# UNIT - V

## APPLICATION LAYER

# APPLICATION LAYER

- Client Server Paradigm
- Peer to Peer Paradigm
- Communication using TCP and UDP services
- Domain Name System (DNS)
- Hyper Text Transfer Protocol (HTTP)
- Email: SMTP, MIME, POP3, Webmail
- FTP, TELNET
- Dynamic Host Control Protocol (DHCP)
- Simple Network Management Protocol (SNMP)

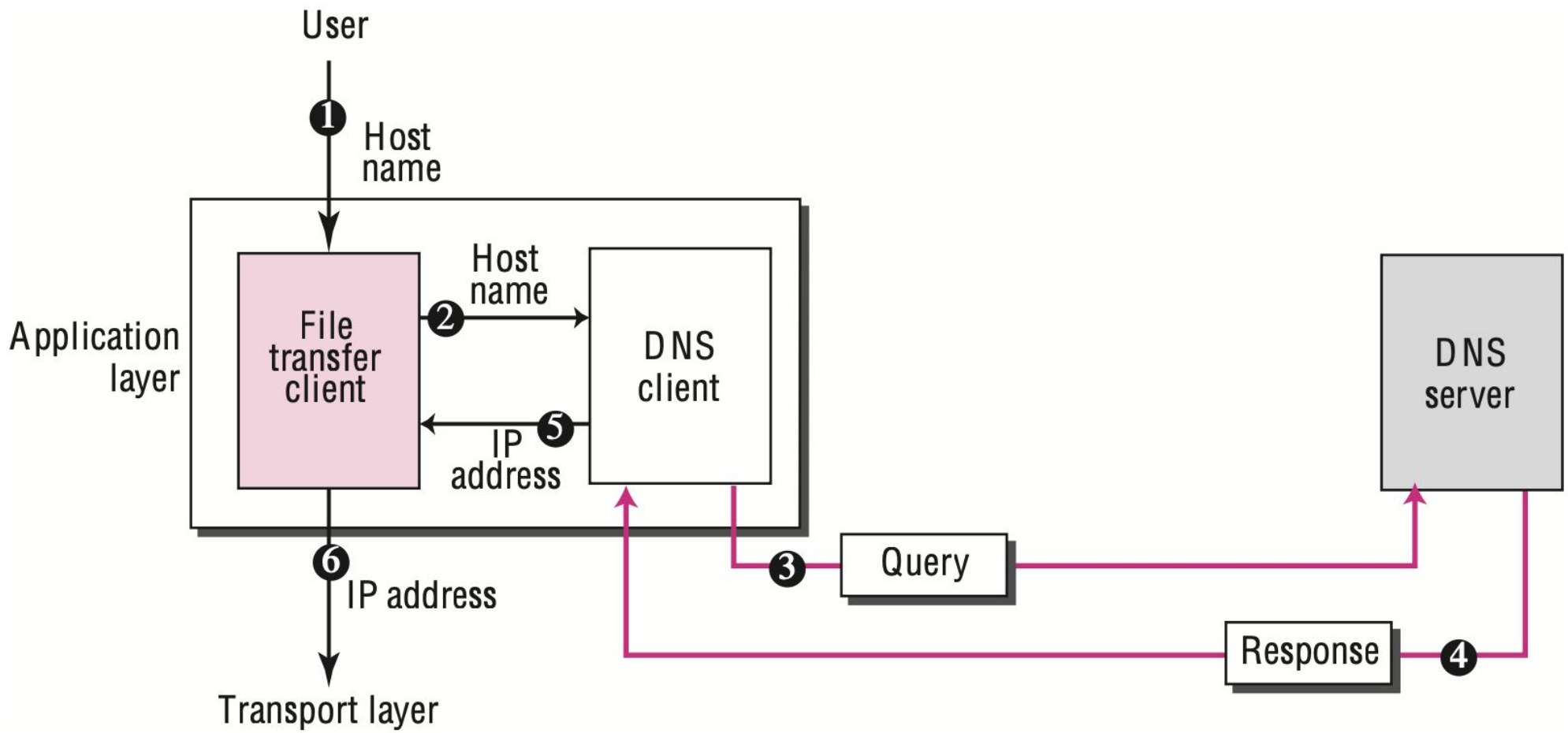
# DNS: DOMAIN NAME SYSTEM

- NEED FOR DNS
- To identify an entity, TCP/IP protocols use the IP address, which uniquely identifies the connection of a host to the Internet.
- However, people prefer to use names instead of numeric addresses.
- Therefore, we need a system that can map a name to an address or an address to a name.

# What is a DNS Server?

- A DNS server is a computer with a database containing the public IP addresses associated with the names of the websites an IP address brings a user to.
- DNS acts like a phonebook for the internet.
- Whenever people type domain names, like Facebook.com or Yahoo.com, into the address bar of web browsers, the DNS finds the right IP address.
- The site's IP address is what directs the device to go to the correct place to access the site's data.

## Figure 19.1 *Purpose of DNS*

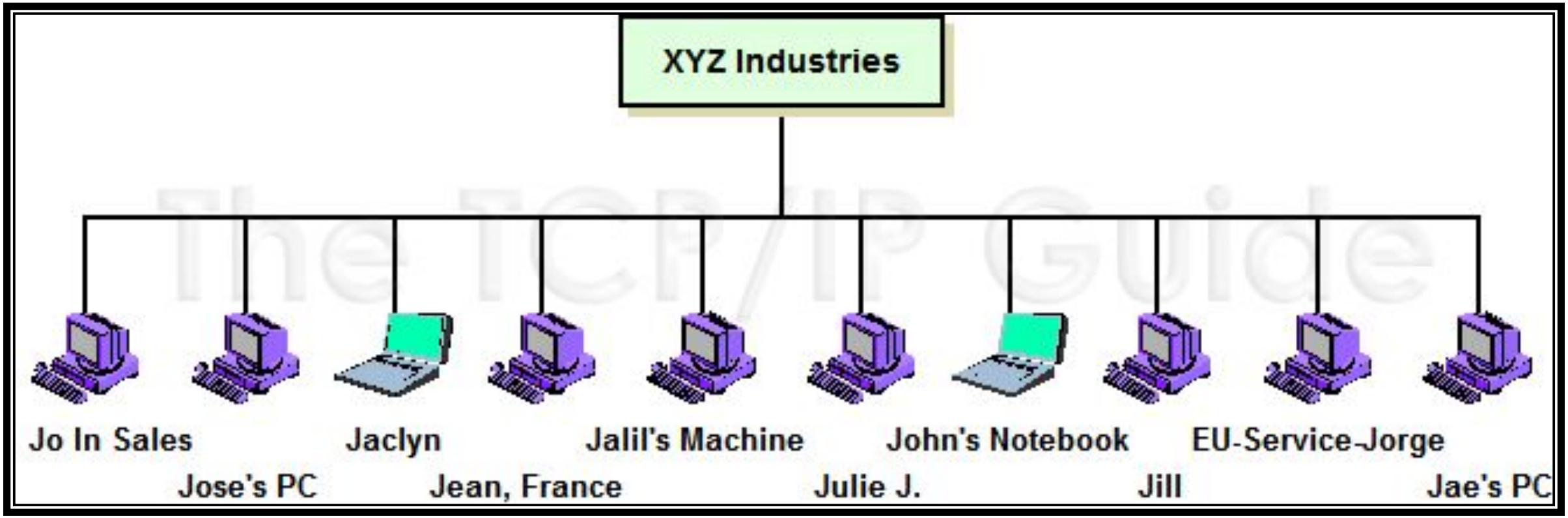


# NAME SPACE

- 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 name space
  - Hierarchical name space

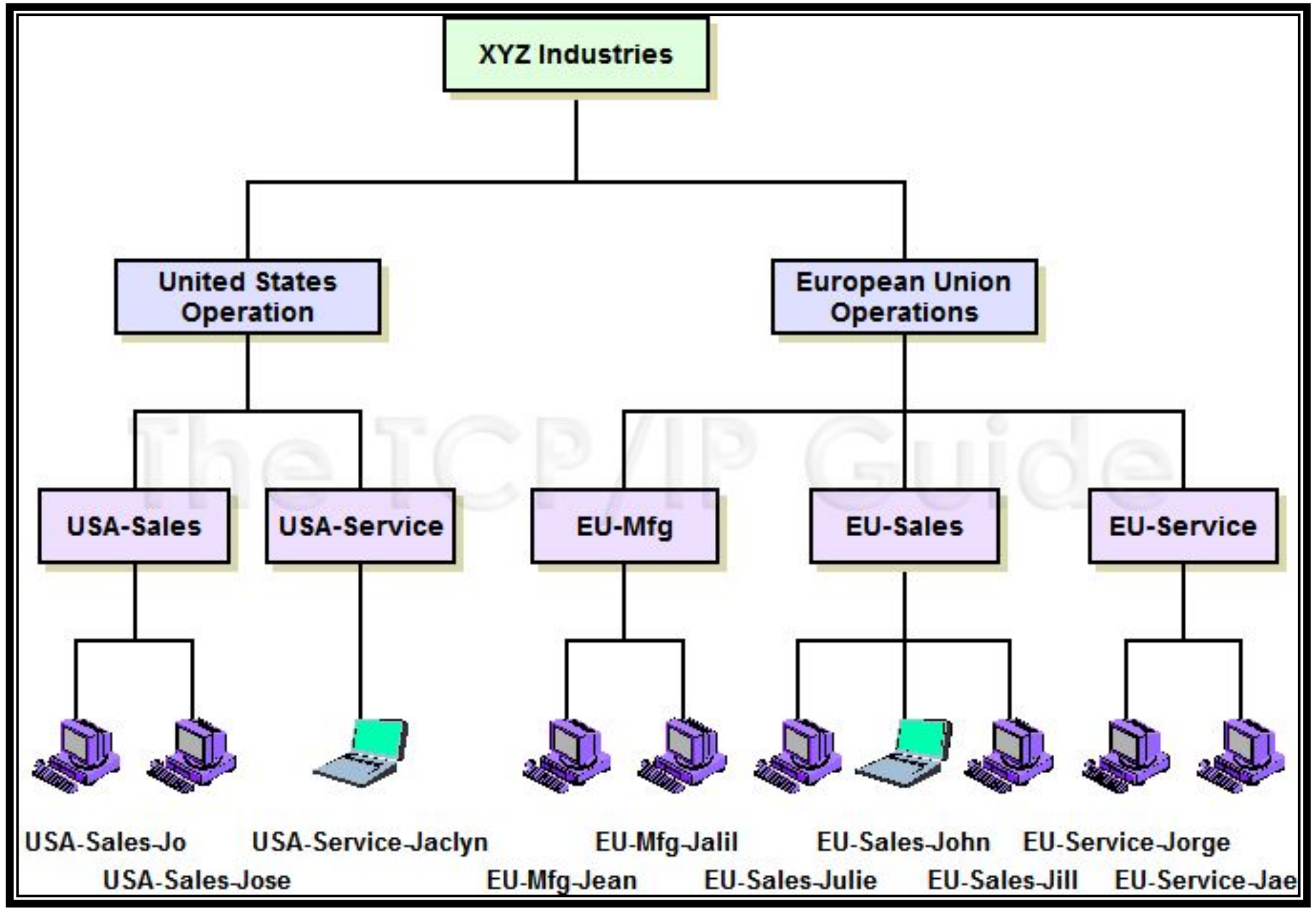
# FLAT NAME SPACE

- 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.


XYZ Industries
Jo In Sales
Jose's PC
Jaclyn
Jean, France
Jalil's Machine
Julie J.
John's Notebook
Jill
EU-Service-Jorge
Jae's PC

# HIERARCHICAL NAME SPACE

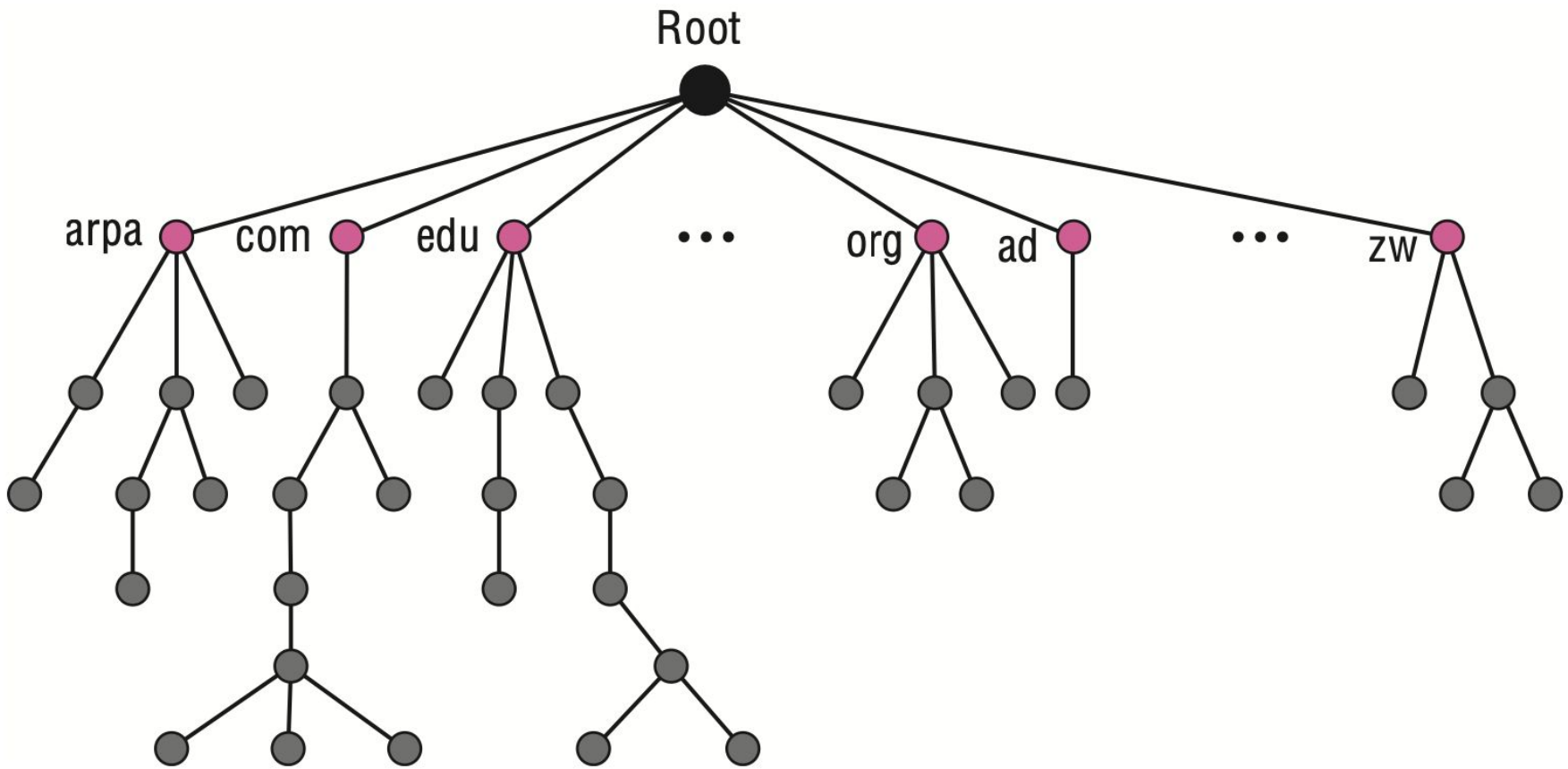
- In a hierarchical name space, each name is made of several parts.
- The first part can define the nature of the organization, the second part can define the name of an organization, the third part can define departments in the organization, and so on.


