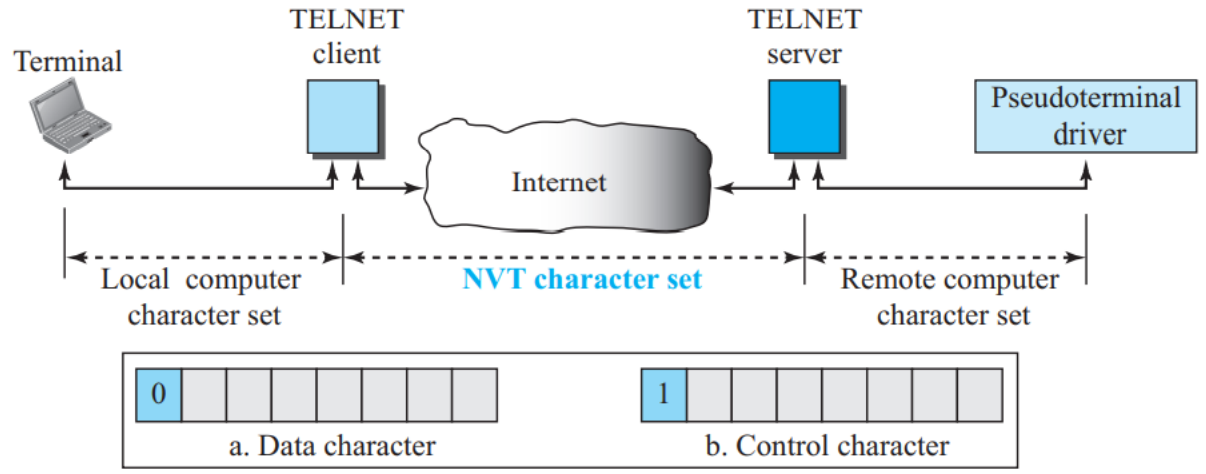
Figure 26.23 *Local versus remote logging*



TELNET

- The TELNET client, transforms the characters into a universal character set called Network Virtual Terminal (NVT) characters and delivers them to the local TCP/IP stack.

Figure 26.24 *Concept of NVT*



TELNET

Table 26.11 *Examples of interface commands*

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

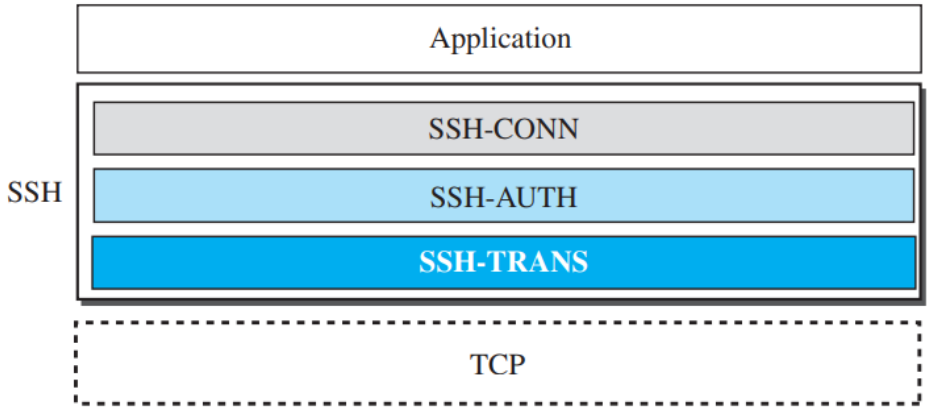
SECURE SHELL

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

SSH Transport-Layer Protocol (SSH-TRANS)

1. Privacy or confidentiality of the message exchanged
2. Data integrity, which means that it is guaranteed that the messages exchanged between the client and server are not changed by an intruder
3. Server authentication, which means that the client is now sure that the server is the one that it claims to be
4. Compression of the messages, which improves the efficiency of the system and makes attack more difficult

Figure 26.25 *Components of SSH*



SECURE SHELL (SSH)

- SSH Authentication Protocol (SSH-AUTH)
  - SSH can call another procedure that can **authenticate the client for the server**.
  - Authentication starts with the client, which sends a request message to the server. The request includes the user name, server name, the method of authentication, and the required data. The server responds with either a success message, which confirms that the client is authenticated, or a failed message
- SSH Connection Protocol (SSH-CONN)
  - One of the services provided by the SSH-CONN protocol is **multiplexing.** SSH-CONN takes the secure channel established by the two previous protocols and lets the client create multiple logical channels over it. Each channel can be used for a different purpose, such as remote logging, file transfer, and so on.

SECURE SHELL (SSH)

- Applications:
  - SSH for Remote Logging
  - SSH for File Transfer
  - Port Forwarding

Figure 26.26 *Port forwarding*

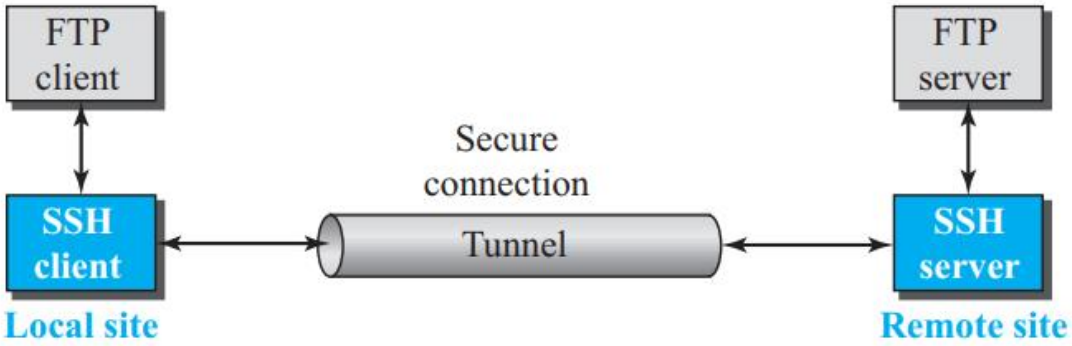
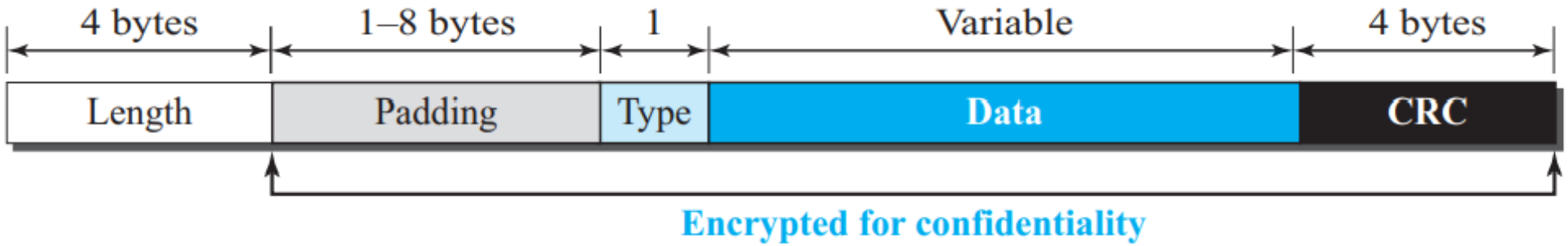


Figure 26.27 *SSH packet format*



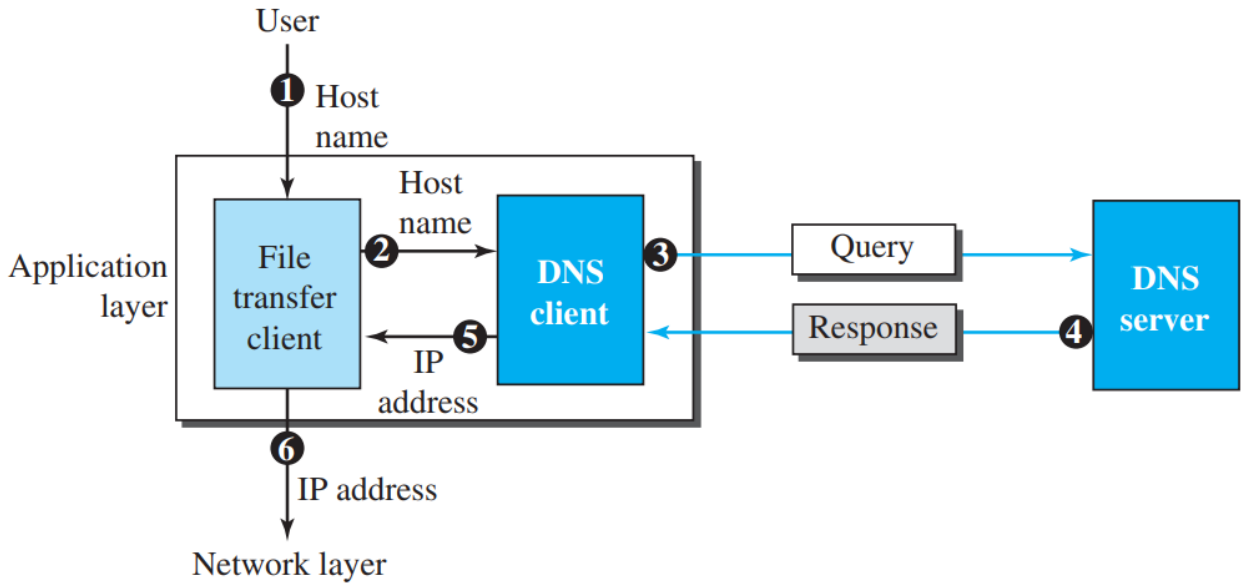
DOMAIN NAME SYSTEM (DNS)

- People prefer to use names instead of numeric addresses. Therefore, the Internet needs to have a directory system that can map a name to an address
- One central directory system cannot hold all the mapping.
  - if the central computer fails, the whole communication network will collapse.
- A better solution is to distribute the information among many computers in the world.
  - the host that needs mapping can contact the closest computer holding the needed information. This method is used by the **Domain Name System (DNS).**

DOMAIN NAME SYSTEM (DNS)

- Figure 26.28 shows how TCP/IP uses a DNS client and a DNS server to map a name to an address.
- A user wants to use a file transfer client to access the corresponding file transfer server running on a remote host. The user knows only the file transfer server name, such as afilesource.com. However, the TCP/IP suite needs the IP address of the file transfer server to make the connection

Figure 26.28 *Purpose of DNS*



DOMAIN NAME SYSTEM (DNS)

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

We need at least two connections in this case. The first is for mapping the name to an IP address; the second is for transferring files

Name Space

- To be unambiguous, the names assigned to machines must be carefully selected from a name space with complete control over the binding between the names and IP addresses. In other words, the names must be unique because the addresses are unique. A name space that maps each address to a unique name can be organized in two ways: flat or hierarchical.
  - In a flat name space, a name is assigned to an address. A name in this space is a sequence of characters without structure. The names may or may not have a common section; if they do, it has no meaning. The main disadvantage of a flat name space is that it cannot be used in a large system such as the Internet because it must be centrally controlled to avoid ambiguity and duplication.
  - In a hierarchical name space, each name is made of several parts. The first part can define the nature of the organization, the second part can define the name of an organization, the third part can define departments in the organization, and so on.

