Chapter 2

APPLICATION LAYER

Topics

- Domain Name Space (DNS)
- Electronic Mail
- HTTP
- Delay and throughput in Packet- switched Network

Introduction

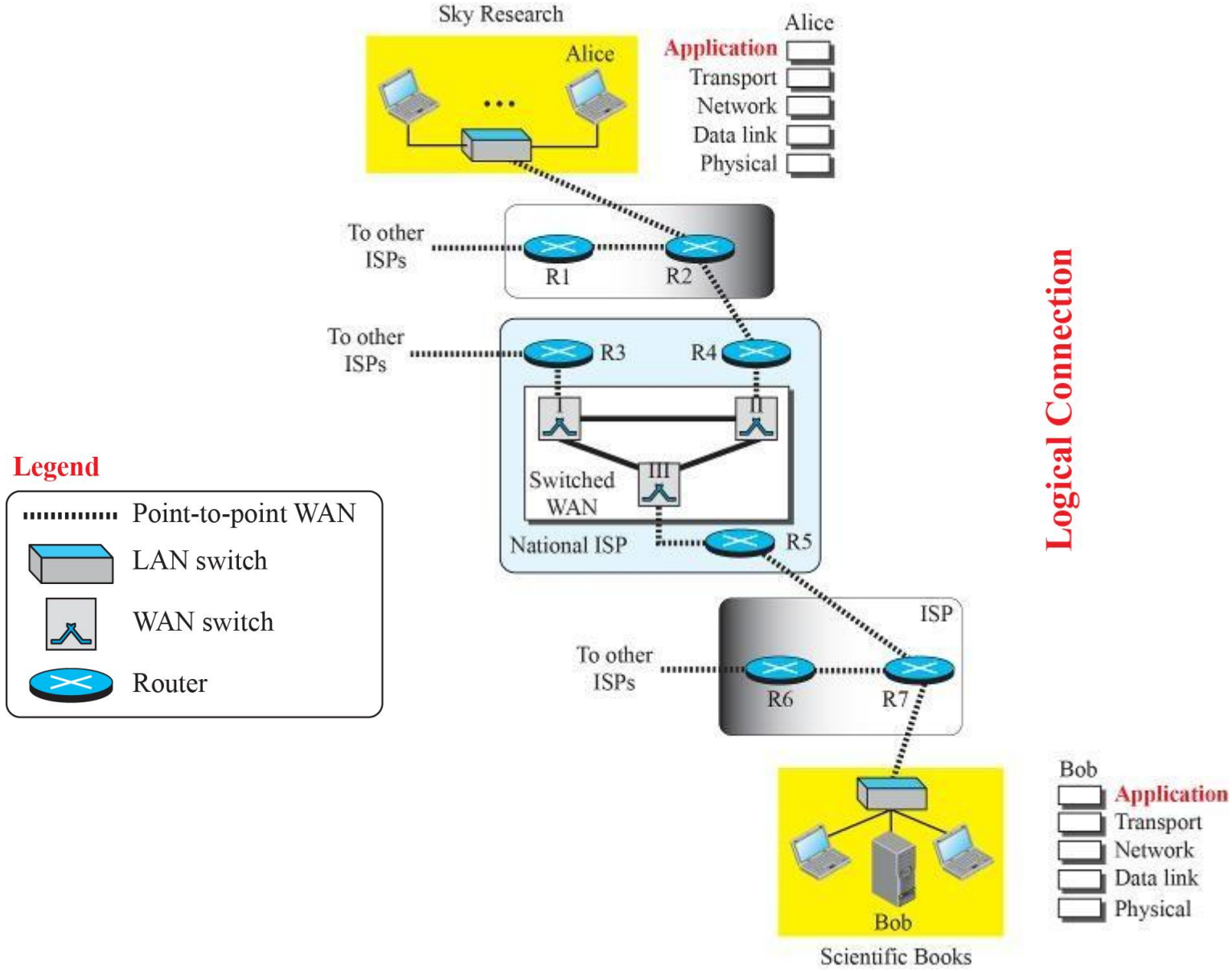
The Application Layer is topmost layer in the Open System Interconnection (OSI) model.

This layer provides several ways for manipulating the data (information) which actually enables any type of user to access network with ease.

This layer also makes a request to its bottom layer, which is presentation layer for receiving various types of information from it.

The Application Layer interface directly interacts with application and provides common web application services.

This layer is basically highest level of open system, which provides services directly for application process.


Sky Research
Alice
Alice
Application
Transport
Network
Data link
Physical
To other ISPs
R1
R2
To other ISPs
R3
R4
I
II
III
Switched WAN
National ISP
R5
Legend
Point-to-point WAN
LAN switch
WAN switch
Router
ISP
To other ISPs
R6
R7
Bob
Scientific Books
Bob
Application
Transport
Network
Data link
Physical
Logical Connection

Providing services

- Internet's Original Purpose:
  - Global service provision to worldwide users.
- Significance of Application Layer:
  - Sole provider of services to Internet users.
  - Enables seamless addition of new application protocols.
- Evolution of Application Protocols:
  - Internet's lifetime was marked by protocol expansion.
  - Initial availability of a limited number of protocols.
  - Constant addition of new protocols over time.