XYZ Industries
United States Operation
European Union Operations
USA-Sales
USA-Service
EU-Mfg
EU-Sales
EU-Service
USA-Sales-Jo
USA-Service-Jaclyn
EU-Mfg-Jalil
EU-Sales-John
EU-Service-Jorge
USA-Sales-Jose
EU-Mfg-Jean
EU-Sales-Julie
EU-Sales-Jill
EU-Service-Jae

# 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 19.2** *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 19.3** *Domain names and labels*

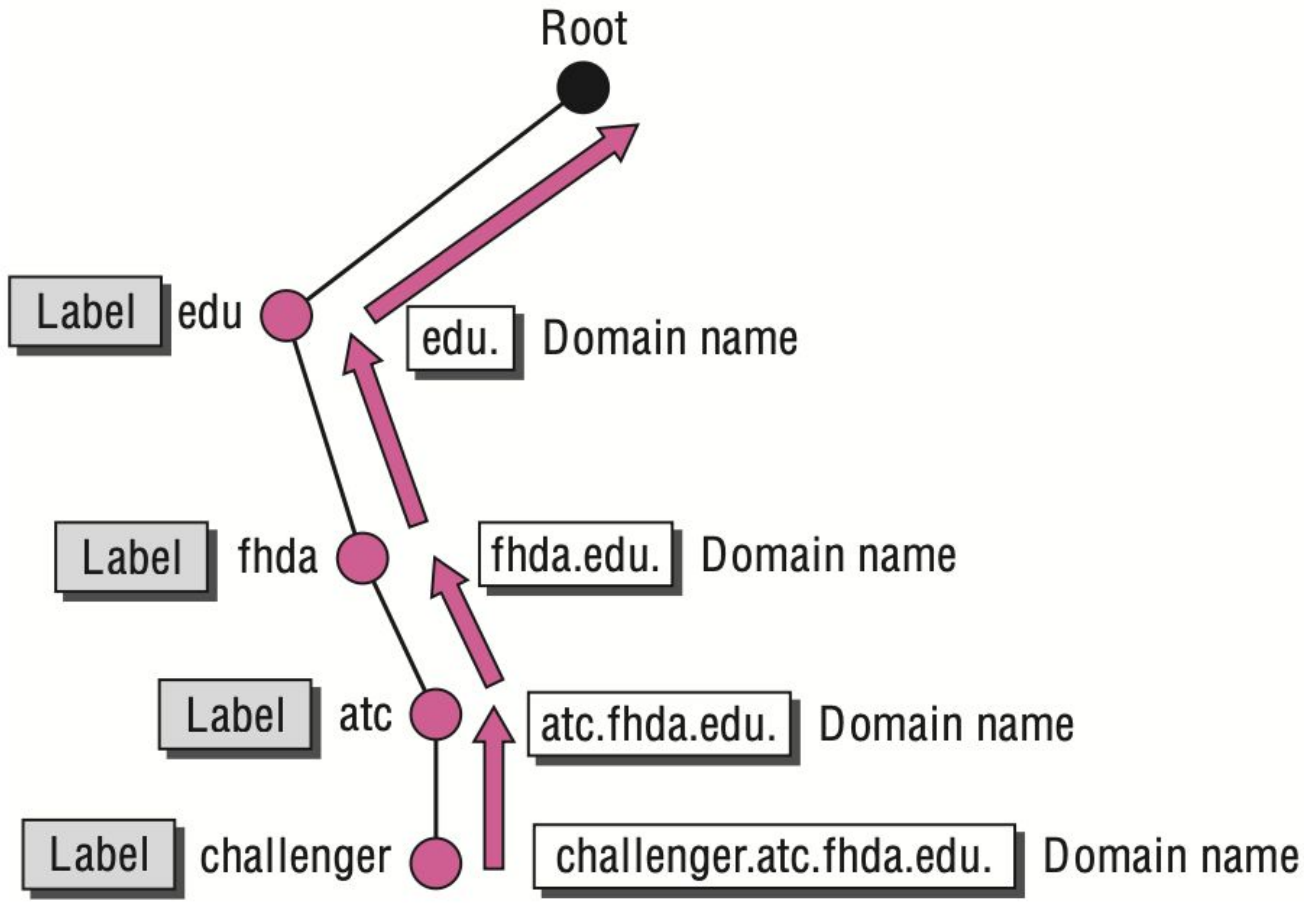
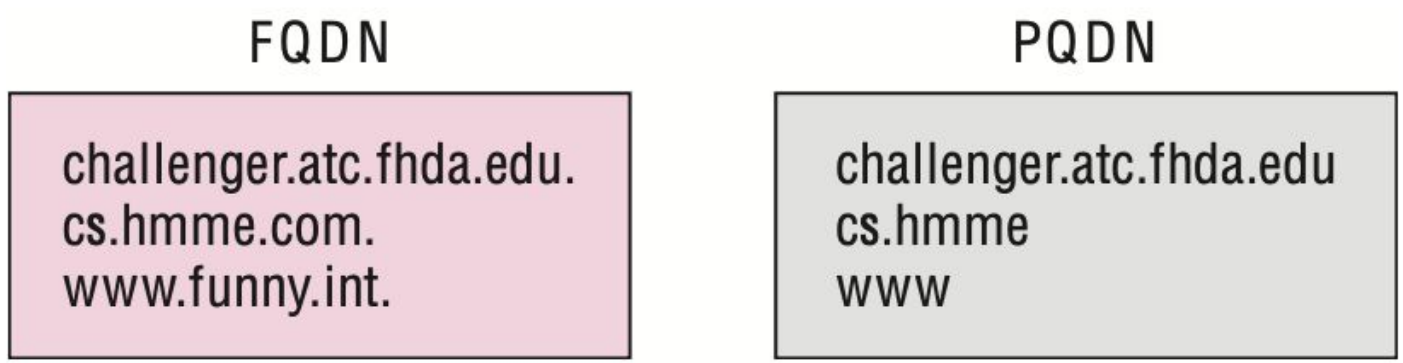


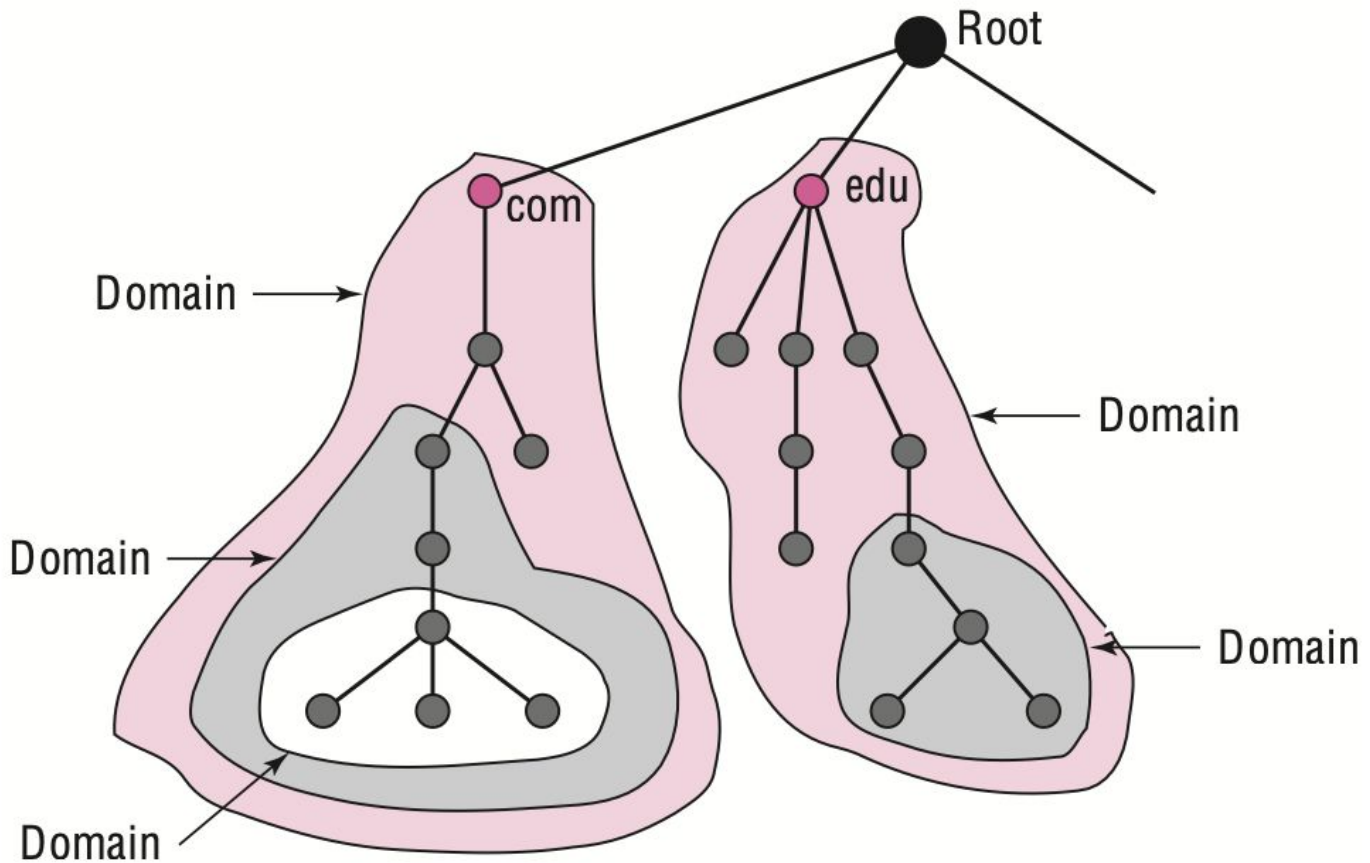
**Figure 19.4** *FQDN and PQDN*



# 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.
- Note that a domain may itself be divided into domains (or subdomains as they are sometimes called).

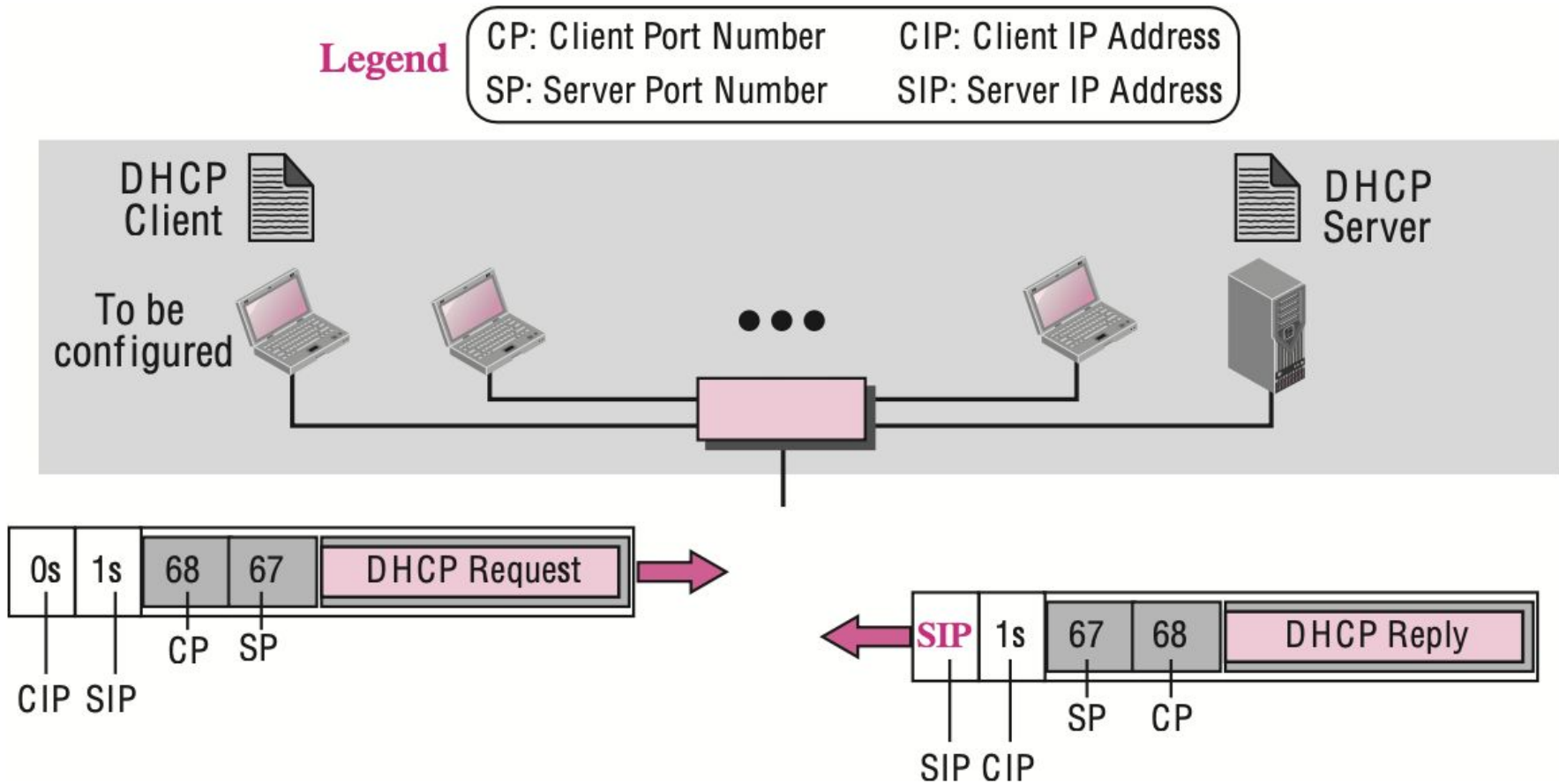
**Figure 19.5** *Domains*



# DHCP

- Dynamic Host Configuration Protocol (DHCP) is a client/server protocol that automatically provides an Internet Protocol (IP) host with its IP address and other related configuration information such as the subnet mask and default gateway.
- The DHCP client and server can either be on the same network or on different networks.
- **Same Network**
- The administrator may put the client and the server on the same network.