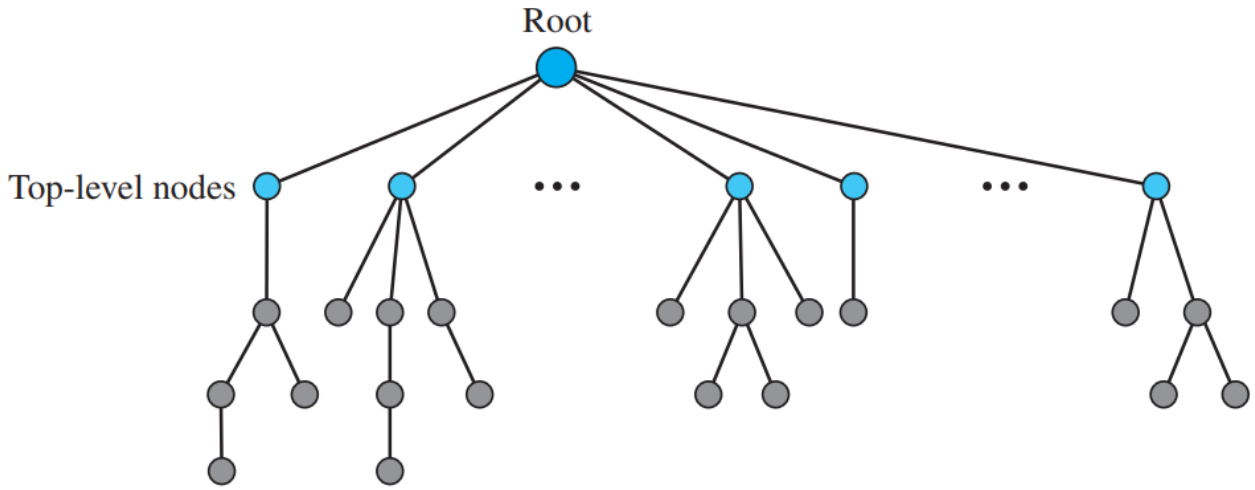
Name Space

- Hierarchical name space
- In this case, the authority to assign and control the name spaces can be decentralized. A central authority can assign the part of the name that defines the nature of the organization and the name of the organization. The responsibility for the rest of the name can be given to the organization itself. The organization can add suffixes (or prefixes) to the name to define its host or resources. The management of the organization need not worry that the prefix chosen for a host is taken by another organization because, even if part of an address is the same, the whole address is different.
- For example, assume two organizations call one of their computers caesar. The first organization is given a name by the central authority, such as first.com, the second organization is given the name second.com. When each of these organizations adds the name caesar to the name they have already been given, the end result is two distinguishable names: ceasar.first.com and ceasar.second.com. The names are unique

Domain Name Space

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

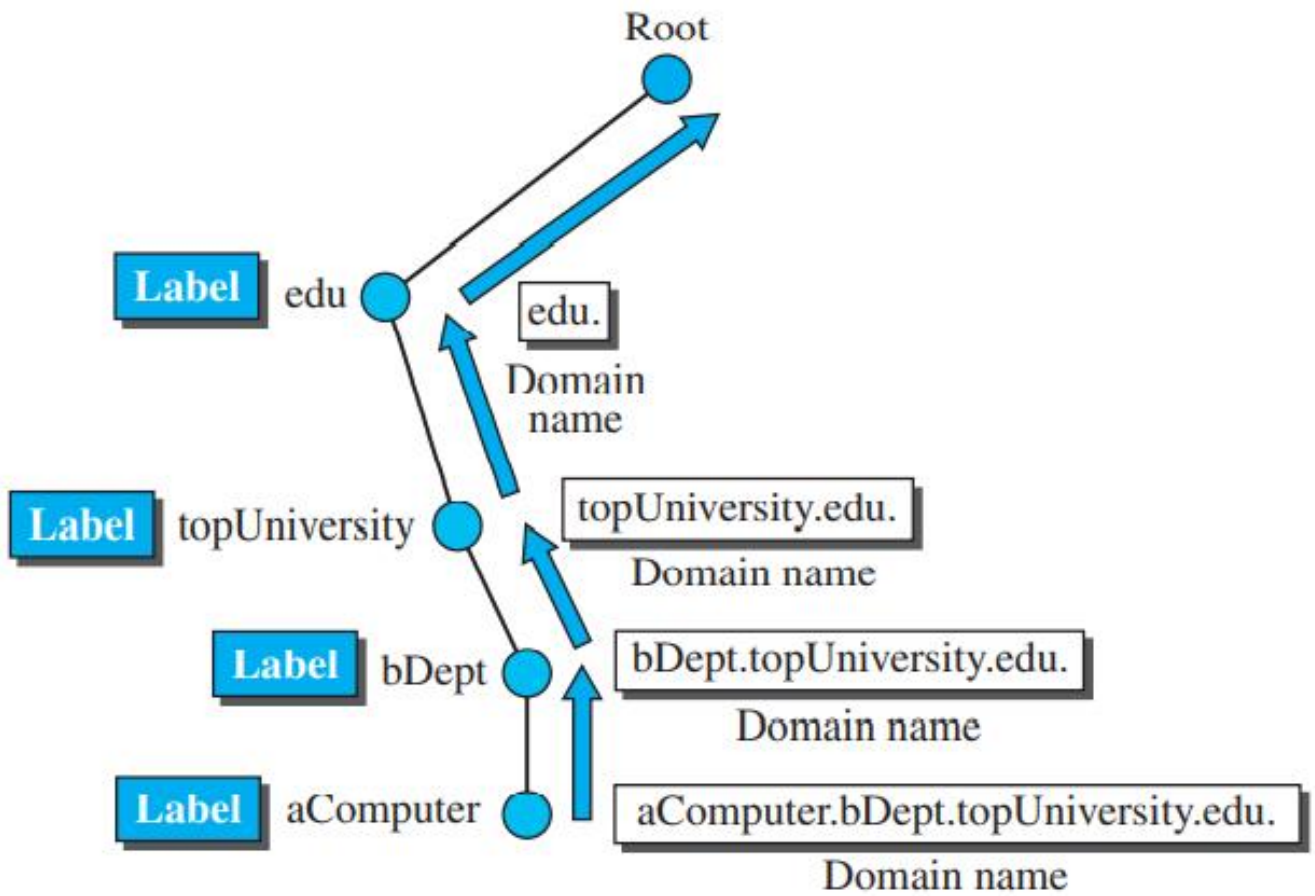
Figure 26.29 *Domain name space*



Domain Name Space

- **Label**
  - Each node in the tree has a label, which is a string with a maximum of 63 characters. The root label is a null string (empty string).
  - DNS requires that children of a node (**nodes that branch from the same node**) have different labels, which guarantees the uniqueness of the domain names.
- **Domain Name**
  - Each node in the tree has a domain name.
  - A full domain name is a sequence of labels separated by dots (.).
  - The domain names are always read from the node up to the root. The last label is the label of the root (null). This means that a full domain name always ends in a null label, which means the last character is a dot because the null string is nothing.
  - Figure 26.30 shows some domain names.
  - If a label is terminated by a null string, it is called a **fully qualified domain name (FQDN)**.
  - The name must end with a null label, but because null means nothing, the label ends with a dot.
  - If a label is not terminated by a null string, it is called a **partially qualified domain name (PQDN).** A PQDN starts from a node, but it does not reach the root. It is used when the name to be resolved belongs to the same site as the client. Here the resolver can supply the missing part, called the suffix, to create an FQDN.

Figure 26.30 *Domain names and labels*



Domain Name Space

- **Domain**
- A domain is a subtree of the domain name space. The name of the domain is the name of the node at the top of the subtree.
- Figure 26.31 shows some domains. Note that a domain may itself be divided into domains.

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

Figure 26.31 *Domains*

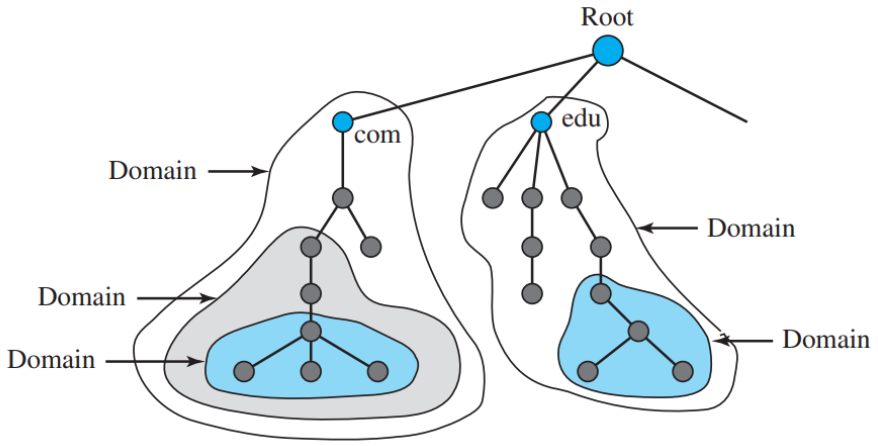
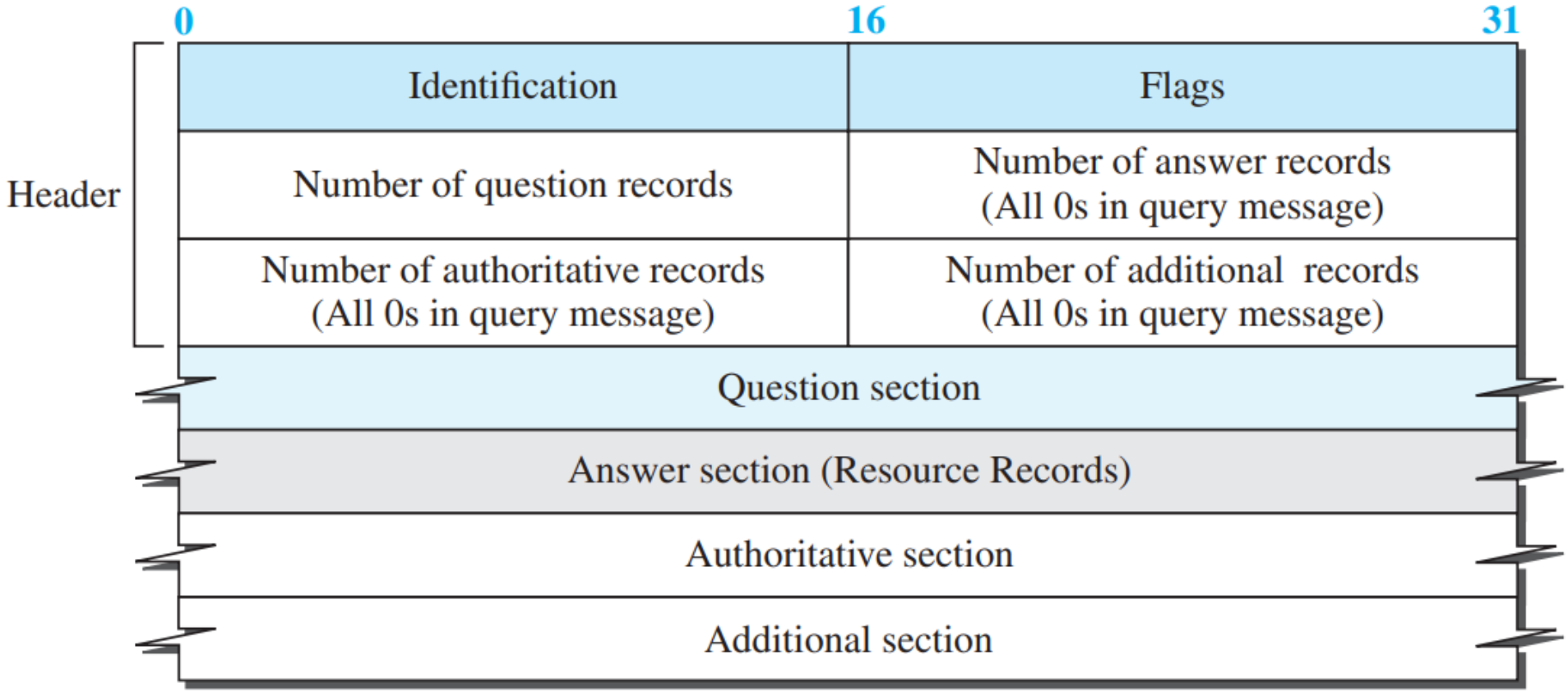