Functions of Application Layer

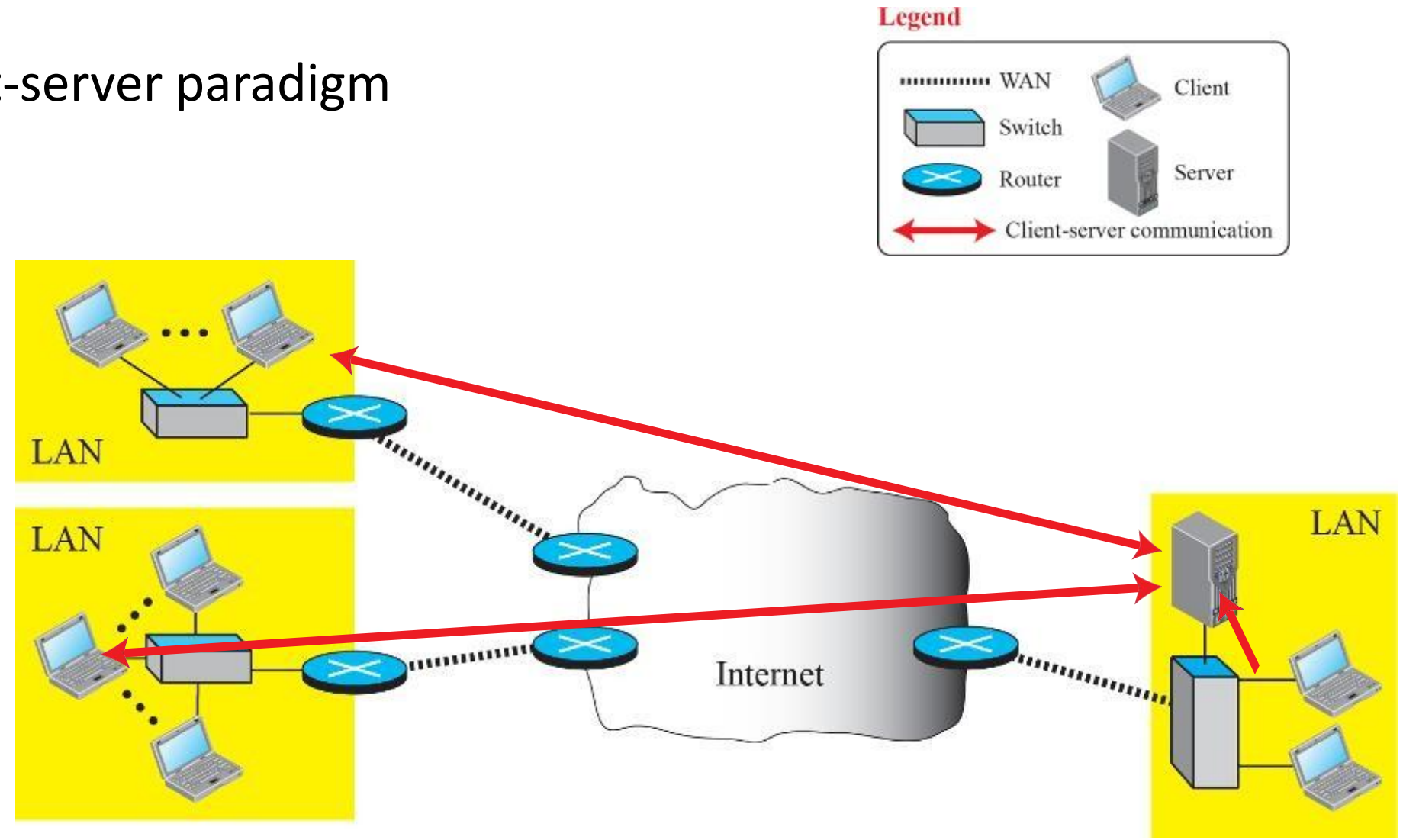
Application Layer provides a facility by which users can forward several **emails** and it also provides a **storage facility**. This layer allows users to access, retrieve and manage **files** in a remote computer. It allows users to log on as a **remote host**. It provides protocols that allow software to send and receive information and present meaningful data to users.
It handles issues such as **network transparency**, **resource allocation** and so on. The application layer is actually an **abstraction layer** that specifies the **shared protocols** and **interface methods** used by hosts in a communication network.

This application layer basically **interacts with Operating System (OS)** and thus further preserves the data in a suitable manner.

This layer allows users to **interact** with other **software applications**.

In this layer, data is in visual form, which makes users truly understand data rather than remembering or visualize the data in the binary format (0's or 1's).