## Figure 18.1 *Client and server on the same network*



# DHCP OPERATION

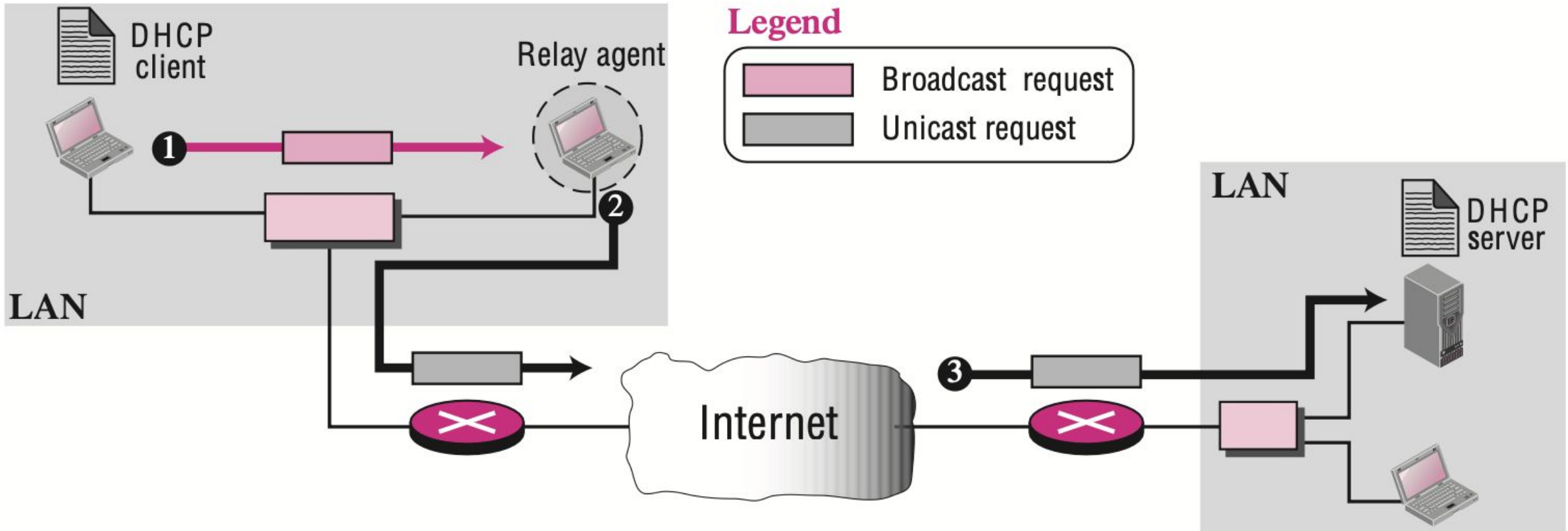
1. The DHCP server issues a passive open command on UDP port number 67 and waits for a client.
2. A booted client issues an active open command on port number 68. The message is encapsulated in a UDP user datagram, using the destination port number 67 and the source port number 68. The UDP user datagram, in turn, is encapsulated in an IP datagram. The reader may ask how a client can send an IP datagram when it knows neither its own IP address (the source address) nor the server's IP address (the destination address). The client uses all 0s as the source address and all 1s as the destination address.

# DHCP OPERATION

3. The server responds with either a broadcast or a unicast message using UDP source port number 67 and destination port number 68. The response can be unicast because the server knows the IP address of the client. It also knows the physical address of the client, which means it does not need the services of ARP for logical to physical address mapping.

- **Different Networks**
- A client can be in one network and the server in another,

**Figure 18.2** *Client and server on two different networks*



# TRANSITION STATES

- To provide dynamic address allocation, the DHCP client acts as a state machine that performs transitions from one state to another depending on the messages it receives or sends.
- The type of the message in this case is defined by the option with tag 53 that is included in the DHCP packet.

**Figure 18.7** *Option with tag 53*

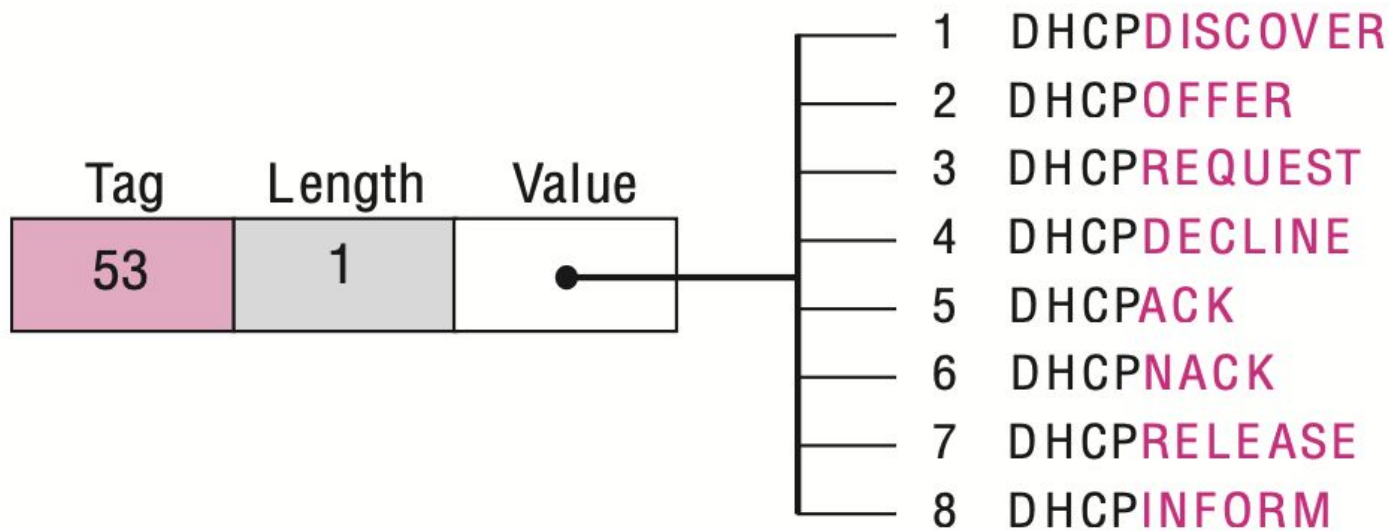
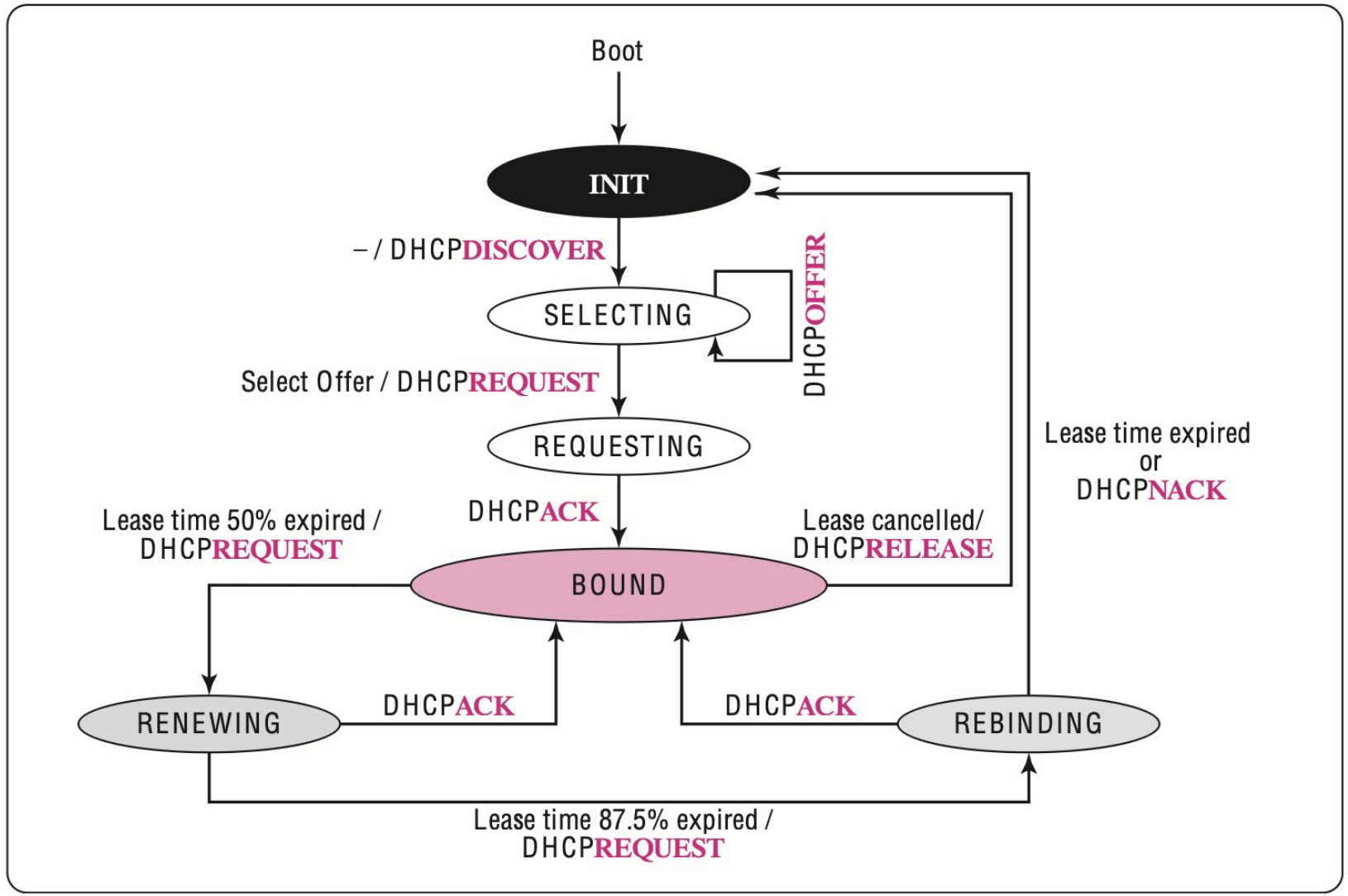


**Figure 18.8** *DHCP client transition diagram*



# INIT STATE

- When the DHCP client first starts, it is in the INIT state (initializing state).
- The client broadcasts a DHCPDISCOVER message using port 67.

# SELECTING STATE

- After sending the DHCPDISCOVER message, the client goes to the selecting state.
- Those servers that can provide this type of service respond with a DHCPOFFER message.
- In these messages, the servers offer an IP address. They can also offer the lease duration. The default is 1 hour.
- The server that sends a DHCPOFFER locks the offered IP address so that it is not available to any other clients. The client chooses one of the offers and sends a DHCPREQUEST message to the selected server. It then goes to the requesting state.

# SELECTING STATE

- However, if the client receives no DHCPOFFER message, it tries four more times, each with a span of 2 seconds.
- If there is no reply to any of these DHCPDISCOVERs, the client sleeps for 5 minutes before trying again.

# REQUESTING STATE

- The client remains in the requesting state until it receives a DHCPACK message from the server that creates the binding between the client physical address and its IP address.
- After receipt of the DHCPACK, the client goes to the bound state.

# BOUND STATE

- In this state, the client can use the IP address until the lease expires.
- When 50 percent of the lease period is reached, the client sends another DHCPREQUEST to ask for renewal.
- It then goes to the renewing state.
- When in the bound state, the client can also cancel the lease and go to the initializing state.

# RENEWING STATE

- The client remains in the renewing state until one of two events happens.
- It can receive a DHCPACK, which renews the lease agreement.
- In this case, the client resets its timer and goes back to the bound state.
- Or, if a DHCPACK is not received, and 87.5 percent of the lease time expires, the client goes to the rebinding state.

# REBINDING STATE

- The client remains in the rebinding state until one of three events happens.
- If the client receives a DHCPNACK or the lease expires, it goes back to the initializing state and tries to get another IP address.
- If the client receives a DHCPACK, it goes to the bound state and resets the timer.

# TELNET

- TELNET is an abbreviation for TErminaL NETwork.
- It is the standard TCP/IP protocol for virtual terminal service as proposed by ISO.
- TELNET enables the establishment of a connection to a remote system in such a way that the local terminal appears to be a terminal at the remote system.

# TIME-SHARING ENVIRONMENT

- TELNET was designed at a time when most operating systems, such as UNIX, were operating in a time-sharing environment.
- In such an environment, a large computer supports multiple users.
- The interaction between a user and the computer occurs through a terminal, which is usually a combination of keyboard, monitor, and mouse.