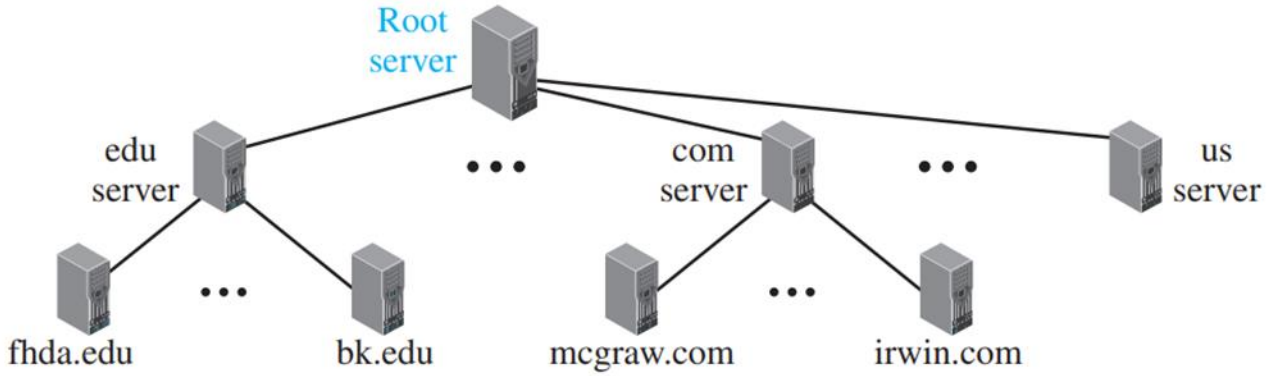
Figure 26.38 *DNS message*



Hierarchy of Name Servers

- The solution to these problems is to distribute the information among many computers called DNS servers.
- One way to do this is to **divide the whole space into many domains based on the first level.**
- we let the root stand alone and create as many domains (subtrees) as there are first-level nodes.
- DNS allows domains to be divided further into smaller domains (subdomains).
- Each 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

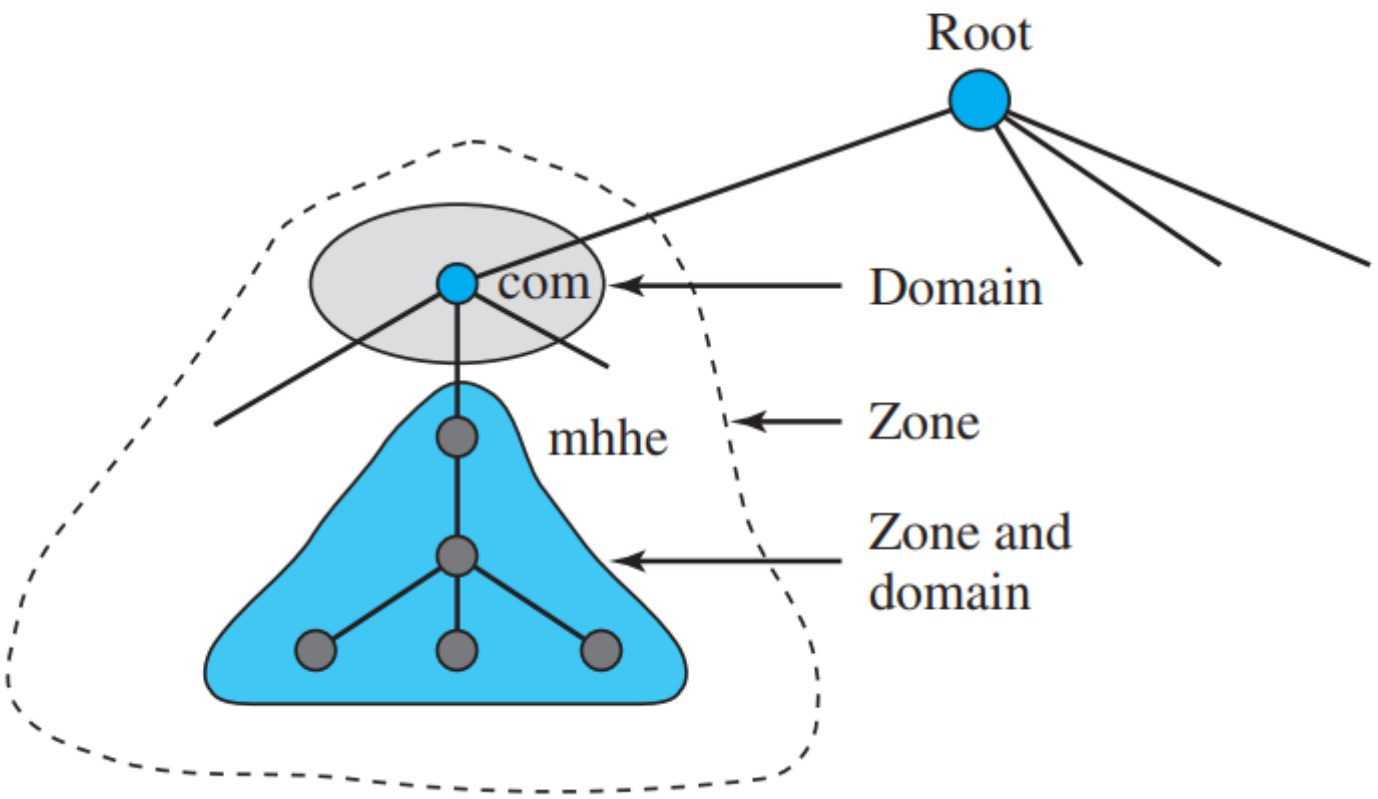
Figure 26.32 *Hierarchy of name servers*



Hierarchy of Name Servers

- **Zone**
- Since the complete domain name hierarchy cannot be stored on a single server, it is divided among many servers. What a server is responsible for or has authority over is **called a zone**. We can define a zone as a contiguous part of the entire tree.
- If a server accepts responsibility for a domain and does not divide the domain into smaller domains, **the "domain" and the "zone" refer to the same thing**. The server makes a database called a **zone file** and keeps all the information for every node under that domain.
- If a server divides its domain into subdomains and delegates part of its authority to other servers, **"domain" and "zone" refer to different things**. The information about the nodes in the subdomains is stored in the servers at the lower levels, with the original server keeping some sort of reference to these lower-level servers. The original server does not free itself from responsibility totally. It still has a zone, but the detailed information is kept by the lower-level servers.

Figure 26.33 *Zone*



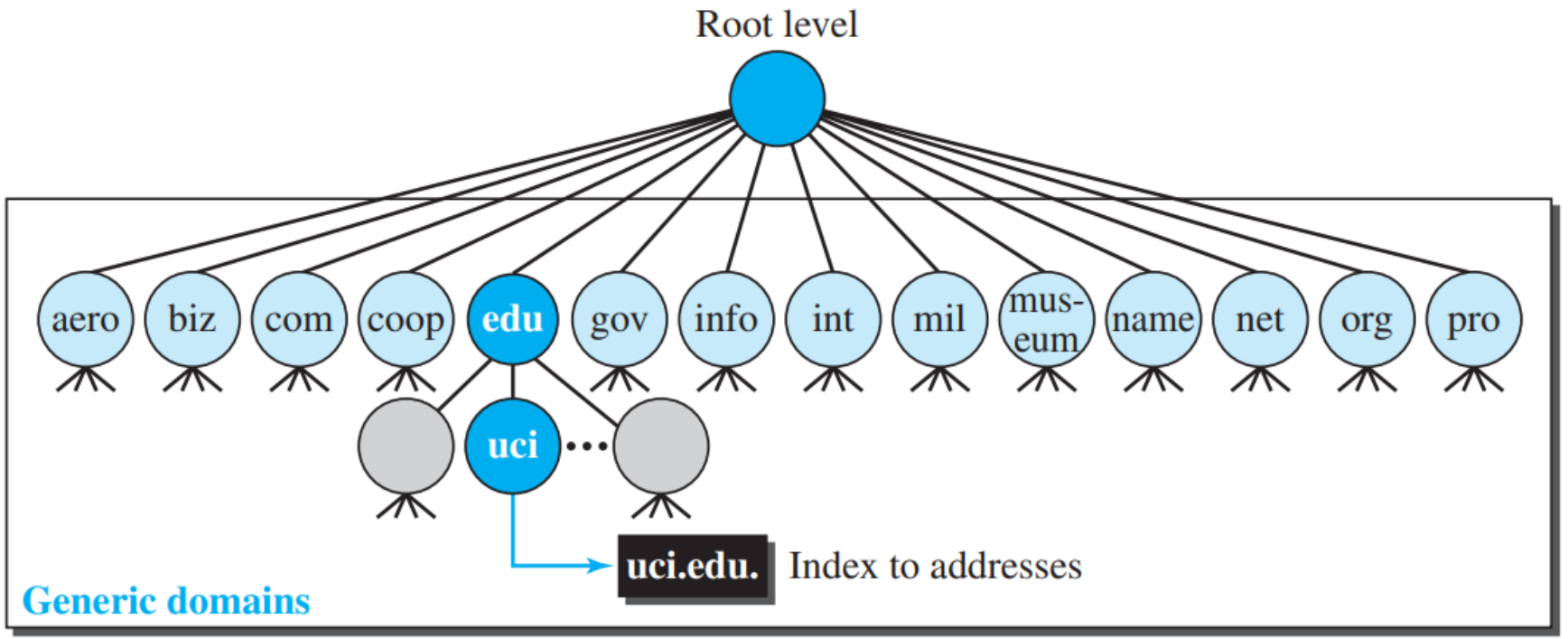
Name Servers

- 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.
  - The root servers are distributed all around the world.
- Primary and Secondary Servers
  - DNS defines two types of servers: **primary and secondary.**
  - A **primary server is** a server that stores a file about the zone for which it is an authority. It is responsible for creating, maintaining, and updating the zone file. It stores the zone file on a local disk.
  - A **secondary server** is a server that transfers the complete information about a zone from another server (primary or secondary) and stores the file on its local disk.
  - The **secondary server** neither creates nor updates the zone files. If updating is required, it must be done by the primary server, which sends the updated version to the secondary.
  - The **primary and secondary servers** are both authoritative for the zones they serve. The idea is not to put the secondary server at a lower level of authority but to create redundancy for the data so that if one server fails, the other can continue serving clients. Note also that a server can be a primary server for a specific zone and a secondary server for another zone. Therefore, when we refer to a server as a primary or secondary server, we should be careful about which zone we refer to.