Client-server paradigm



Client-Server Model/Architecture

The Client-server model is a distributed application structure that partitions task or workload between the providers of a resource or service, called **servers**, and service requesters called **clients**.

In the client-server architecture, when the client computer sends a request for data to the server through the internet, the server accepts the requested process and deliver the data packets requested back to the client.

Clients do not share any of their resources.

Examples of Client-Server Model are Email, World Wide Web, etc.

How the browser interacts with the servers ?

There are few steps to follow to interact client with the servers.

User enters the **URL**(Uniform Resource Locator) of the website or file. The Browser then requests the **DNS**(DOMAIN NAME SYSTEM) Server.

DNS Server lookup for the address of the **WEB Server**.

DNS Server responds with the **IP address** of the **WEB Server**.

Browser sends over an **HTTP/HTTPS** request to **WEB Server's IP** (provided by **DNS server**).

Server sends over the necessary files of the website.

Browser then renders the files and the website is displayed. This rendering is done with the help of **DOM** (Document Object Model) interpreter, **CSS** interpreter and **JS Engine** collectively known as the **JIT** or (Just in Time) Compilers.

Advantages of Client-Server model:

Centralized system with all data in a single place.

Cost efficient requires less maintenance cost and Data recovery is possible.

The capacity of the Client and Servers can be changed separately.

Disadvantages of Client-Server model:

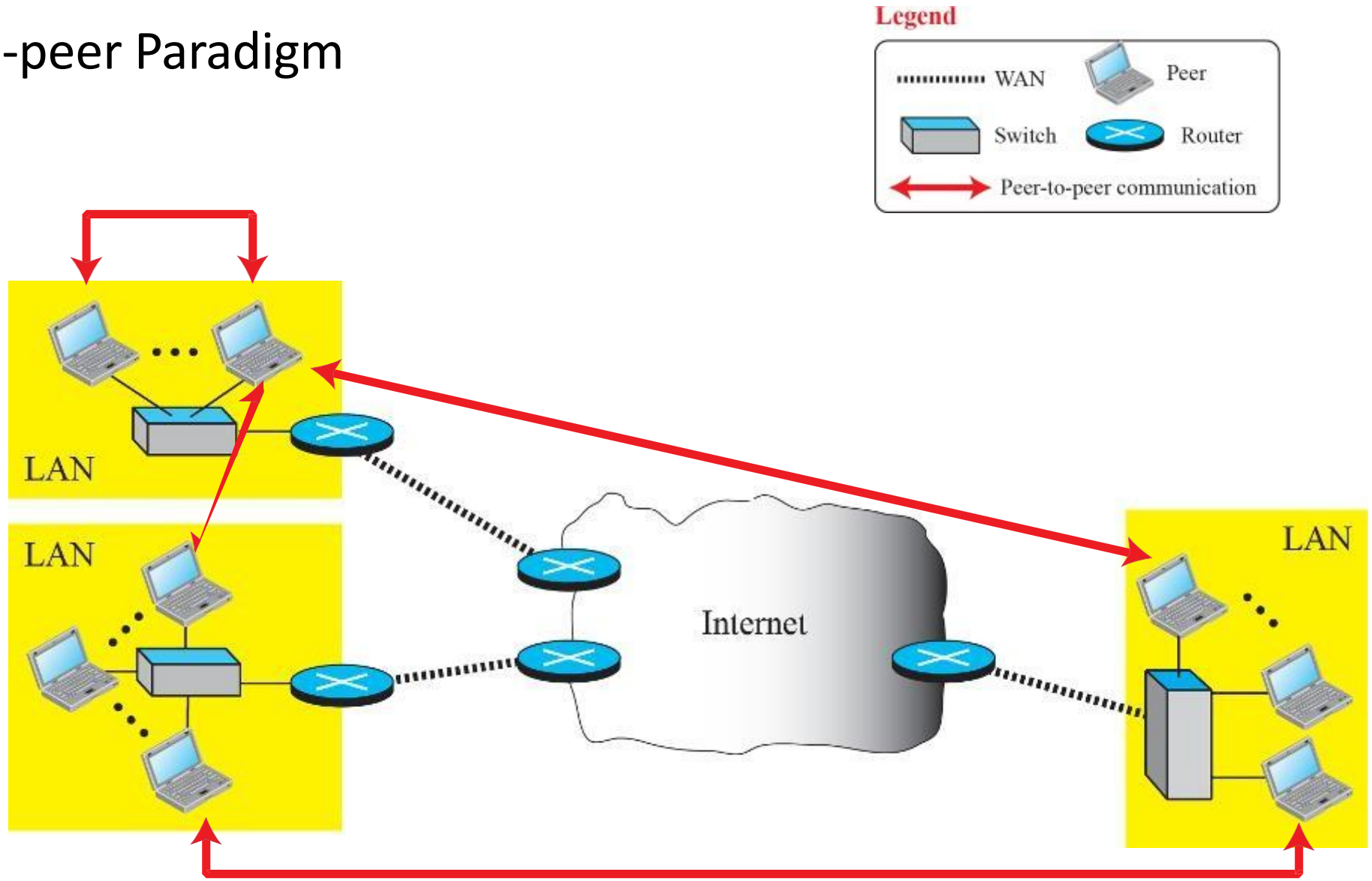
Clients are prone to viruses, Trojans and worms if present in the Server or uploaded into the Server.