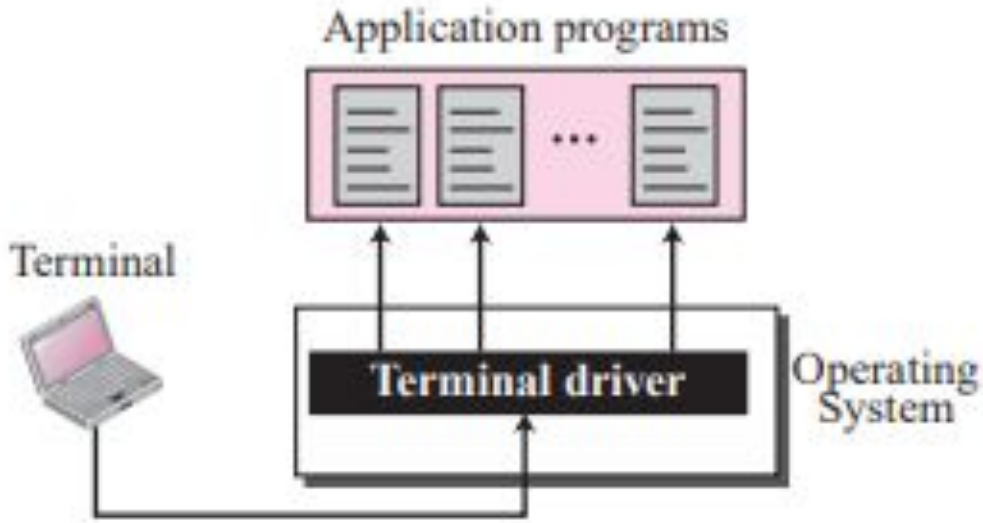
# LOGIN

- In a time-sharing environment, users are part of the system with some right to access resources.
- Each authorized user has an identification and probably a password.
- The user identification defines the user as part of the system.
- To access the system, the user logs into the system with a user id or login name.
- The system also includes password checking to prevent an unauthorized user from accessing the resources

# LOCAL LOGIN

- When a user logs into a local time-sharing system, it is called local login.
- As a user types at a terminal or at a workstation running a terminal emulator, the keystrokes are accepted by the terminal driver.
- The terminal driver passes the characters to the operating system.
- The operating system, in turn, interprets the combination of characters and invokes the desired application program or utility

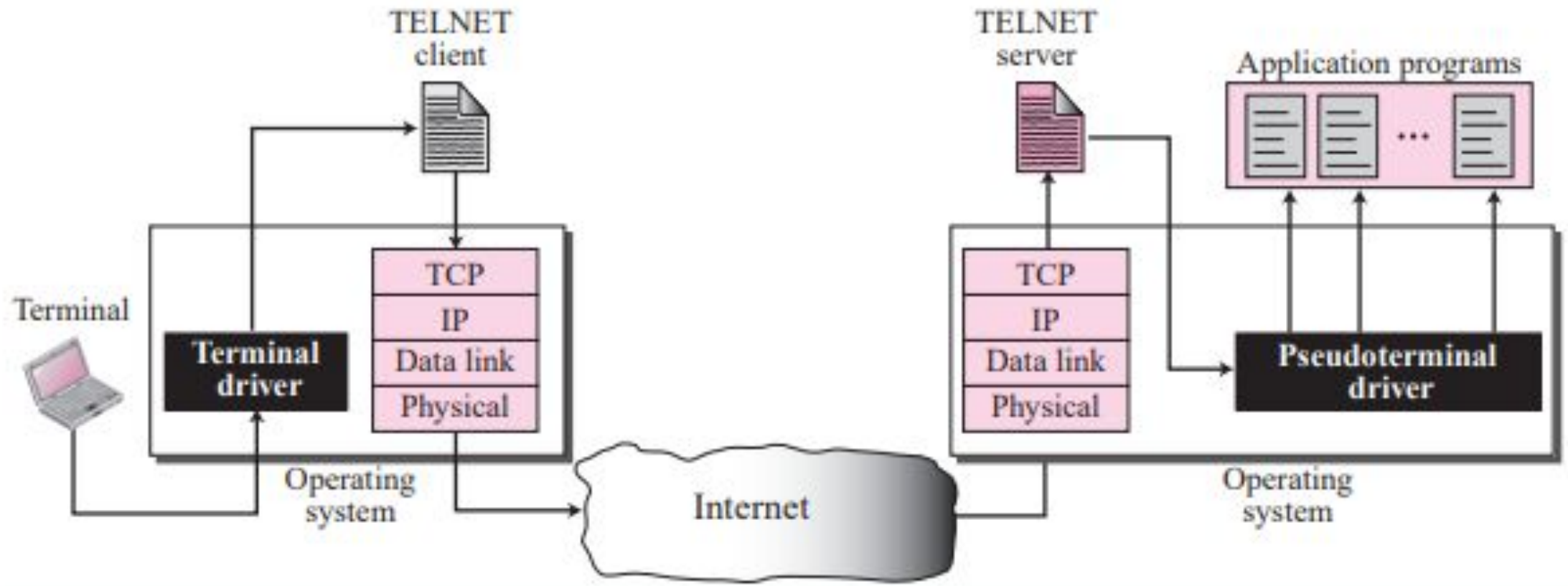
**Figure 20.1** *Local login*



# REMOTE LOGIN

- When a user wants to access an application program or utility located on a remote machine, he or she performs remote login.
- Here the TELNET client and server programs come into use.
- The user sends the keystrokes to the terminal driver where the local operating system accepts the characters but does not interpret them.
- The characters are sent to the TELNET client, which transforms the characters to a universal character set called Network Virtual Terminal (NVT) characters and delivers them to the local TCP/IP stack

**Figure 20.2** *Remote login*



# REMOTE LOGIN

- The commands or text, in NVT form, travel through the Internet and arrive at the TCP/IP stack at the remote machine.
- Here the characters are delivered to the operating system and passed to the TELNET server, which changes the characters to the corresponding characters understandable by the remote computer.
- However, the characters cannot be passed directly to the operating system because the remote operating system is not designed to receive characters from a TELNET server: It is designed to receive characters from a terminal driver.

# REMOTE LOGIN

- The solution is to add a piece of software called a pseudo terminal driver, which pretends that the characters are coming from a terminal. The operating system then passes the characters to the appropriate application program.

# NVT CHARACTER SET

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

**Figure 20.4** *Format of data and control characters*


a. Data Character

b. Control Character

**Table 20.1** *Some NVT control characters*

| *Character* | *Decimal* | *Binary* | *Meaning* |
|---|---|---|---|
| EOF | 236 | 11101100 | End of file |
| EOR | 239 | 11101111 | End of record |
| SE | 240 | 11110000 | Suboption end |
| NOP | 241 | 11110001 | No operation |
| DM | 242 | 11110010 | Data mark |
| BRK | 243 | 11110011 | Break |
| IP | 244 | 11110100 | Interrupt process |
| AO | 245 | 11110101 | Abort output |
| AYT | 246 | 11110110 | Are you there? |
| EC | 247 | 11110111 | Erase character |
| EL | 248 | 11111000 | Erase line |
| GA | 249 | 11111001 | Go ahead |
| SB | 250 | 11111010 | Suboption begin |
| WILL | 251 | 11111011 | Agreement to enable option |
| WONT | 252 | 11111100 | Refusal to enable option |
| DO | 253 | 11111101 | Approval to option request |
| DONT | 254 | 11111110 | Denial of option request |
| IAC | 255 | 11111111 | Interpret (the next character) as control |

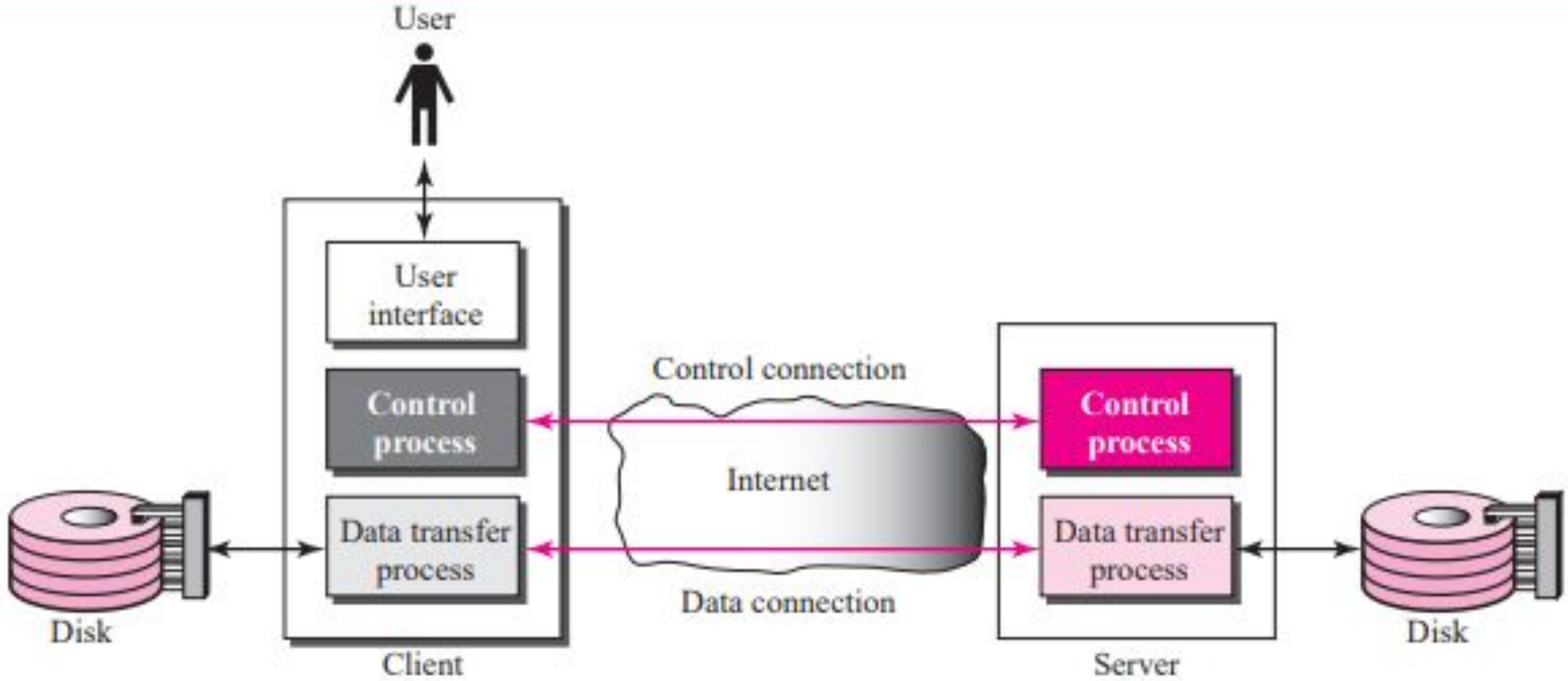
# FILE TRANSFER PROTOCOL

- File Transfer Protocol (FTP) is the standard mechanism provided by TCP/IP for copying a file from one host to another.
- Although transferring files from one system to another seems simple and straightforward, some problems must be dealt with first.
- For example, two systems may use different file name conventions. Two systems may have different ways to represent text and data.
- Two systems may have different directory structures.
- All of these problems have been solved by FTP in a very simple and elegant approach.

# FILE TRANSFER PROTOCOL

- FTP uses two well-known TCP ports: Port 21 is used for the control connection, and port 20 is used for the data connection

**Figure 21.1** *FTP*



# FTP

- The client has three components: user interface, client control process, and the client data transfer process.
- The server has two components: the server control process and the server data transfer process.
- The control connection is made between the control processes.
- The data connection is made between the data transfer processes.

# FTP

- The control connection remains connected during the entire interactive FTP session.
- The data connection is opened and then closed for each file transferred.
- It opens each time commands that involve transferring files are used, and it closes when the file is transferred.
- In other words, when a user starts an FTP session, the control connection opens. While the control connection is open, the data connection can be opened and closed multiple times if several files are transferred.

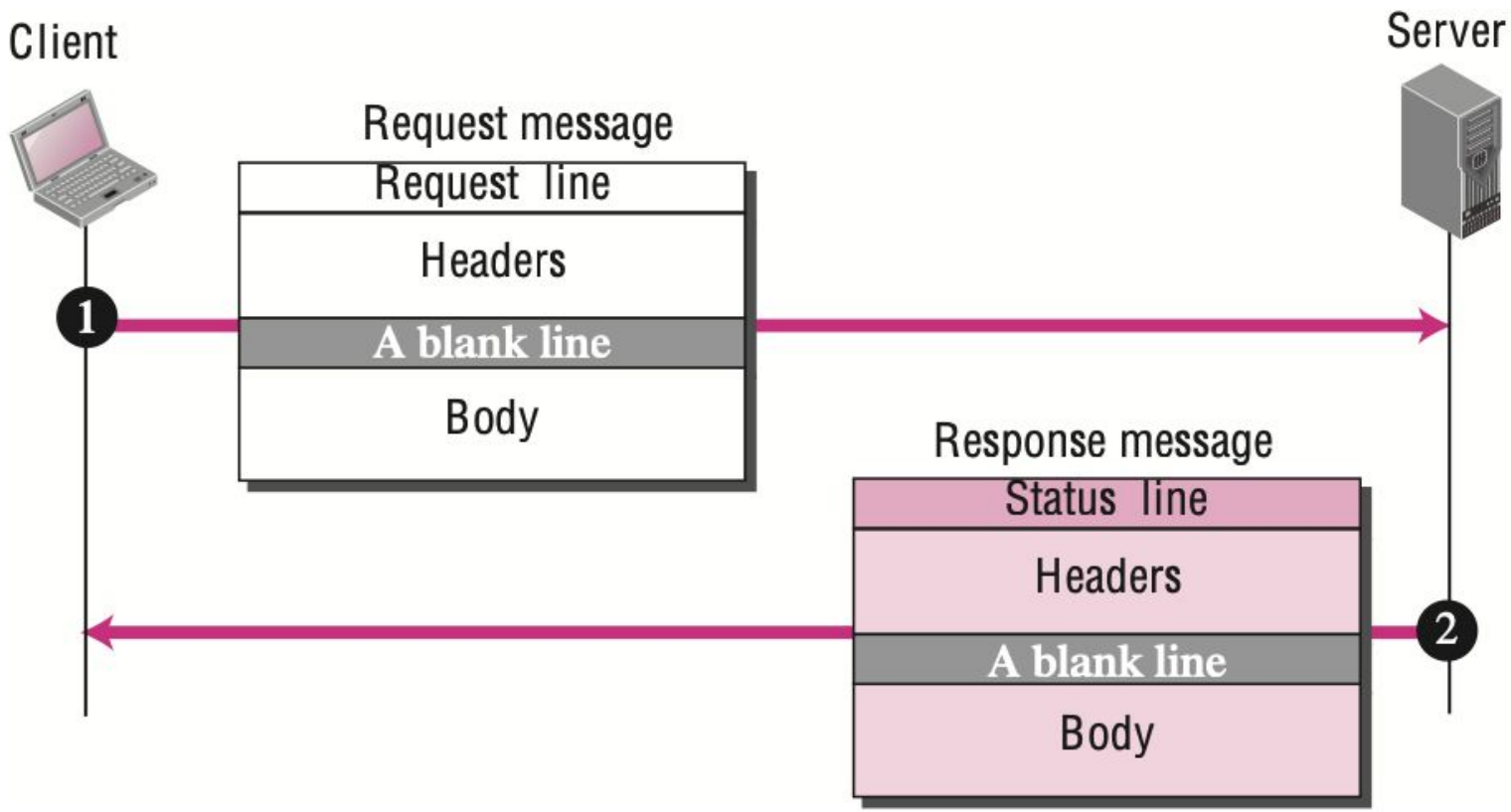
# HYPERTEXT TRANSFER PROTOCOL

- The Hypertext Transfer Protocol (HTTP) is a protocol used mainly to access data on the World Wide Web.
- It is similar to FTP because it transfers files and uses the services of TCP.
- However, it is much simpler than FTP because it uses only one TCP connection.
- There is no separate control connection; only data are transferred between the client and the server.
- HTTP uses the services of TCP on well-known port 80.