Name Servers

- **Primary and Secondary Servers**
  - DNS defines two types of servers: primary and secondary. A primary server is a server that stores a file about the zone for which it is an authority. It is responsible for creating, maintaining, and updating the zone file. It stores the zone file on a local disk. A secondary server is a server that transfers the complete information about a zone from another server (primary or secondary) and stores the file on its local disk.
  - The secondary server neither creates nor updates the zone files. If updating is required, it must be done by the primary server, which sends the updated version to the secondary.
  - The primary and secondary servers are both authoritative for the zones they serve. The idea is not to put the secondary server at a lower level of authority but to create redundancy for the data so that if one server fails, the other can continue serving clients.
  - Note also that a server can be a primary server for a specific zone and a secondary server for another zone. Therefore, when we refer to a server as a primary or secondary server, we should be careful about which zone we refer to.
- **DNS in the Internet**
  - DNS is a protocol that can be used in different platforms. In the Internet, the domain name space (tree) was originally divided into three different sections: generic domains, country domains, and the inverse domains. However, due to the rapid growth of the Internet, it became extremely difficult to keep track of the inverse domains, which could be used to find the name of a host when given the IP address. The inverse domains are now deprecated (see RFC 3425). We, therefore, concentrate on the first two.
- **Generic Domains**
  - The generic domains define registered hosts according to their generic behavior. Each node in the tree defines a domain, which is an index to the domain name space database (see Figure 26.34).

Figure 26.34 *Generic domains*



- Looking at the tree, we see that the first level in the generic domains section allows 14 possible labels. These labels describe the organization types as listed in Table 26.12.

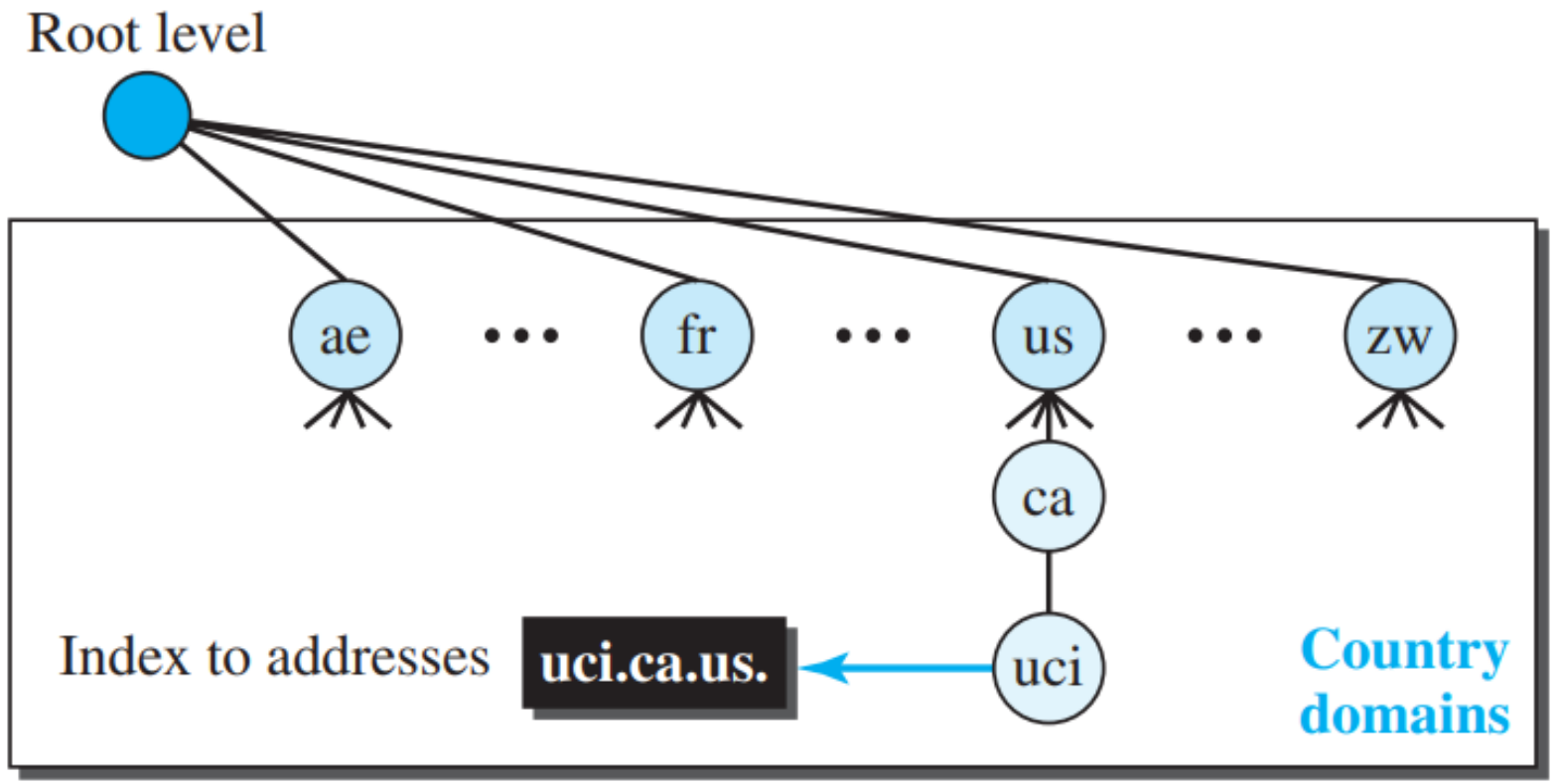
Table 26.12 *Generic domain labels*

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

Country Domains

- The country domains section uses two-character country abbreviations (e.g., us for United States). Second labels can be organizational, or they can be more specific national designations. The United States, for example, uses state abbreviations as a subdivision of us (e.g., ca.us.).
- Figure 26.35 shows the country domains section. The address uci.ca.us. can be translated to University of California, Irvine, in the state of California in the United States.

Figure 26.35 *Country domains*



Resolution

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

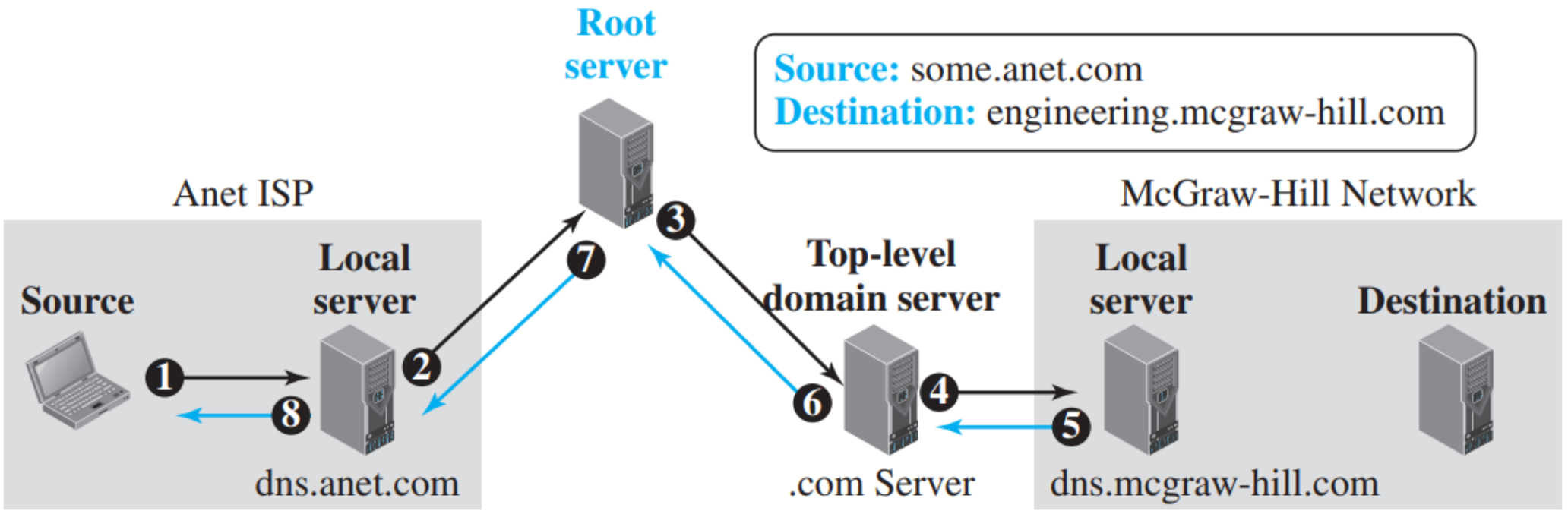
Resolution

- Figure 26.36 shows a simple example of a recursive resolution. We assume that an application program running on a host named some.anet.com needs to find the IP address of another host named engineering.mcgraw-hill.com to send a message to.
- The source host is connected to the Anet ISP; the destination host is connected to the McGraw-Hill network.

Recursive Resolution

- The application program on the source host calls the DNS resolver (client) to find the IP address of the destination host. The resolver, which does not know this address, sends the query to the local DNS server (for example, dns.anet.com) running at the Anet ISP site (event 1). We assume that this server does not know the IP address of the destination host either. It sends the query to a root DNS server, whose IP address is supposed to be known to this local DNS server (event 2).
- Root servers do not normally keep the mapping between names and IP addresses, but a root server should at least know about one server at each top level domain (in this case, a server responsible for com domain). The query is sent to this top-level-domain server (event 3).
- We assume that this server does not know the name-address mapping of this specific destination, but it knows the IP address of the local DNS server in the McGraw-Hill company (for example, dns.mcgraw-hill.com). The query is sent to this server (event 4), which knows the IP address of the destination host. The IP address is now sent back to the top-level DNS server (event 5), then back to the root server (event 6), then back to the ISP DNS server, which may cache it for the future queries (event 7), and finally back to the source host (event 8).