Server are prone to Denial of Service (DOS) attacks.

Data packets may be spoofed or modified during transmission.

Phishing or capturing login credentials or other useful information of the user are common and MITM(Man in the Middle) attacks are common.

Peer-to-peer Paradigm



Peer-to-Peer Architecture

A peer-to-peer network is a simple network of computers.

It first came into existence in the late 1970s.

Here each computer acts as a node for file sharing within the formed network.

Here each node acts as a server and thus there is no central server in the network.

This allows the sharing of a huge amount of data.

The tasks are equally divided amongst the nodes.

Each node connected in the network shares an equal workload.

For the network to stop working, all the nodes need to individually stop working. This is because each node works independently.

eg - Bit-torrent, Skype

Types of P2P networks

Unstructured P2P networks: In this type of P2P network, each device is able to make an equal contribution.

This network is easy to build as devices can be connected randomly in the network.

For example, Napster, Gnutella, etc.

Structured P2P networks: It is designed using software that creates a virtual layer in order to put the nodes in a specific structure.

These are not easy to set up but can give easy access to users to the content.

For example, P-Grid, Kademlia, etc.

Hybrid P2P networks: It combines the features of both P2P networks and client-server architecture.

Features of P2P network

These networks do not involve a large number of nodes, usually less than 12.

All the computers in the network store their own data but this data is accessible by the group.

Unlike client-server networks, P2P uses resources and also provides them. This results in additional resources if the number of nodes increases. It requires specialized software. It allows resource sharing among the network.

Since the nodes act as clients and servers, there is a constant threat of attack.

Almost all OS today support P2P networks.

Advantages of P2P Network

Easy to maintain: The network is easy to maintain because each node is independent of the other.

Less costly: Since each node acts as a server, therefore the cost of the central server is saved. Thus, there is no need to buy an expensive server.

No network manager: In a P2P network since each node manages his or her own computer, thus there is no need for a network manager.

Adding nodes is easy: Adding, deleting, and repairing nodes in this network is easy.

Less network traffic: In a P2P network, there is less network traffic than in a client/ server network.

Disadvantages of P2P Network

Data is vulnerable: Because of no central server, data is always vulnerable to getting lost because of no backup.

Less secure: It becomes difficult to secure the complete network because each node is independent.

Slow performance: In a P2P network, each computer is accessed by other computers in the network which slows down the performance of the user.

Files hard to locate: In a P2P network, the files are not centrally stored, rather they are stored on individual computers which makes it difficult to locate the files.

Electronic mail

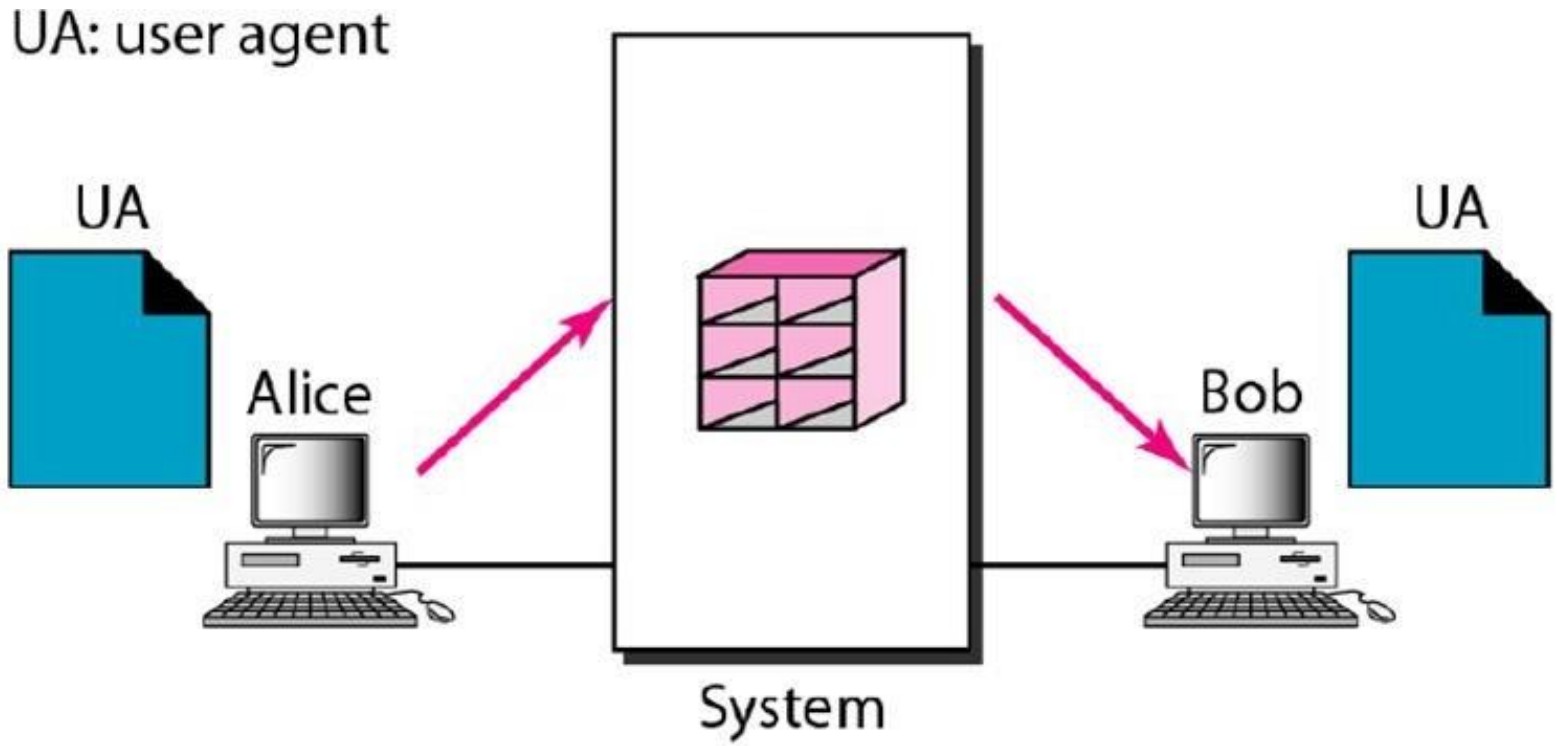
At the beginning of the Internet era, the messages sent by electronic mail were short and consisted of text only. Today, electronic mail is much more complex. It allows a message to include text, audio, and video. It also allows one message to be sent to one or more recipients.