# HTTP TRANSACTION

- Although HTTP uses the services of TCP, HTTP itself is a stateless protocol, which means that the server does not keep information about the client.
- The client initializes the transaction by sending a request.
- The server replies by sending a response.

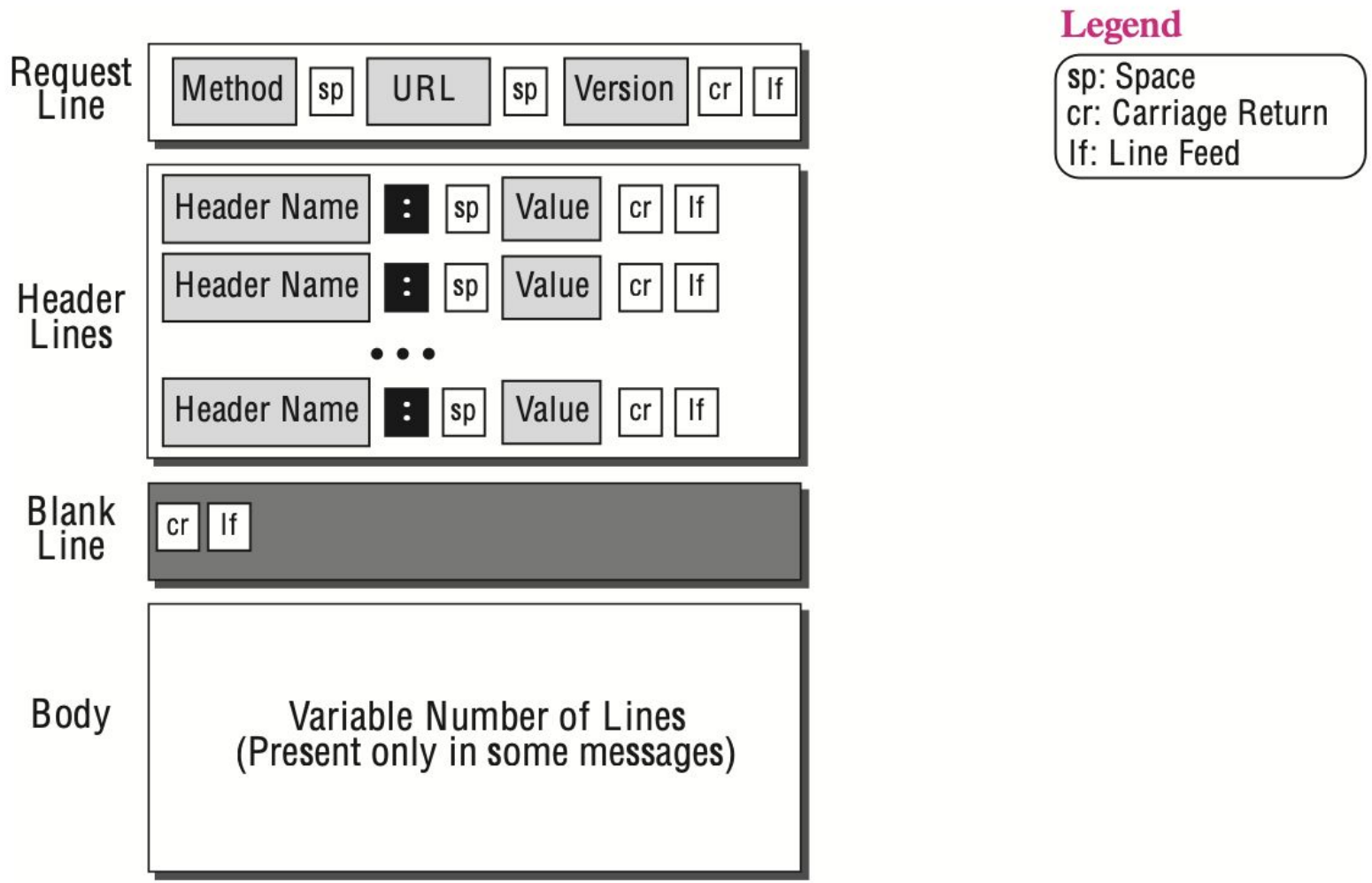
## Figure 22.10 *HTTP transaction*



# REQUEST MESSAGE

- A request message consists of a request line, a header, and sometimes a body.
- **Request Line:**
- The first line in a request message is called a request line.
- There are three fields in this line separated by some character delimiter.
- The fields are called methods, URL, and Version.
- These three should be separated by a space character.
- At the end two characters, a carriage return followed by a line feed, terminate the line. The method field defines the request type.

**Figure 22.11** *Format of the request message*



# METHOD IN REQUEST MESSAGE

- In version 1.1 of HTTP, several methods are defined.

| *Method* | *Action* |
|---|---|
| GET | Requests a document from the server |
| HEAD | Requests information about a document but not the document itself |
| POST | Sends some information from the client to the server |
| PUT | Sends a document from the server to the client |
| TRACE | Echoes the incoming request |
| CONNECT | Reserved |
| DELETE | Remove the Web page |
| OPTIONs | Enquires about available options |

# REQUEST MESSAGE

- The second field is URL.
- It defines the address and name of corresponding Web page.
- The third field, version, gives the version of the protocol; the most current version of HTTP is 1.1.

# HEADER LINES IN REQUEST MESSAGE

- After the request line, we can have zero or more request header lines.
- Each header line sends additional information from the client to the server.
- For example, the client can request that the document be sent in a special format.
- Each header line has a header name, a colon, a space, and a header value

**Table 22.2** *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 |
| If-Modified-Since | Returns the cookie to the server |

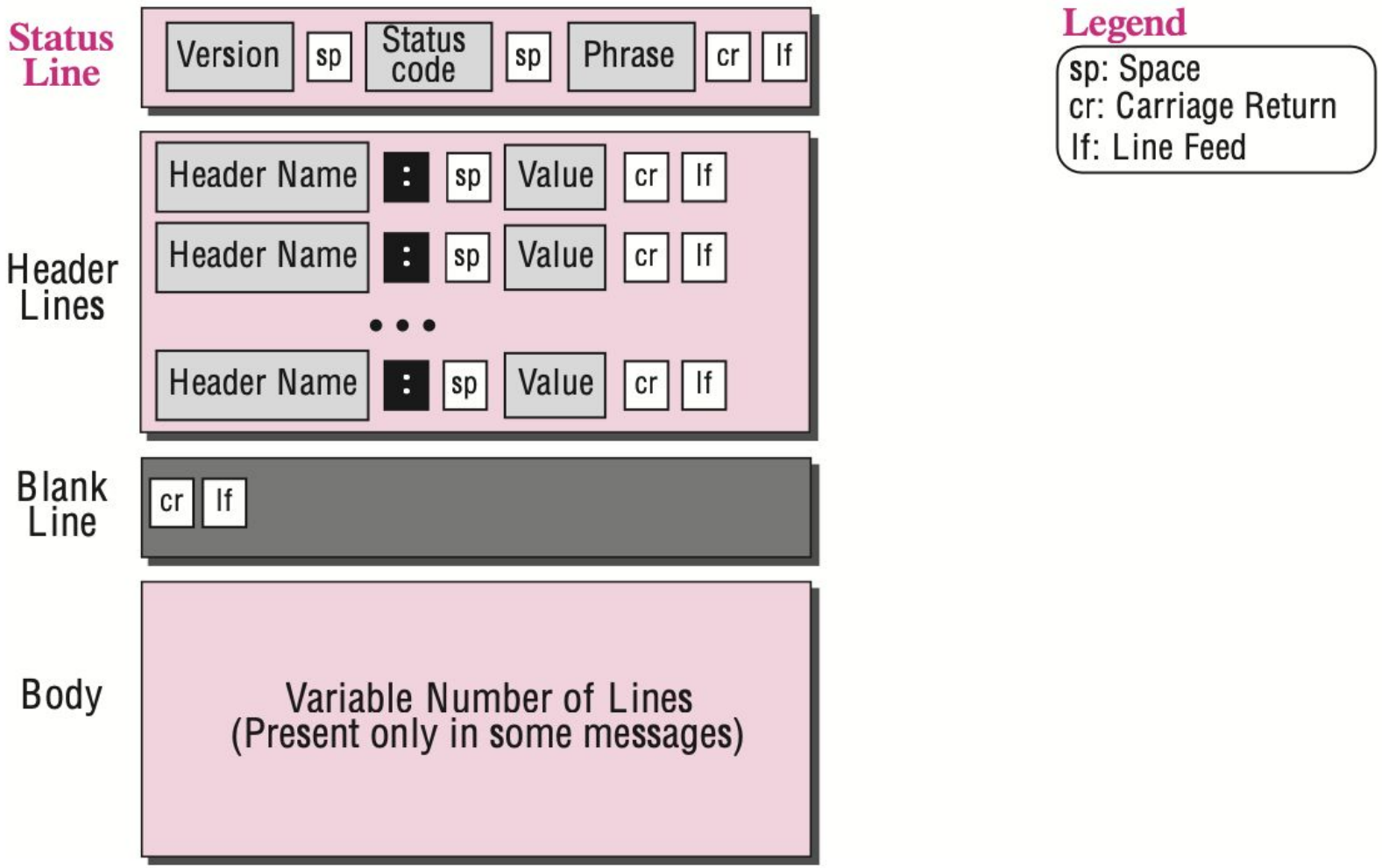
# BODY IN REQUEST MESSAGE

- The body can be present in a request message.
- Usually, it contains the comment to be sent.

# RESPONSE MESSAGE

- A response message consists of a status line, header lines, a blank line and sometimes a body.
- **Status Line:**
- The first line in a response message is called the status line.
- There are three fields in this line separated by spaces and terminated by a carriage return and line feed.
- The first field defines the version of HTTP protocol, currently 1.1.
- The status code field defines the status of the request.

**Figure 22.12** *Format of the response message*



# STATUS CODE IN RESPONSE MESSAGE

- It consists of three digits.
- Whereas the codes in the 100 range are only informational, the codes in the 200 range indicate a successful request.
- The codes in the 300 range redirect the client to another URL, and the codes in the 400 range indicate an error at the client site.
- Finally, the codes in the 500 range indicate an error at the server site.
- The status phrase explains the status code in text form.

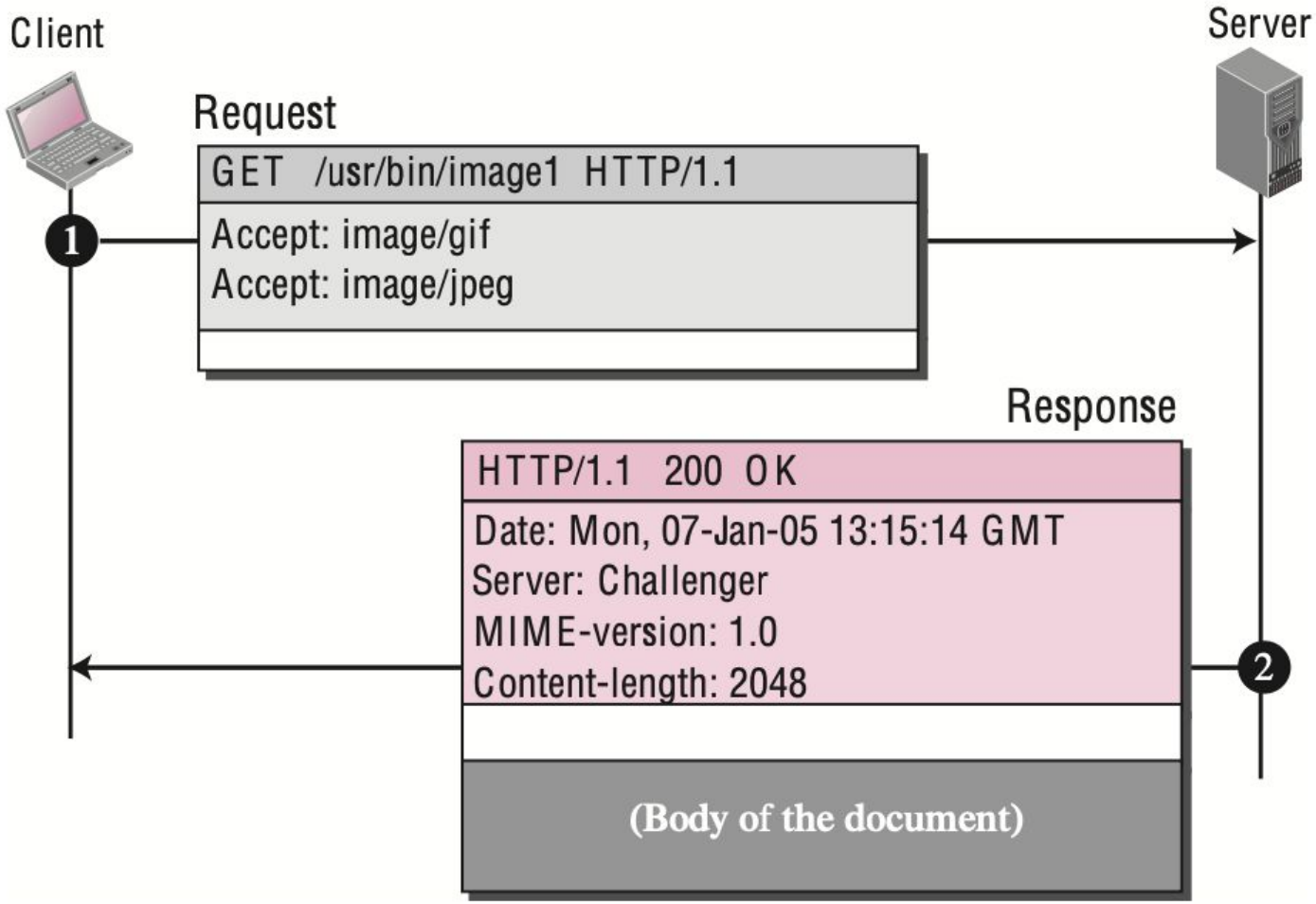
**Table 22.3** *Status Codes and Status Phrases*

| *Status Code* | *Status Phrase* | *Description* |
|---|---|---|
| | **Informational** | |
| 100 | Continue | The initial part of the request received, continue. |
| 101 | Switching | The server is complying to switch protocols. |
| | **Success** | |
| 200 | OK | The request is successful. |
| 201 | Created | A new URL is created. |
| 202 | Accepted | The request is accepted, but it is not immediately acted upon. |
| 204 | No content | There is no content in the body. |
| | **Redirection** | |
| 301 | Moved permanently | The requested URL is no longer used by the server. |
| 302 | Moved temporarily | The requested URL has moved temporarily. |
| 304 | Not modified | The document has not modified. |
| | **Client Error** | |
| 400 | Bad request | There is a syntax error in the request. |
| 401 | Unauthorized | The request lacks proper authorization. |
| 403 | Forbidden | Service is denied. |
| 404 | Not found | The document is not found. |
| 405 | Method not allowed | The method is not supported in this URL. |
| 406 | Not acceptable | The format requested is not acceptable. |
| | **Server Error** | |
| 500 | Internal server error | There is an error, such as a crash, at the server site. |
| 501 | Not implemented | The action requested cannot be performed. |
| 503 | Service unavailable | The service is temporarily unavailable. |