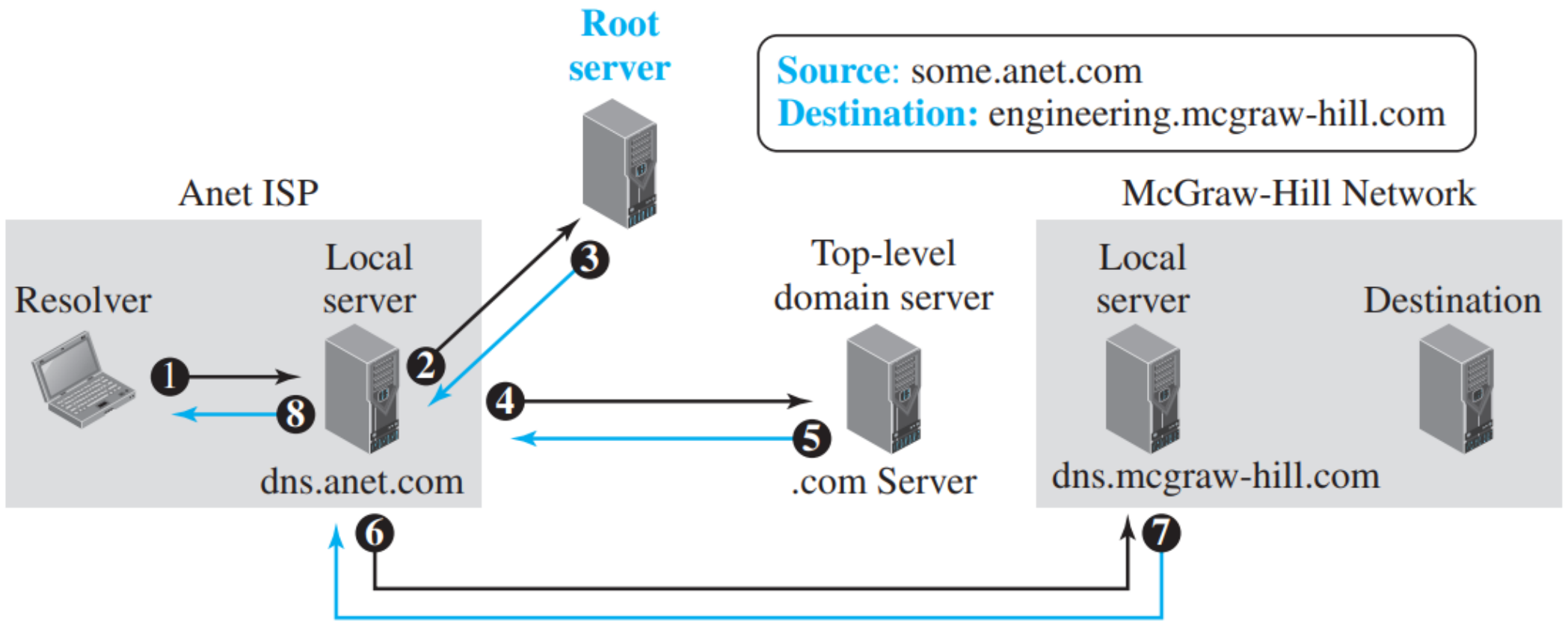
Figure 26.36 *Recursive resolution*



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.
- Normally the iterative resolution takes place between two local servers; the original resolver gets the final answer from the local server.
- Note that the messages shown by events 2, 4, and 6 contain the same query.
- However, the message shown by event 3 contains the IP address of the top-level domain server, the message shown by event 5 contains the IP address of the McGraw-Hill local DNS server, and the message shown by event 7 contains the IP address of the destination. When the Anet local DNS server receives the IP address of the destination, it sends it to the resolver (event 8).

Figure 26.37 *Iterative resolution*



Caching

- Each time a server receives a query for a name that is not in its domain, it needs to search its database for a server IP address. Reduction of this search time would increase a mapping from another server and receives the response, it stores this information in its cache memory before sending it to the client. If the same or another client asks for the same mapping, it can check its cache memory and resolve the problem. However, to inform the client that the response is coming from the cache memory and not from an authoritative source, the server marks the response as unauthoritative.

- Caching speeds up resolution, but it can also be problematic. If a server caches a mapping for a long time, it may send an outdated mapping to the client. To counter this, two techniques are used. First, the authoritative server always adds information to the mapping called time to live (TTL). It defines the time in seconds that the receiving server can cache the information. After that time, the mapping is invalid and any query must be sent again to the authoritative server. Second, DNS requires that each server keep a TTL counter for each mapping it caches. The cache memory must be searched periodically and those mappings with an expired TTL must be purged

Resource Records

- The zone information associated with a server is implemented as a set of resource records. In other words, a name server stores a database of resource records.
- A **resource record** is a 5-tuple structure, as shown below:
- The domain name field is what identifies the resource record.
- The value defines the information kept about the domain name.
- The TTL defines the number of seconds for which the information is valid. The class defines the type of network; we are only interested in the class IN (Internet). The type defines how the value should be interpreted.
- Table 26.13 lists the common types and how the value is interpreted for each type.

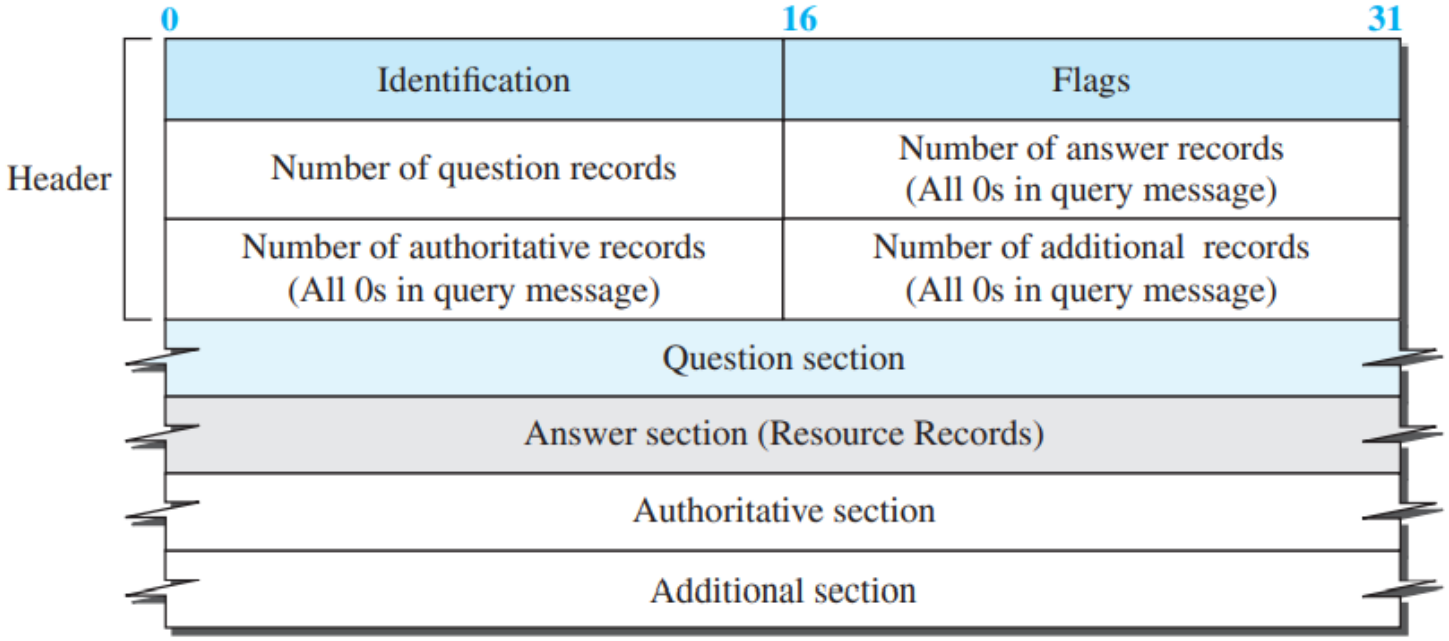
Table 26.13 *Types*

Type	*Interpretation of value*
A	A 32-bit IPv4 address (see Chapter 18)
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 22)

DNS Messages

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

Figure 26.38 *DNS message*



DNS Messages

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

DNS

- Example 26.13 In UNIX and Windows, the nslookup utility can be used to retrieve address/name mapping. The following shows how we can retrieve an address when the domain name is given.

```
$nslookup www.forouzan.biz
Name:  www.forouzan.biz
Address: 198.170.240.179
```

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

DNS

- **Registrars**
- New domains are added to DNS through a registrar, a commercial entity accredited by ICANN.
- A registrar first verifies that the requested domain name is unique and then enters it into the DNS database and a fee is charged.
- To register, the organization needs to give the name of its server and the IP address of the server. For example, a new commercial organization named wonderful with a server named ws and IP address 200.200.200.5 needs to give the following information to one of the registrars
- **DDNS**
- When the DNS was designed, no one predicted that there would be so many address changes. In DNS, when there is a change, such as adding a new host, removing a host, or changing an IP address, the change must be made to the DNS master file.
- These involve a lot of manual updating. The size of today's Internet does not allow for this kind of manual operation.
- The DNS master file must be updated dynamically. The Dynamic Domain Name System (DDNS) therefore was devised to respond to this need.
- In DDNS, when a binding between a name and an address is determined, the information is sent, usually by DHCP to a primary DNS server.
- The primary server updates the zone.
- The secondary servers are notified either actively or passively. In active notification, the primary server sends a message to the secondary servers about the change in the zone, whereas in passive notification, the secondary servers periodically check for any changes.
- In either case, after being notified about the change, the secondary server requests information about the entire zone (called the zone transfer). To provide security and prevent unauthorized changes in the DNS records, DDNS can use an authentication mechanism.