Architecture

To explain the architecture of e-mail, we give four scenarios. We begin with the simplest situation and add complexity as we proceed. The fourth scenario is the most common in the exchange of email.

First Scenario

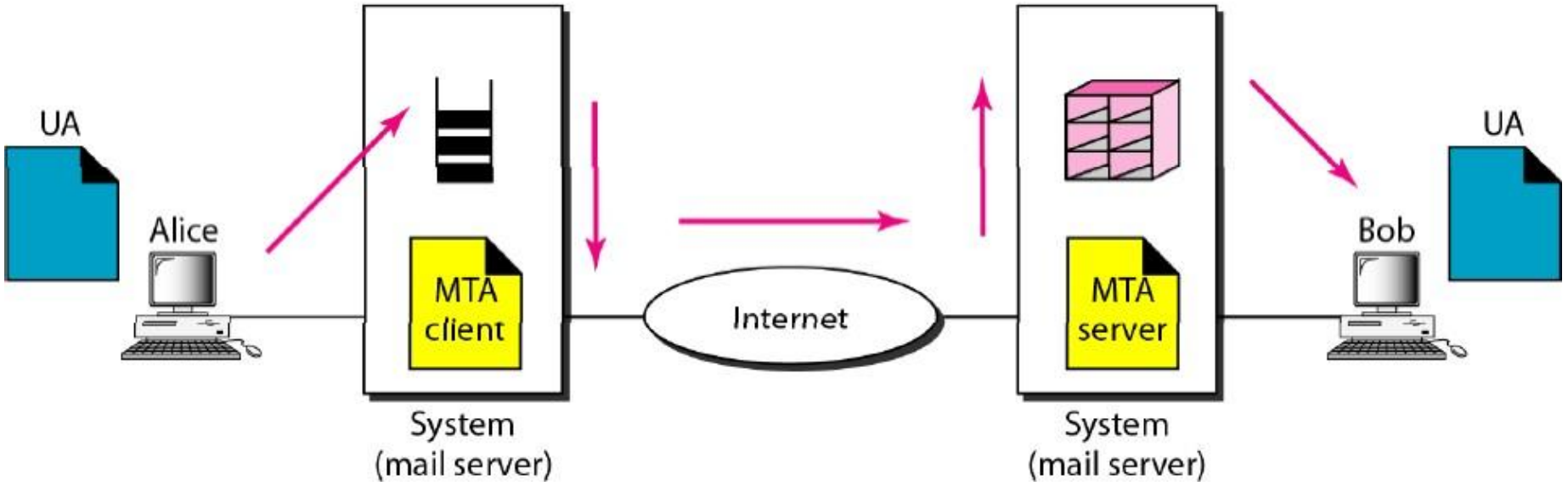
- In the first scenario, the sender and the receiver of the e-mail are users (or application programs) on the **same system;** they are directly connected to a shared system.
- The administrator has created one **mailbox** for each user where the received messages are stored. A *mailbox* is part of a **local hard drive**, a special file with permission restrictions.
- Only the owner of the mailbox has **access** to it.
- When Alice, a user, needs to send a message to Bob, another user, Alice runs a **user agent (UA)** program to prepare the message and **UA itself** will store it in Bob's mailbox.
- The message has the sender and recipient mailbox addresses (names of files).
- Bob can retrieve and **read the contents** of his mailbox at his convenience, **using a user agent**. Figure below shows the concept.