**Table 22.4** *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 22.13** *Example 22.4*



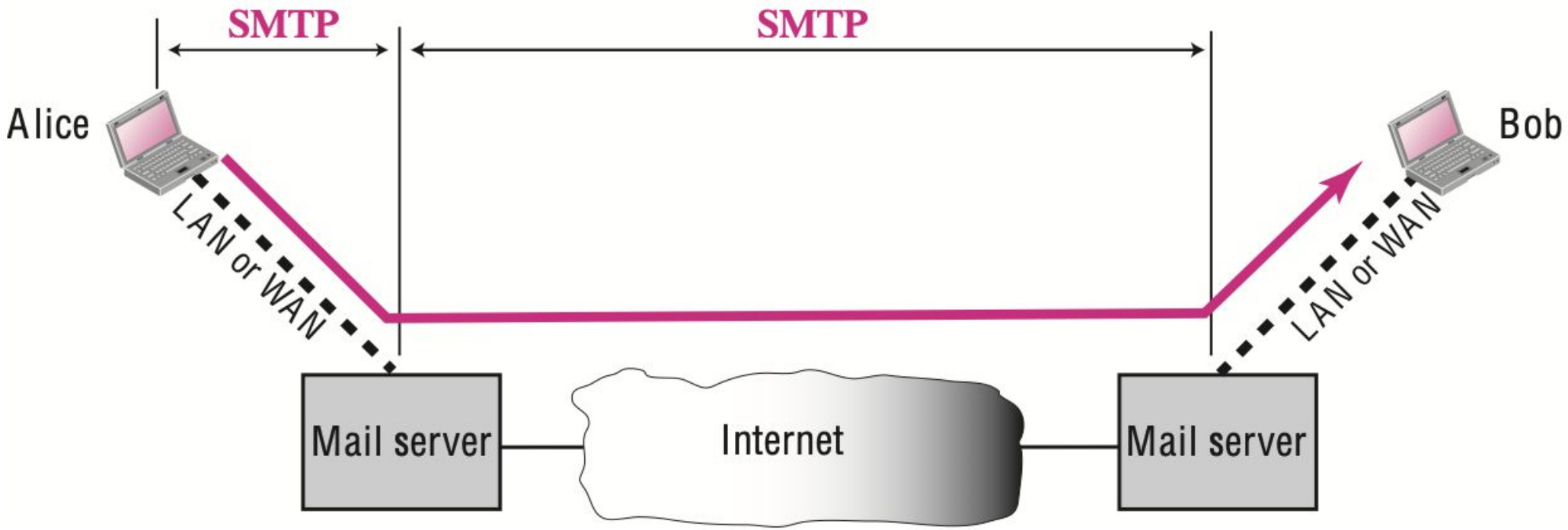
# HTTP SECURITY

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

# SMTP

- The actual mail transfer is done through message transfer agents (MTAs).
- To send mail, a system must have the client MTA, and to receive mail, a system must have a server MTA.
- The formal protocol that defines the MTA client and server in the Internet is called Simple Mail Transfer Protocol (SMTP).

**Figure 23.8** *SMTP range*



# SMTP

- SMTP is used two times, between the sender and the sender's mail server and between the two mail servers.
- **Commands and Responses**
- SMTP uses commands and responses to transfer messages between an MTA client and an MTA server

**Figure 23.9** *Commands and responses*

**Table 23.1** *Commands*

| *Keyword* | *Argument(s)* | *Keyword* | *Argument(s)* |
|---|---|---|---|
| HELO | Sender's host name | NOOP | |
| MAIL FROM | Sender of the message | TURN | |
| RCPT TO | Intended recipient | EXPN | Mailing list |
| DATA | Body of the mail | HELP | Command name |
| QUIT | | SEND FROM | Intended recipient |
| RSET | | SMOL FROM | Intended recipient |
| VRFY | Name of recipient | SMAL FROM | Intended recipient |

- **HELO.** This command is used by the client to identify itself. The argument is the domain name of the client host. The format is

**HELO:** challenger.atc.fhda.edu

**Table 23.2** *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 |
| 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 |

**Figure 23.10** *Connection establishment*

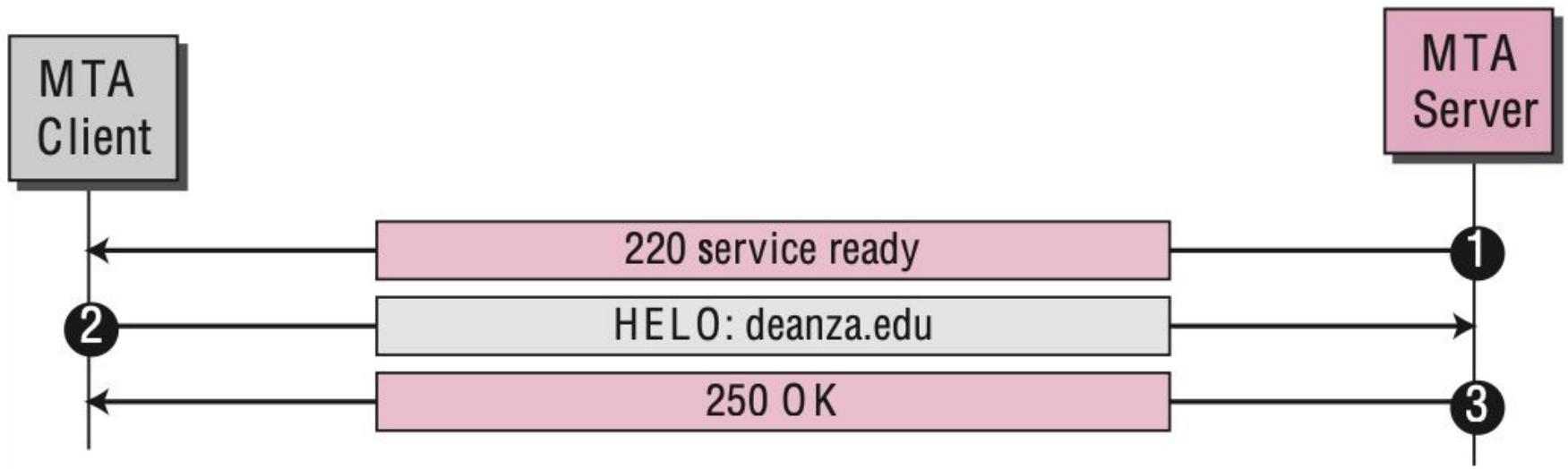


**Figure 23.11** *Message transfer*

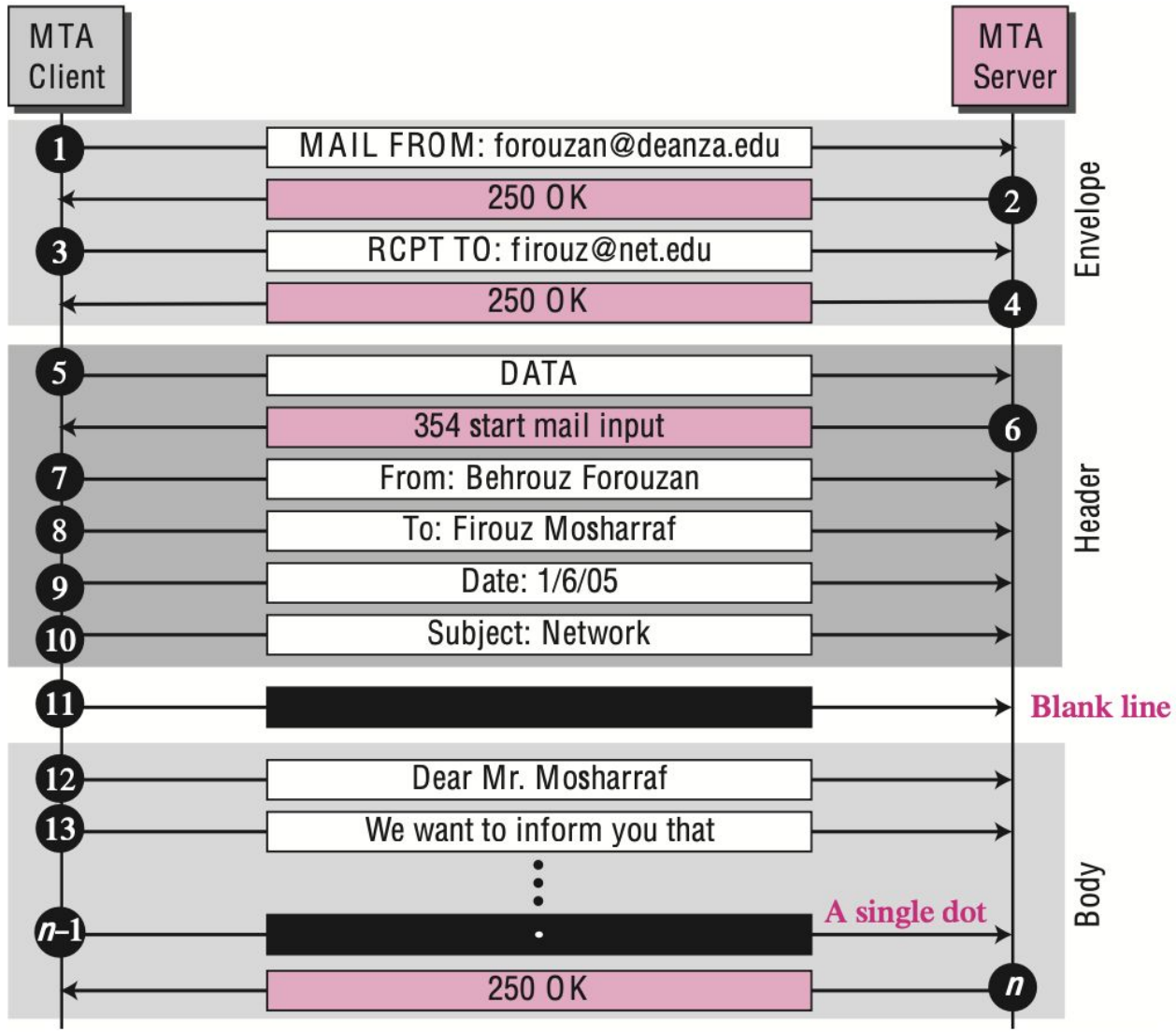
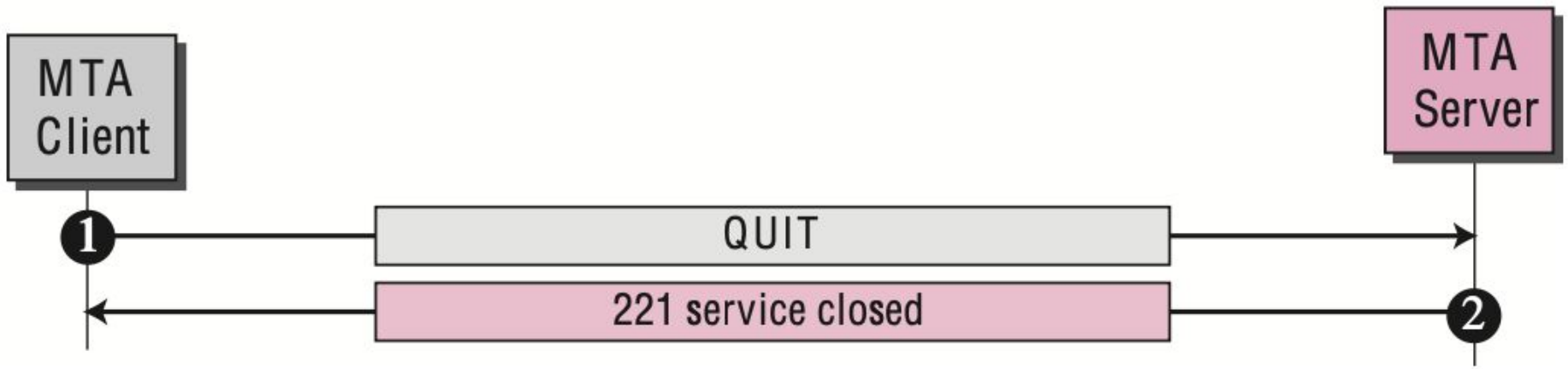


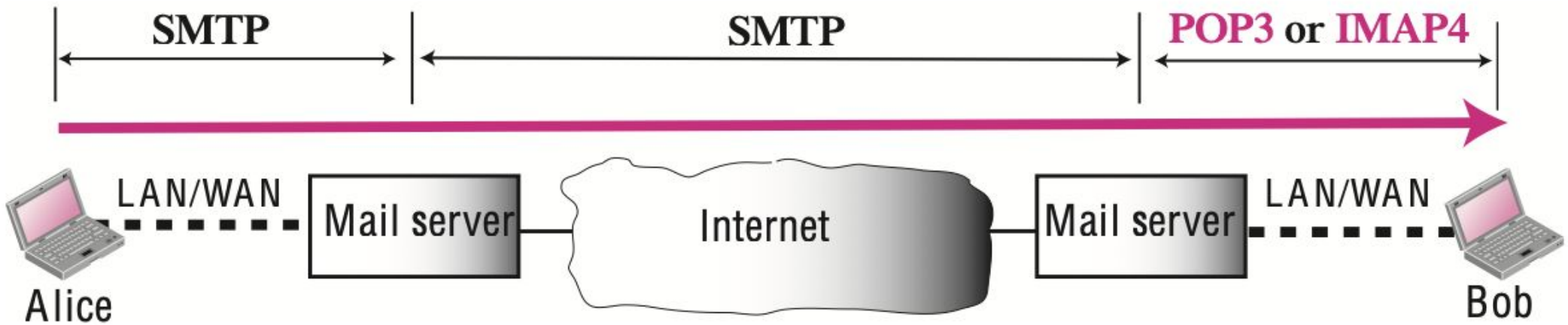
**Figure 23.12** *Connection termination*