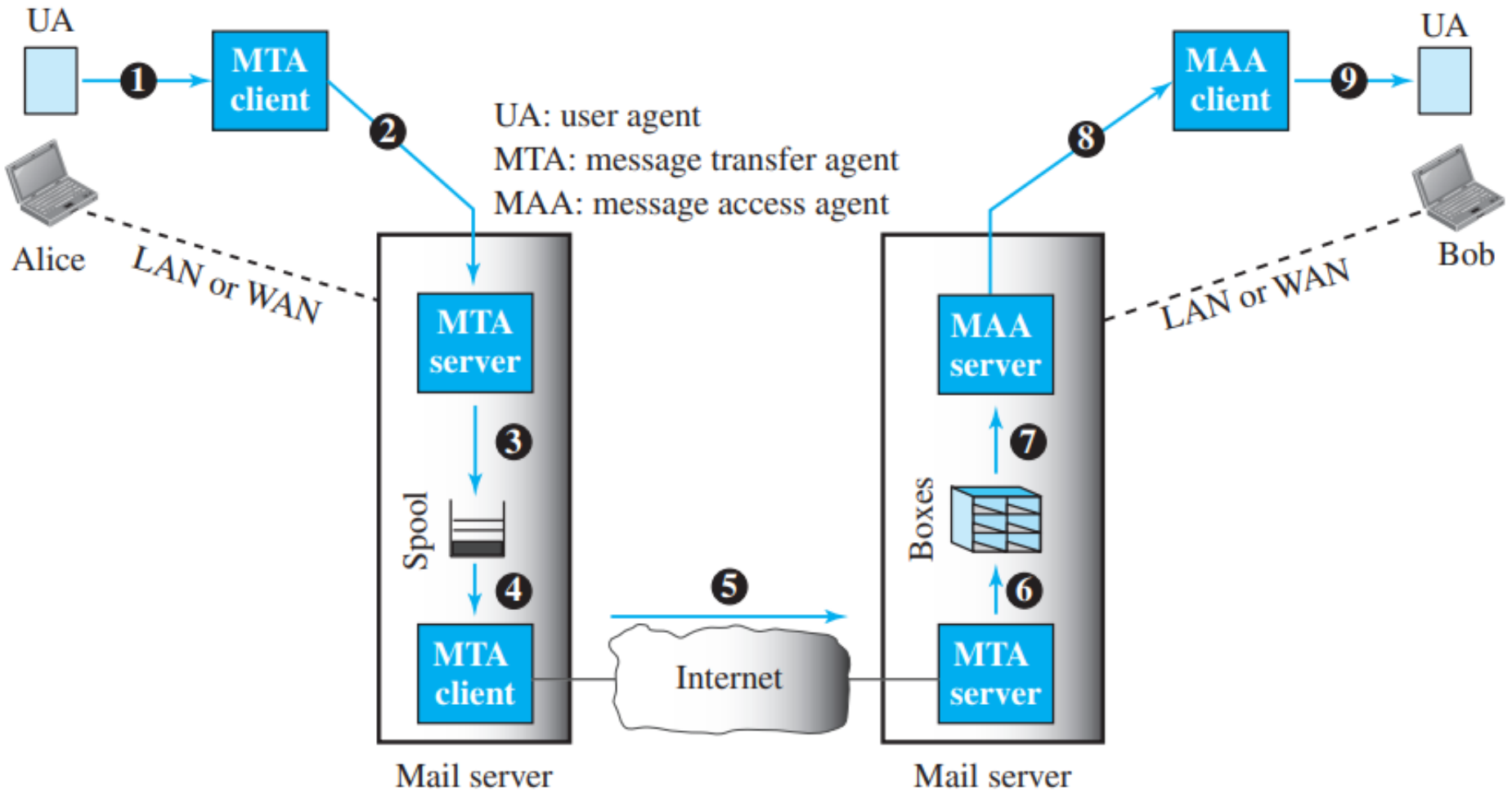
Security of DNS

- DNS is one of the most important systems in the Internet infrastructure; it provides crucial services to Internet users. Applications such as Web access or e-mail are heavily dependent on the proper operation of DNS. DNS can be attacked in several ways including:
- 1. The attacker may read the response of a DNS server to find the nature or names of sites the user mostly accesses. This type of information can be used to find the user's profile. To prevent this attack, DNS messages need to be confidential
- 2. 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 site or domain the attacker wishes the user to access. This type of attack can be prevented using message origin authentication and message integrity.
- 3. The attacker may flood the DNS server to overwhelm it or eventually crash it. This type of attack can be prevented using the provision against denial-of-service attack.
- To protect DNS, IETF has devised a technology named DNS Security (DNSSEC) that provides message origin authentication and message integrity using a security service called digital signature.
- DNSSEC, however, does not provide confidentiality for the DNS messages.
- There is no specific protection against the denial-of-service attack in the specification of DNSSEC. However, the caching system protects the upper-level servers against this attack to some extent.

EMAIL

- e-mail is considered a one-way transaction
- To communicate Alice and Bob use three different agents: a user agent (UA), a message transfer agent (MTA), and a message access agent (MAA).
- There are two important points we need to emphasize here.
  - First, Bob cannot bypass the mail server and use the MTA server directly. To use the MTA server directly, Bob would need to run the MTA server all the time because he does not know when a message will arrive. This implies that Bob must keep his computer on all the time if he is connected to his system through a LAN. If he is connected through a WAN, he must keep the connection up all the time. Neither of these situations is feasible today.
  - Second, note that Bob needs another pair of client-server programs: message access programs. This is because an MTA client-server program is a push program: the client pushes the message to the server. Bob needs a pull program. The client needs to pull the message from the server. We discuss more about MAAs shortly.

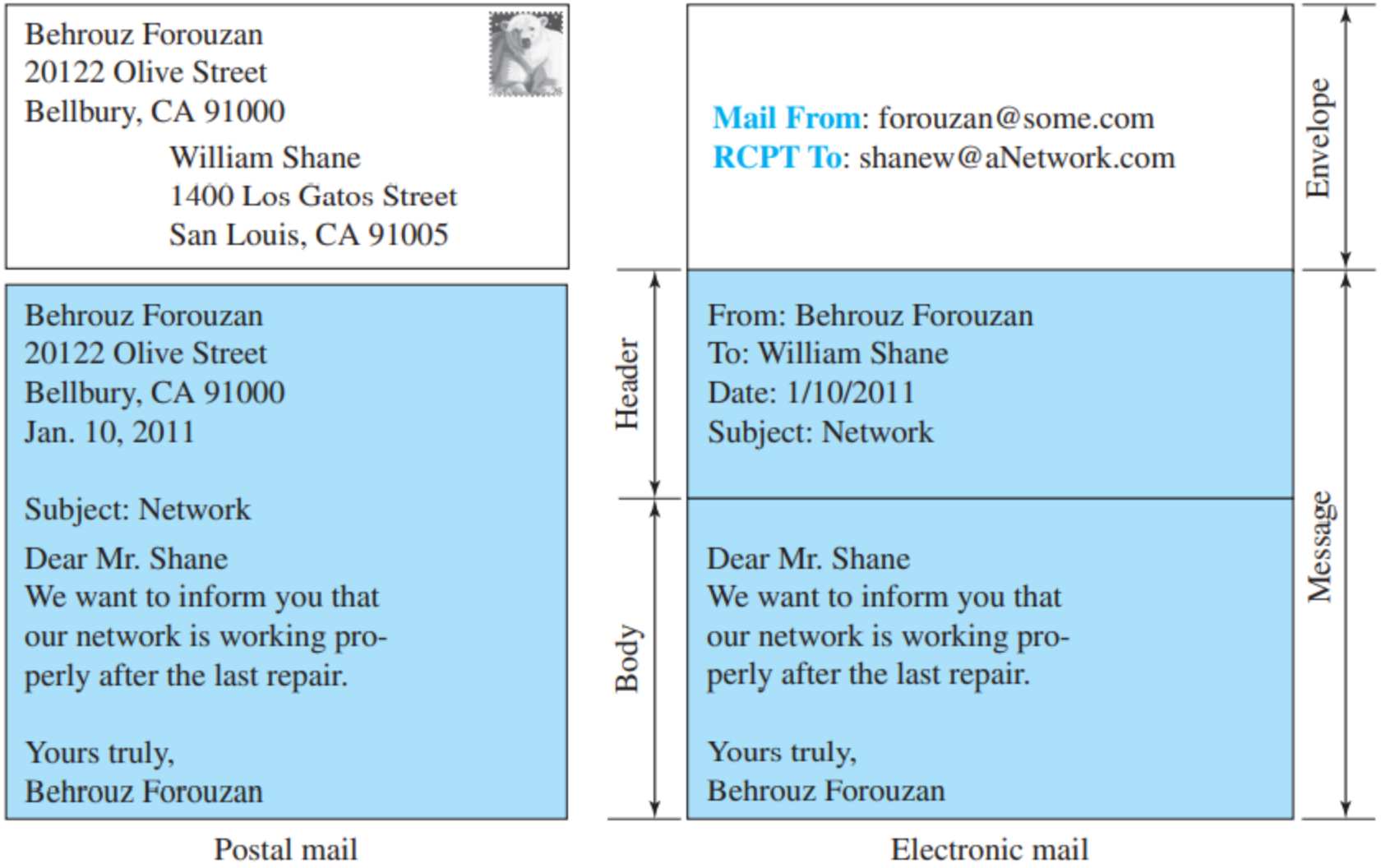
Figure 26.12 *Common scenario*



User Agent

- The first component of an electronic mail system is the user agent (UA).
- A user agent is a software package (program) that composes, reads, replies to, and forwards messages. It also handles local mailboxes on the user computers. There are two types of user agents:
  - Command-driven.
    - Command driven user agents belong to the early days of electronic mail. They are still present as the underlying user agents.
    - a user can type the character r, at the command prompt, to reply to the sender of the message, or type the character R to reply to the sender and all recipients. Some examples of command driven user agents are mail, pine, and elm.
  - GUI-based
    - interact with the software by using both the keyboard and the mouse. They have graphical components such as icons, menu bars, and windows that make the services easy to access. Some examples of GUI-based user agents are Eudora and Outlook

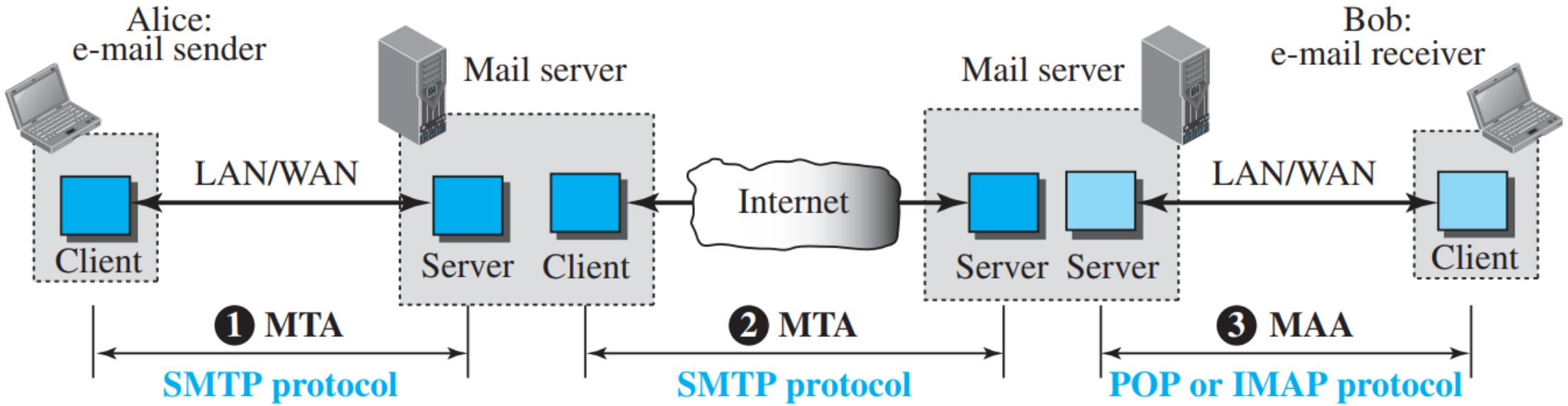
Figure 26.13 *Format of an e-mail*



EMAIL

- Explanation of
  - Sending Mail
  - Receiving Mail
  - Addresses
  - Mailing List or Group List
  - Message Transfer Agent: SMTP
    - e-mail is one of those applications that needs three uses of client-server paradigms to accomplish its task.
    - shows these three client-server applications. We refer to the first and the second as Message Transfer Agents (MTAs), the third as Message Access Agent (MAA).

Figure 26.15 *Protocols used in electronic mail*



Protocols in email

- The formal protocol that defines the MTA client and server in the Internet is called Simple Mail Transfer Protocol (SMTP).
- SMTP is used two times, between the sender and the sender's mail server and between the two mail servers.
- Another protocol is needed between the mail server and the receiver.
- SMTP simply defines how commands and responses must be sent back and forth.
- Commands and Responses
  - SMTP uses commands and responses to transfer messages between an MTA client and an MTA server.
  - The command is from an MTA client to an MTA server; the response is from an MTA server to the MTA client.
  - Each command or reply is terminated by a two character (carriage return and line feed) end-of-line token.
  - Commands : Commands are sent from the client to the server.

Table 26.6 *SMTP commands*

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

- Responses
  - Responses are sent from the server to the client. A response is a threedigit code that may be followed by additional textual information. Table shows the most common response types.

Table 26.7 *Responses*

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

Table 26.7 *Responses (continued)*

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

Mail Transfer Phases

- The process of transferring a mail message occurs in three phases: connection establishment, mail transfer, and connection termination.
- Connection Establishment
  - After a client has made a TCP connection to the wellknown port 25, the SMTP server starts the connection phase. This phase involves the following three steps:
- 1. The server sends code 220 (service ready) to tell the client that it is ready to receive
- mail. If the server is not ready, it sends code 421 (service not available).
- 2. The client sends the HELO message to identify itself, using its domain name
- address. This step is necessary to inform the server of the domain name of the client.
- 3. The server responds with code 250 (request command completed) or some other
- code depending on the situation.