UA: user agent
UA
Alice
System
Bob
UA

Second Scenario

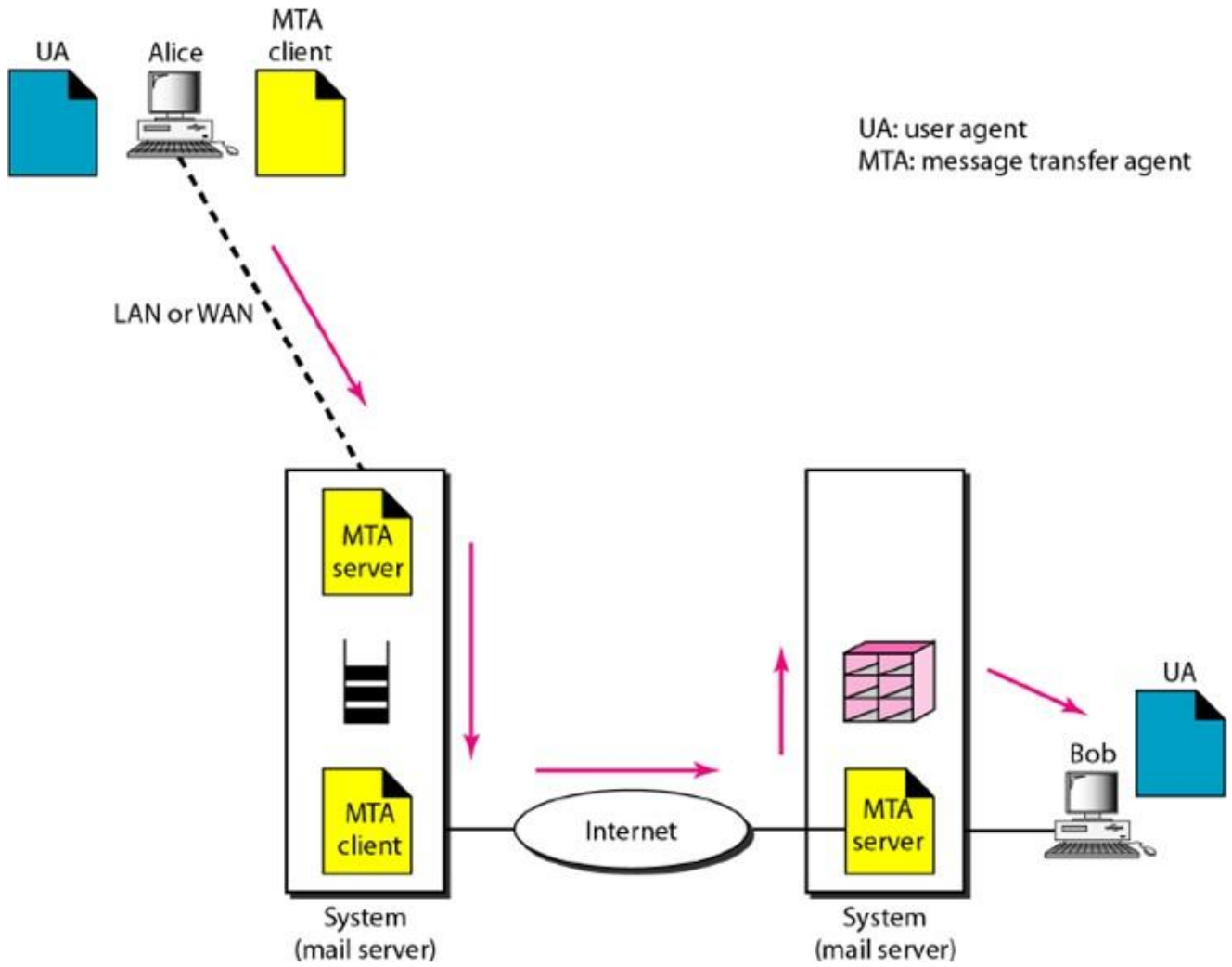
- In the second scenario, the sender and the receiver of the e-mail are users (or application programs) on **two different systems**.
- So the message needs to be sent over the Internet.
- Here we need **user agents (UAs) and message transferagents (MTAs),** as shown in figure below.
- Alice needs to use a **user agent** program to **send** her message to the system **at her own site**.
- The computer (sometimes called the **mail server**) at her site manages all the messages received and makes a **mail queue** to store messages waiting to be sent.
- MTA client keeps on checking the queue and when there is message then it will try to establish connection with MTA server and when connection is established then it pick the messages from queue and send them over internet to MTA server which is running at system from which bob is connected.
- MTA server after receiving messages will store them at mail box of bob.
- Like most client/server programs on the Internet, the server needs to run all the time because it does not know when a client will ask for a connection.
- Bob also needs a **user agent** program to retrieve messages stored in the mailbox of the system at his site.