# POST OFFICE PROTOCOL (POP)

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

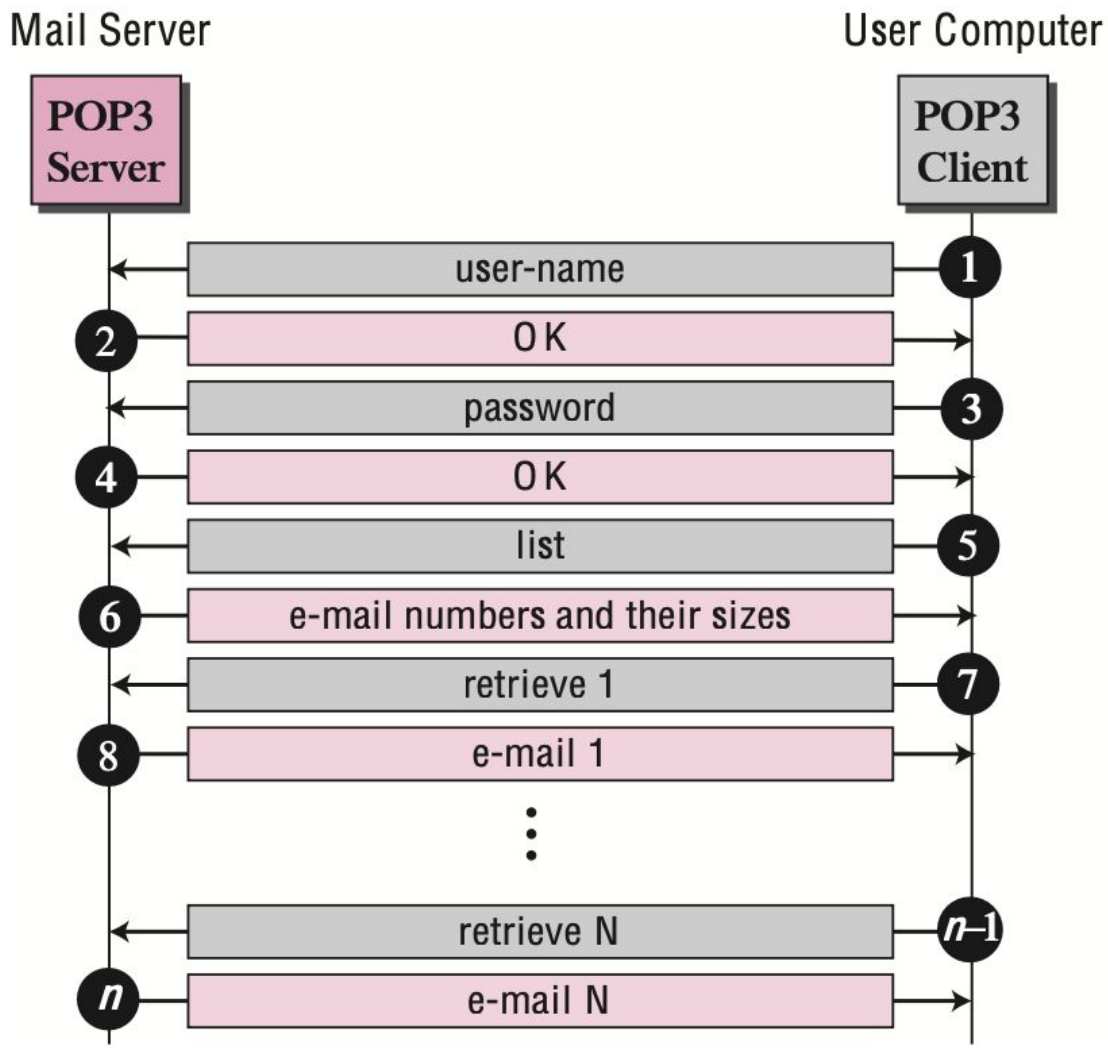
**Figure 23.13** *POP3 and IMAP4*



# POST OFFICE PROTOCOL (POP)

- Post Office Protocol, version 3 (POP3) is simple and 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 23.14** *POP3*



# POST OFFICE PROTOCOL (POP)

- 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.
- In the keep mode, the mail remains in the mailbox after retrieval.

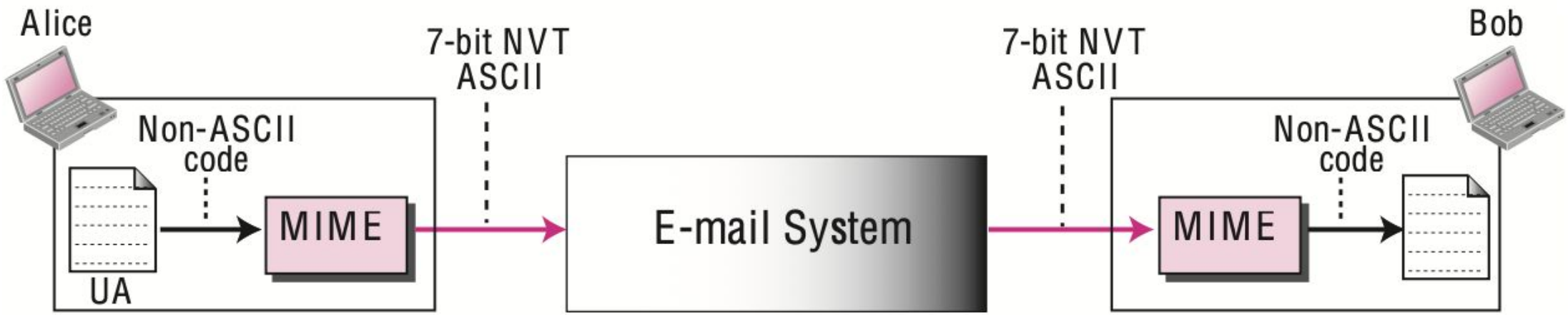
# MIME

- Electronic mail has a simple structure.
- Its simplicity, however, comes with a price.
- It can send messages only in NVT 7-bit ASCII format.
- In other words, it has some limitations. It cannot be used for languages other than English (such as French, German, Hebrew, Russian, Chinese, and Japanese).
- Also, it cannot be used to send binary files or video or audio data.
- Multipurpose Internet Mail Extensions (MIME) is a supplementary protocol that allows non-ASCII data to be sent through e-mail.

# MIME

- MIME transforms non-ASCII data at the sender site to NVT ASCII data and delivers it to the client MTA to be sent through the Internet.
- The message at the receiving site is transformed back to the original data.
- MIME is a set of software functions that transforms non-ASCII data to ASCII data and vice versa.

## Figure 23.15 *MIME*



# MIME HEADERS

- MIME defines five headers that can be added to the original e-mail header section to define the transformation parameters:
  1. MIME-Version
  2. Content-Type
  3. Content-Transfer-Encoding
  4. Content-Id
  5. Content-Description

**Figure 23.16** *MIME header*

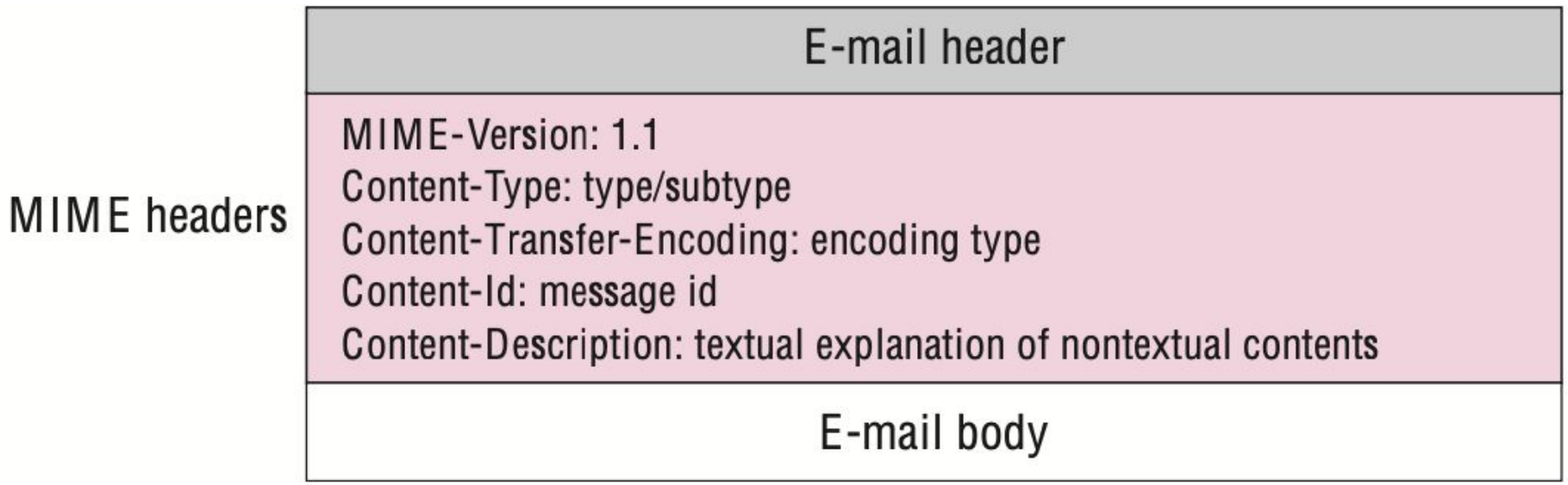


**Table 23.3** *Data Types and Subtypes in MIME*

| *Type* | *Subtype* | *Description* |
|---|---|---|
| Text | Plain | Unformatted |
| | HTML | HTML format (see Appendix E) |
| 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 23.4** *Content-Transfer-Encoding*

| *Type* | *Description* |
|---|---|
| 7bit | NVT ASCII characters and short lines |
| 8bit | Non-ASCII characters and short lines |
| Binary | Non-ASCII characters with unlimited-length lines |
| Base64 | 6-bit blocks of data are encoded into 8-bit ASCII characters |
| Quoted-printable | Non-ASCII characters are encoded as an equal sign plus an ASCII code |

## Figure 23.17 *Base64*

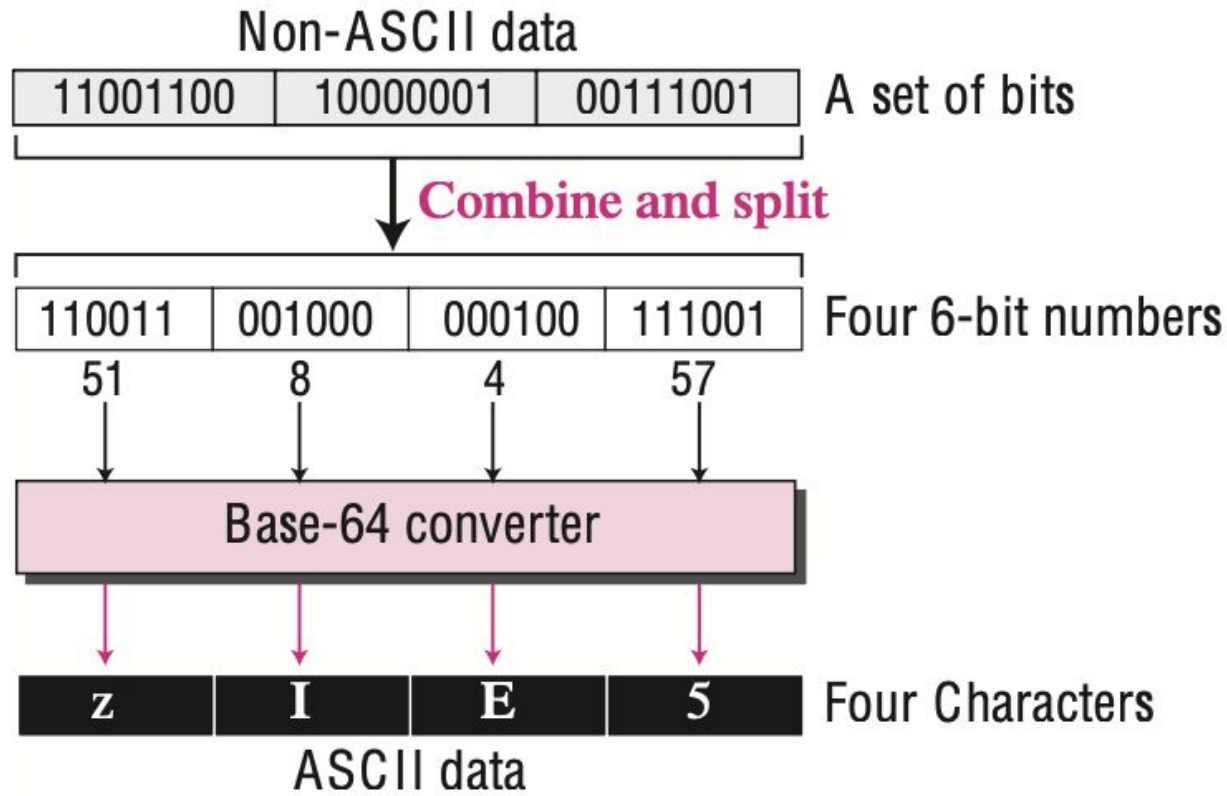
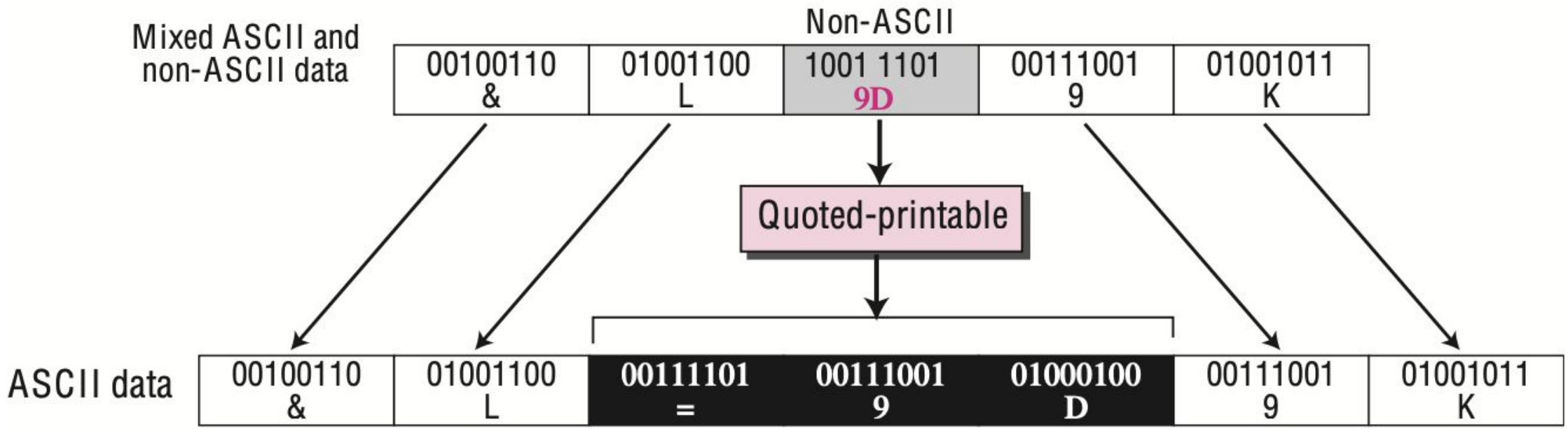