Mail Transfer Phases

- Message Transfer
  - After connection has been established between the SMTP client and server, a single message between a sender and one or more recipients can be exchanged. This phase involves eight steps.
  - Steps 3 and 4 are repeated if there is more than one recipient.

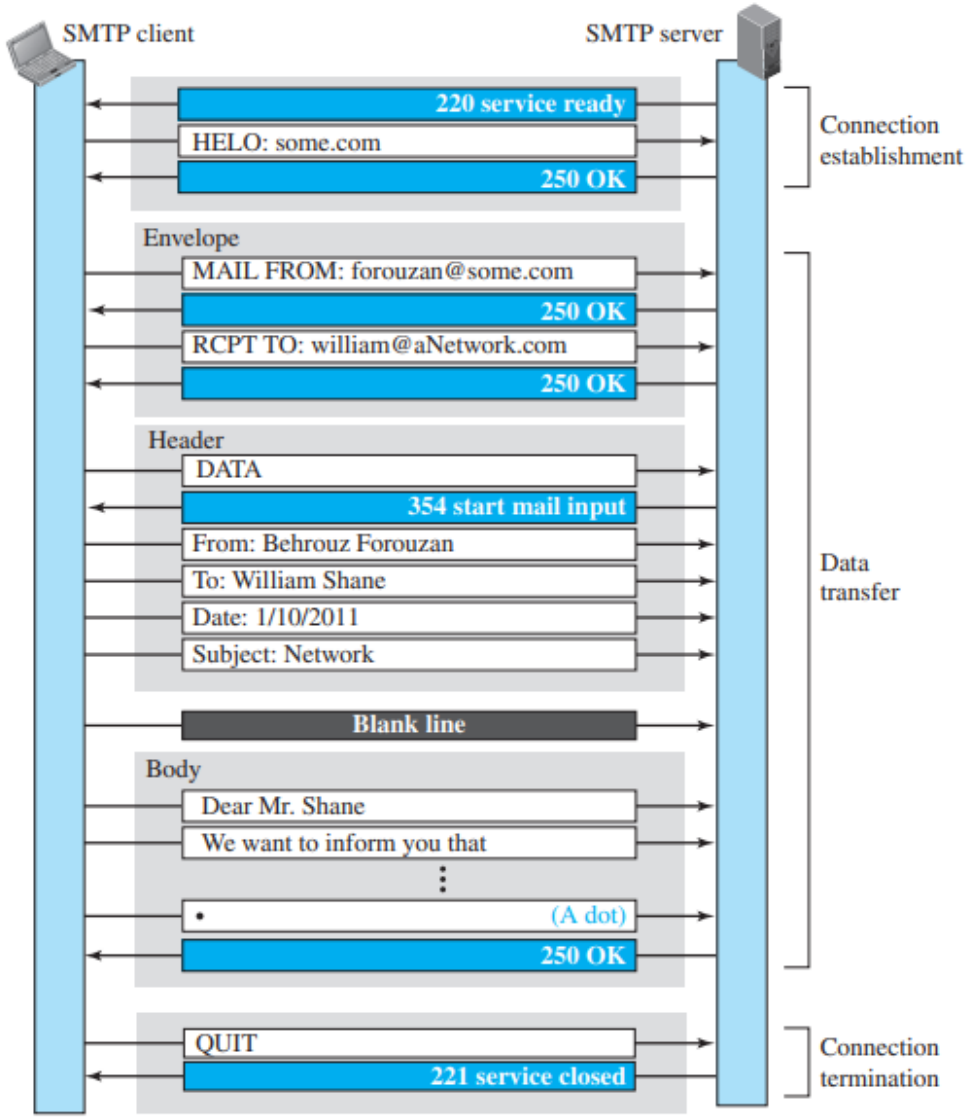
  1. The client sends the MAIL FROM message to introduce the sender of the message. It includes the mail address of the sender (mailbox and the domain name). This step is needed to give the server the return mail address for returning errors and reporting messages.
  2. The server responds with code 250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

Connection Termination

- After the message is transferred successfully, the client terminates the connection. This phase involves two steps.
  1. The client sends the QUIT command.
  2. The server responds with code 221 or some other appropriate code

Example

Figure 26.16 *Example 26.12*



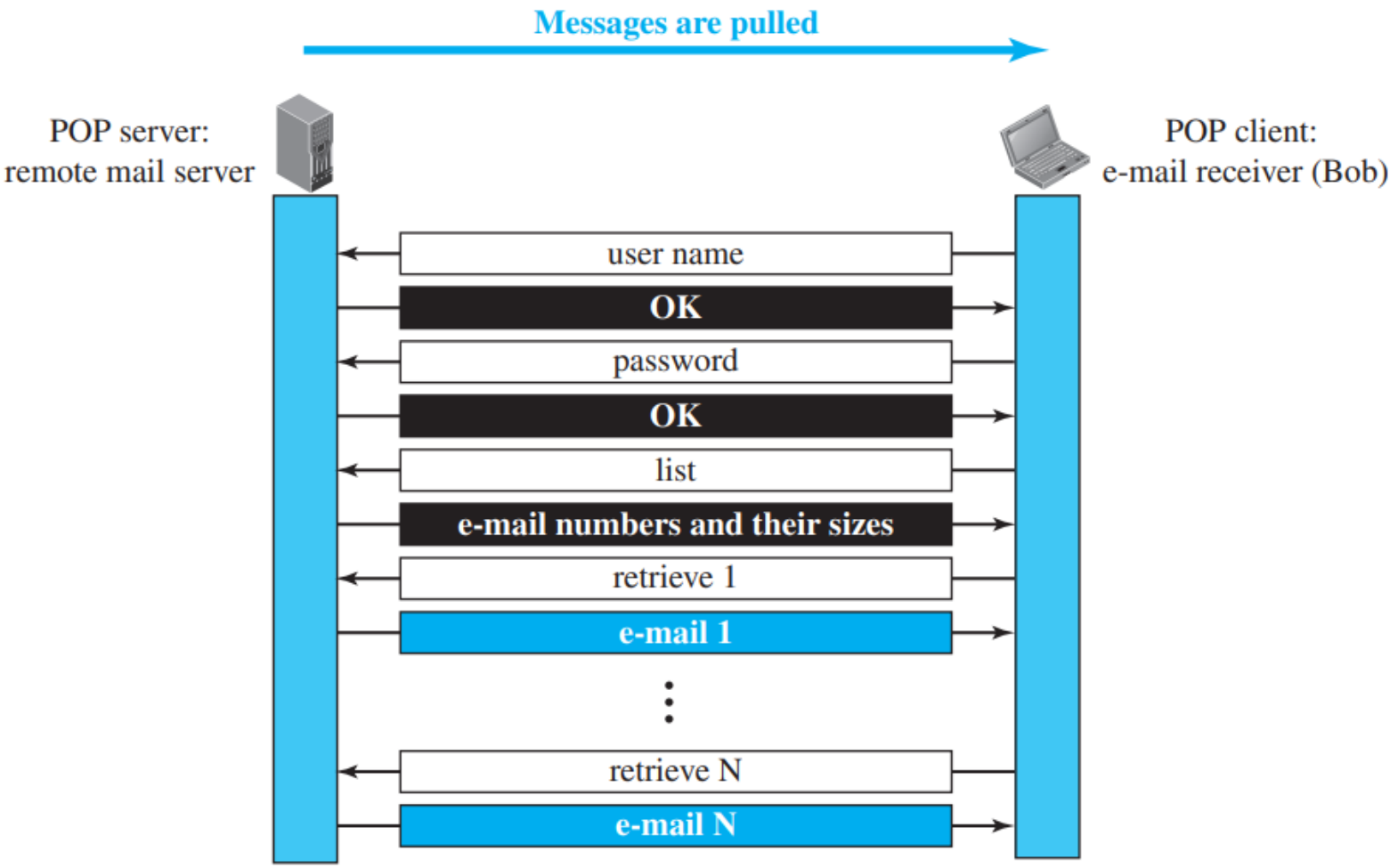
Message Access Agent: POP and IMAP

- the third stage needs a pull protocol; the client must pull messages from the server.
- The direction of the bulk data is from the server to the client.
- The third stage uses a message access agent.
- Currently two message access protocols are available: Post Office Protocol, version 3 (POP3) and Internet Mail Access Protocol, version 4 (IMAP4).

POP3

- Post Office Protocol, version 3 (POP3) is simple but limited in functionality. The client POP3 software is installed on the recipient computer; the server POP3 software is installed on the mail server.
- Mail access starts with the client when the user needs to download its e-mail from the mailbox on the mail server. The client opens a connection to the server on TCP port 110.
- It then sends its user name and password to access the mailbox. The user can then list and retrieve the mail messages, one by one.

Figure 26.17 *POP3*



POP3

- POP3 has two modes: the delete mode and the keep mode.
  - In the delete mode, the mail is deleted from the mailbox after each retrieval.
    - is normally used when the user is working at her permanent computer and can save and organize the received mail after reading or replying.
  - In the keep mode, the mail remains in the mailbox after retrieval.
    - is normally used when the user accesses her mail away from her primary computer (for example, from a laptop). The mail is read but kept in the system for later retrieval and organizing

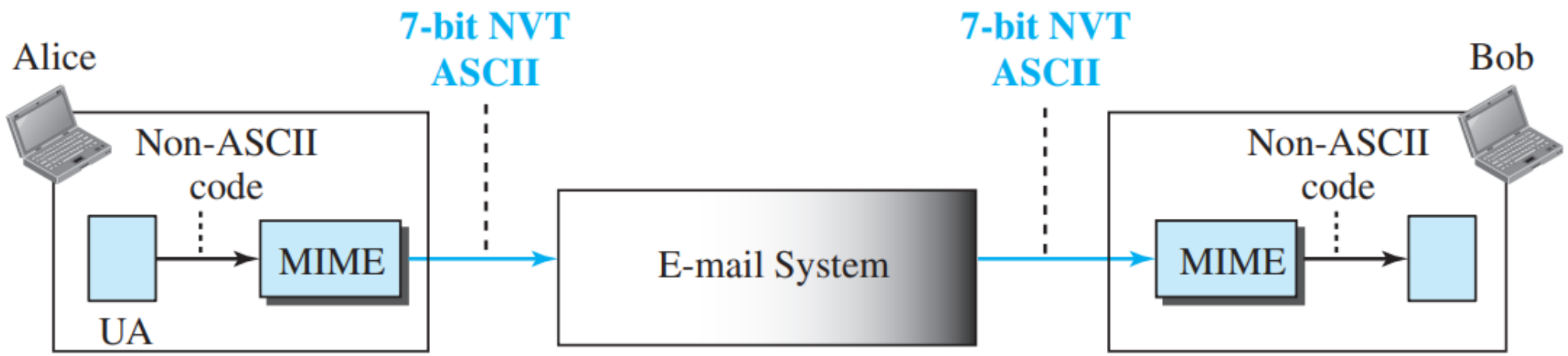
IMAP4

- Another mail access protocol with more features is Internet Mail Access Protocol, version 4 (IMAP4).
- POP3 is deficient in several ways.
  - It does not allow the user to organize her mail on the server;
  - the user cannot have different folders on the server.
  - does not allow the user to partially check the contents of the mail before downloading.
- IMAP4 provides the following extra functions:
  - A user can check the e-mail header prior to downloading.
  - A user can search the contents of the e-mail for a specific string of characters prior to downloading.
  - A user can partially download e-mail. This is especially useful if bandwidth is limited and the e-mail contains multimedia with high bandwidth requirements.
  - A user can create, delete, or rename mailboxes on the mail server.
  - A user can create a hierarchy of mailboxes in a folder for e-mail storage.