UA: user agent
MTA: message transfer agent



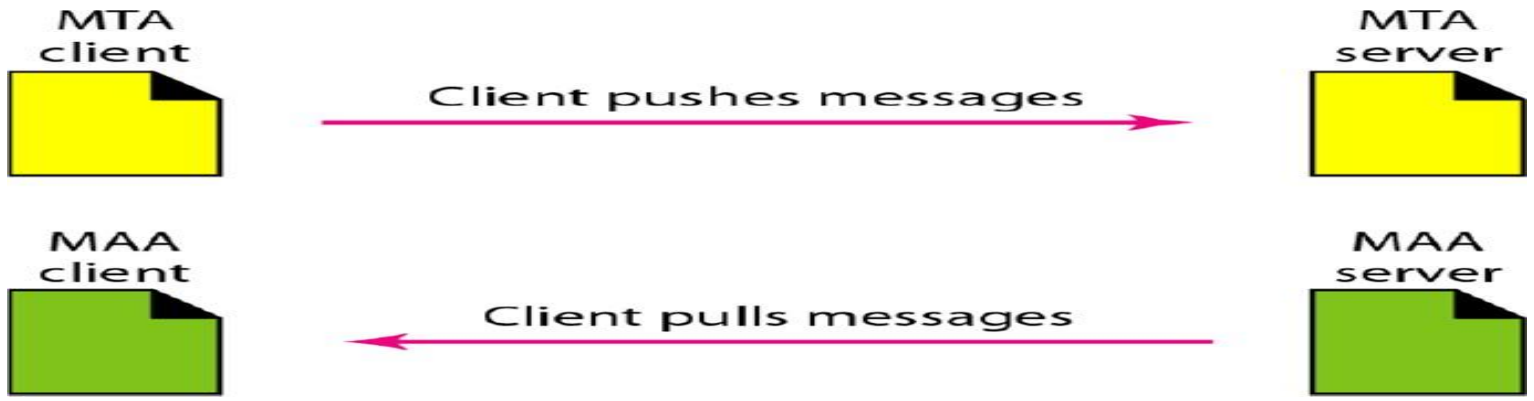
Third Scenario

- In the third scenario, Bob, as in the second scenario, is directly connected to his system.
- Alice, however, is separated from her system.
- Either Alice is connected to the system via a point-to-point WAN, such as a dial-up modem, a DSL, or a cable modem; or she is connected to a LAN in an organization that uses one mail server for handling e-mails-all users need to send their messages to this mail server. Figure below shows the situation.
- Alice still needs a user agent to prepare her message. She then needs to send the message through the LAN or WAN. This can be done through a pair of message transfer agents (client and server).
- Whenever Alice has a message to send, she calls the user agent which, in tum, calls the MTA client. The MTA client establishes a connection with the MTA server on the system, which is running all the time.
- The mail server computer at Alice's site manages all messages received and makes a mail queue. MTA client keeps on checking this queue and whenever there are messages to send it establishes a connection with MTA server at Bob's site and sends the messages. This server receives the messages and stores it in Bob's mailbox.
- At his convenience, Bob uses his user agent to retrieve the message and reads it.