**Table 23.5** *Base-64 Converting Table*

| *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 | | |

**Figure 23.18** *Quoted-printable*



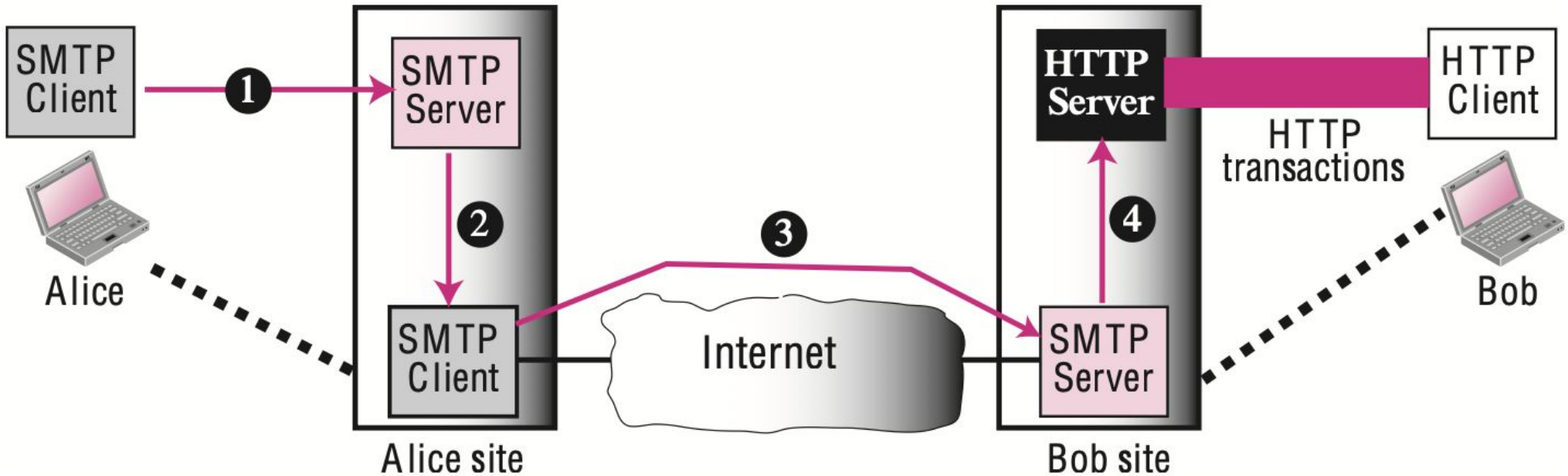
# WEB-BASED MAIL

- **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.
- Mail transfer from Alice's browser to her mail server is done through SMTP.
- The transfer of the message from the sending mail server to the receiving mail server is still through SMTP.
- However, the message from the receiving server (the web server) to Bob's browser is done through HTTP.

# WEB-BASED MAIL

- In other words, instead of using POP3 or IMAP4, HTTP is normally used.
- When Bob needs to retrieve his e-mails, he sends a request HTTP message to the website (Gmail.com, 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.

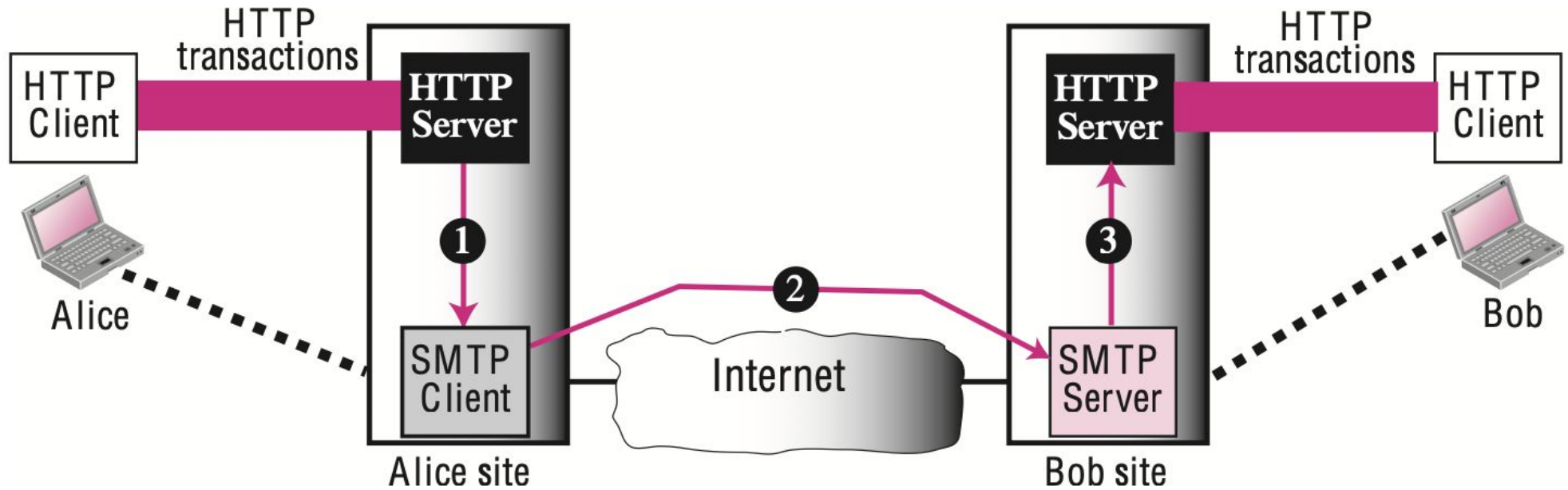
**Figure 23.19** *Web-base e-mail, case I*



# CASE II

- In the second case, 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 mail- box 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 23.20** *Web-based e-mail, case II*



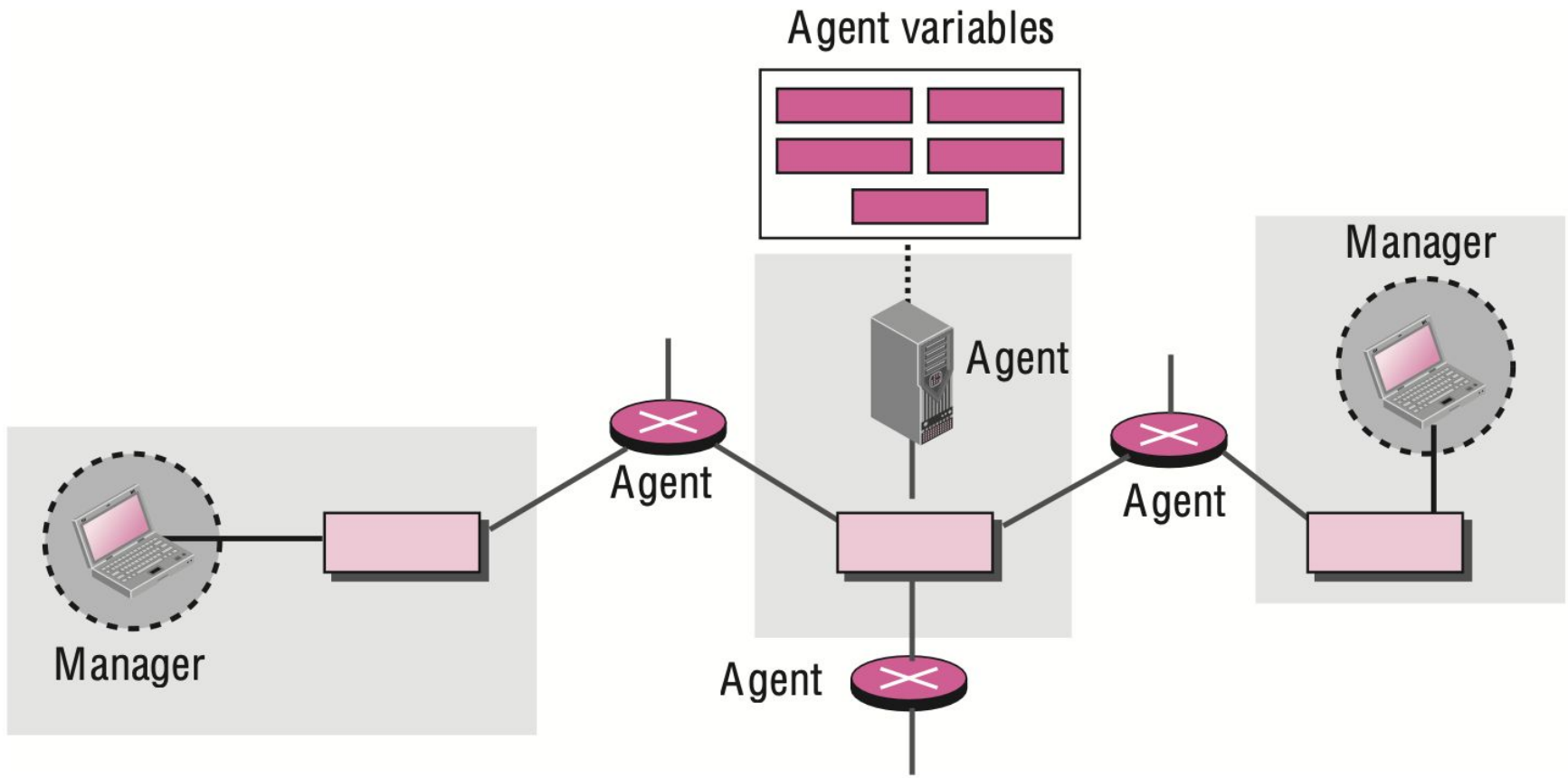
# E-MAIL SECURITY

- E-mail exchanges can be secured using two application-layer securities designed in particular for e-mail systems.
- Two of these protocols, Pretty Good Privacy (PGP) and Secure MIME (SMIME)

# SIMPLE NETWORK MANAGEMENT PROTOCOL

- SNMP uses the concept of manager and agent.
- That is, a manager, usually a host, controls and monitors a set of agents, usually routers or servers.
- SNMP is an application-level protocol in which a few manager stations control a set of agents.
- The protocol is designed at the application level so that it can monitor devices made by different manufacturers and installed on different physical networks.
- It can be used in a heterogeneous internet made of different LANs and WANs connected by routers made by different manufacturers.

**Figure 24.1** *SNMP concept*



# MANAGERS AND AGENTS

- A management station, called a manager, is a host that runs the SNMP client program.
- A managed station, called an agent, is a router (or a host) that runs the SNMP server program.
- Management is achieved through simple interaction between a manager and an agent.
- The agent keeps performance information in a database.
- The manager has access to the values in the database.
- For example, a router can store in appropriate variables the number of packets received and forwarded.

# MANAGERS AND AGENTS

- The manager can fetch and compare the values of these two variables to see if the router is congested or not.
- The manager can also make the router perform certain actions.
- For example, a router periodically checks the value of a reboot counter to see when it should reboot itself.
- It reboots itself, for example, if the value of the counter is 0.
- The manager can use this feature to reboot the agent remotely at any time.
- It simply sends a packet to force a 0 value in the counter.

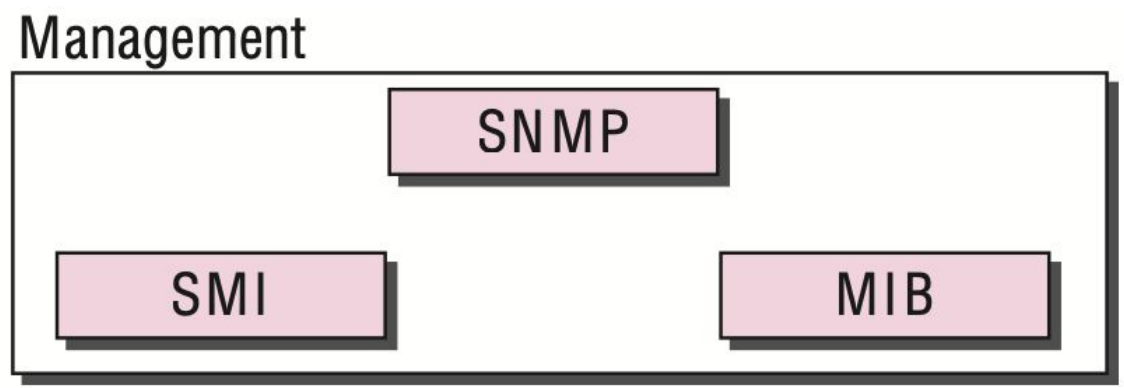
# MANAGERS AND AGENTS

- Agents can also contribute to the management process.
- The server program running on the agent can check the environment and, if it notices something unusual, it can send a warning message (called a trap) to the manager.

# MANAGEMENT COMPONENTS

- To do management tasks, SNMP uses two other protocols: Structure of Management Information (SMI) and Management Information Base (MIB).
- In other words, management on the Internet is done through the cooperation of three protocols: SNMP, SMI, and MIB

**Figure 24.2** *Components of network management on the Internet*



# ROLE OF SNMP

- SNMP has some very specific roles in network management.
- It defines the format of the packet to be sent from a manager to an agent and vice versa.
- It also interprets the result and creates statistics (often with the help of other management software).
- The packets exchanged contain the object (variable) names and their status (values).
- SNMP is responsible for reading and changing these values.

# ROLE OF SMI

- To use SNMP, we need rules. We need rules for naming objects.
- This is particularly important because the objects in SNMP form a hierarchical structure (an object may have a parent object and some child objects).
- Part of a name can be inherited from the parent. We also need rules to define the type of the objects.
- SMI is a protocol that defines these rules. However, we must understand that SMI only defines the rules; it does not define how many objects are managed in an entity or which object uses which type.
- SMI is a collection of general rules to name objects and to list their types.

# ROLE OF MIB

- For each entity to be managed, this protocol must define the number of objects, name them according to the rules defined by SMI, and associate a type to each named object.
- MIB creates a set of objects defined for each entity similar to a database
- MIB creates a collection of named objects, their types, and their relationships to each other in an entity to be managed.

**Figure 24.3** *Comparing computer programming and network management*