MIME- Multipurpose Internet Mail Extensions

- Electronic mail has a simple structure. Its simplicity, however, comes with a price. It can send messages only in NVT 7-bit ASCII format.
- Limitations.
  - It cannot be used for languages other than English (such as French, German, Hebrew, Russian, Chinese, and Japanese).
  - cannot be used to send binary files or video or audio data.
- MIME is a supplementary protocol that allows non-ASCII data to be sent through e-mail.
- MIME transforms non-ASCII data at the sender site to NVT ASCII data and delivers it to the client MTA to be sent through the Internet.
- The message at the receiving site is transformed back to the original data.
- We can think of MIME as a set of software functions that transforms non-ASCII data to ASCII data and vice versa

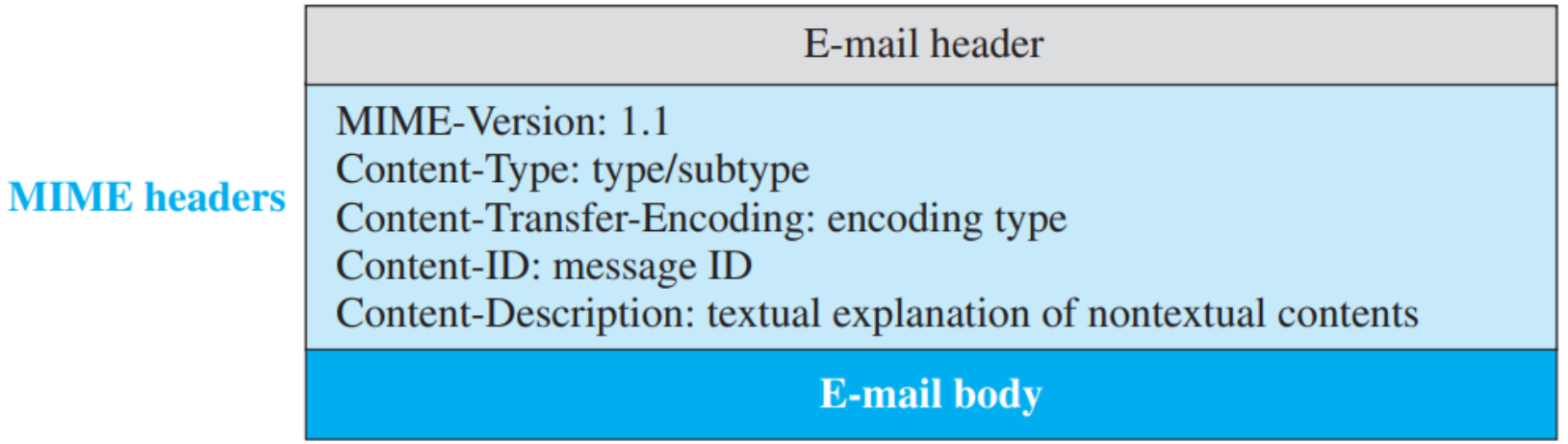
Figure 26.18 *MIME*



MIME Headers

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

Figure 26.19 *MIME header*



MIME-Version

- This header defines the version of MIME used. The current version is 1.1.
- Content-Type
  - This header defines the type of data used in the body of the message.
  - The content type and the content subtype are separated by a slash.
  - Depending on the subtype, the header may contain other parameters.
  - MIME allows seven different types of data

Table 26.8 *Data types and subtypes in MIME*

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

Content-Transfer-Encoding

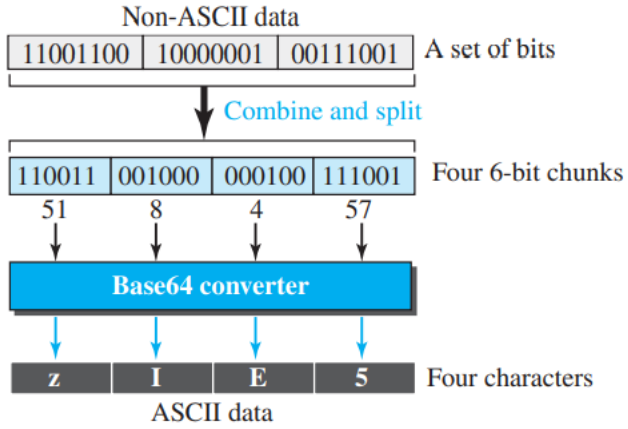
- This header defines the method used to encode the messages into 0s and 1s for transport

Table 26.9 *Methods for Content-Transfer-Encoding*

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

The last two encoding methods are interesting. In the Base64 encoding, data, as a string of bits, is first divided into 6-bit chunks as shown in Figure 26.20.

Figure 26.20 *Base64 conversion*



Content-Transfer-Encoding

Each 6-bit section is then converted into an ASCII character according to Table 26.10.

Table 26.10 *Base64 converting table*

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

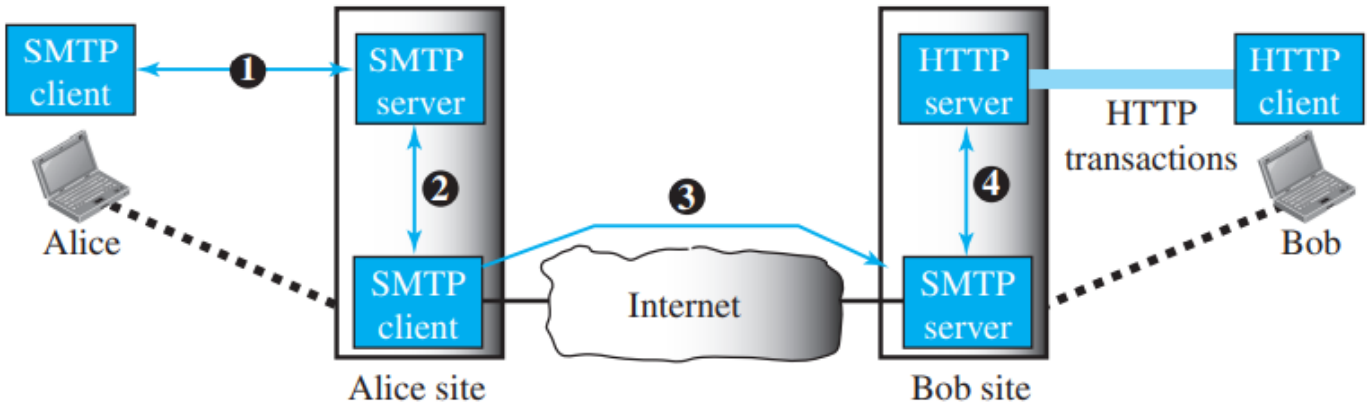
- Base64 is a redundant encoding scheme
  - every six bits become one ASCII character and are sent as eight bits.
  - overhead of 25 percent. If the data consist mostly of ASCII characters with a small non-ASCII portion,
  - we can use quoted-printable encoding.
  - In quoted-printable, if a character is ASCII, it is sent as is.
  - If a character is not ASCII, it is sent as three characters.
  - The first character is the equal sign (=). The next two characters are the hexadecimal representations of the byte.
  - In the example, the third character is a non-ASCII because it starts with bit 1.
  - It is interpreted as two hexadecimal digits (9D16), which is replaced by three ASCII characters (=, 9, and D).
- Content-ID
  - This header uniquely identifies the whole message in a multiple message environment.
- Content-Description
  - This header defines whether the body is image, audio, or video

Web-Based Mail

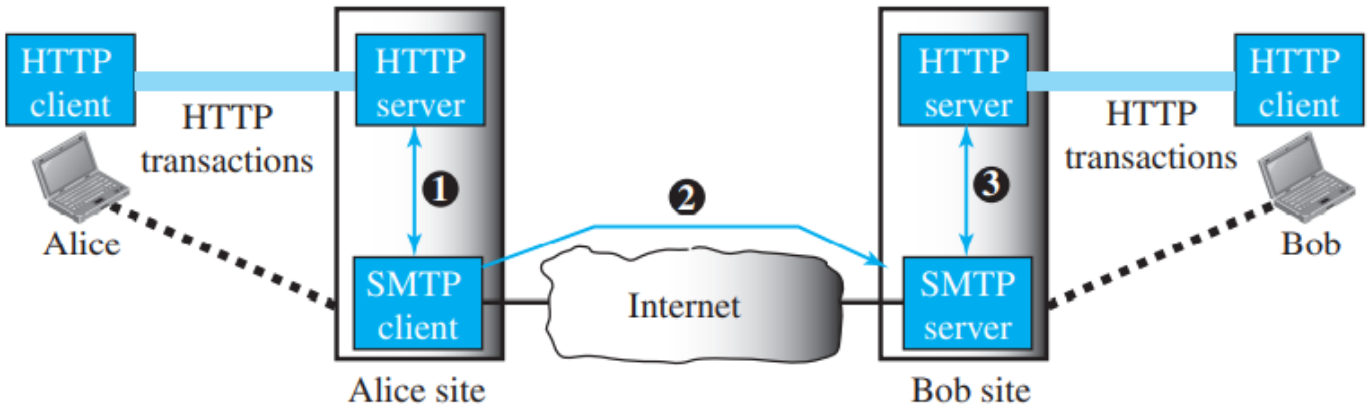
- E-mail is such a common application that some websites today provide this service to anyone who accesses the site.
- Three common sites are Hotmail, Yahoo, and Google mail. The idea is very simple.
- Case I
- In the first case, Alice, the sender, uses a traditional mail server; Bob, the receiver, has an account on a web-based server.
  - Alice's browser -> mail server is done through SMTP.
  - sending mail server -> receiving mail server is still through SMTP.
  - message from the receiving server (the web server) -> Bob's browser is done through HTTP instead of using POP3 or IMAP4
  - Bob needs to retrieve his e-mails, he sends a request HTTP message to the website (Hotmail, for example). The website sends a form to be filled in by Bob, which includes the log-in name and the password. If the log-in name and password match, the list of e-mails is transferred from the web server to Bob's browser in HTML format. Now Bob can browse through his received e-mails and then, using more HTTP transactions, can get his e-mails one by one.

- Case II
  - both Alice and Bob use web servers, but not necessarily the same server.
  - Alice sends the message to the web server using HTTP transactions. Alice sends an HTTP request message to her web server using the name and address of Bob's mailbox as the URL. The server at the Alice site passes the message to the SMTP client and sends it to the server at the Bob site using SMTP protocol. Bob receives the message using HTTP transactions. However, the message from the server at the Alice site to the server at the Bob site still takes place using SMTP protocol.

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



Case 1: Only receiver uses HTTP



Case 2: Both sender and receiver use HTTP

E-Mail Security

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