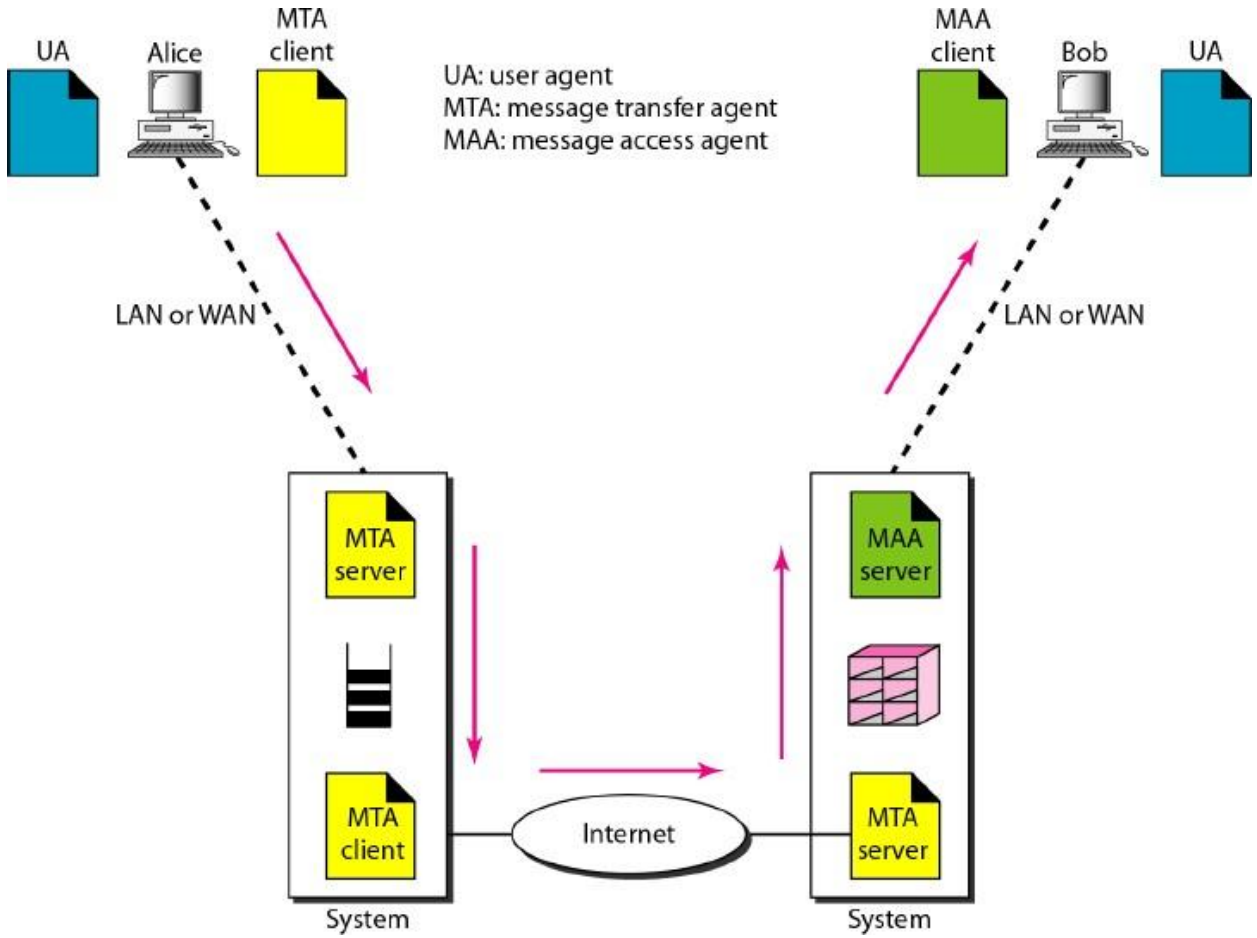

UA
Alice
MTA
client
UA: user agent
MTA: message transfer agent
LAN or WAN
MTA
server
MTA
client
Internet
MTA
server
UA
Bob
System
(mail server)
System
(mail server)

Fourth Scenario

- In the fourth and **most common scenario**, Bob is also connected to his mail server by a WAN or a LAN. After the message has arrived at Bob's mail server, Bob needs to retrieve it.
- We can’t uses of MTA client at Bob computer MTA server at mail server computer. This is because MTA client is **push program** which pushes the messages to MTA server, it can’t pull messages from MTA server.
- Other option is use MTA sever at Bob computer which is impossible as now Bob can’t shut down her computer because she doesn’t know when message will arrive.
- Hence, we need another set of client/server agents, which we call **message access agents** (MAAs).
- Unlike MTA client, MAA client is a **pull program** and it can pull messages from MAA server.



- So Bob uses an MAA client to retrieve his messages. The client sends a request to the MAA server, which is running all the time, and requests the transfer of the messages. Figure on next page shows the situation.

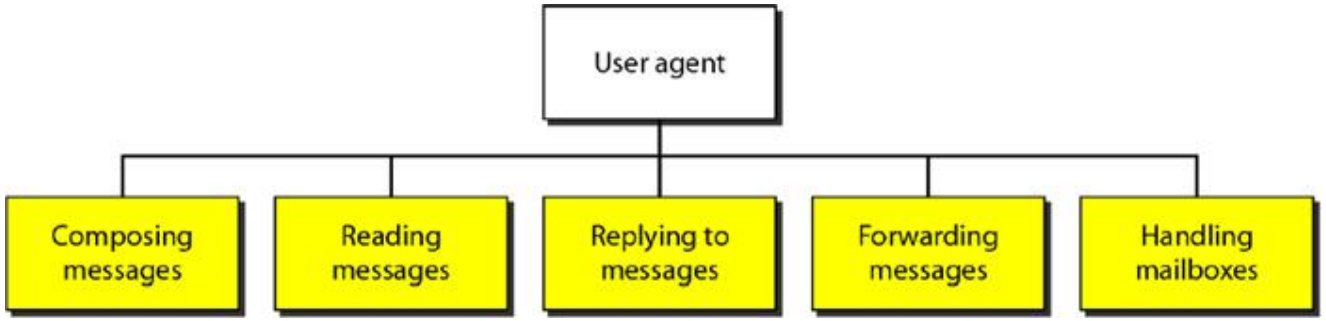

UA
Alice
MTA client
UA: user agent
MTA: message transfer agent
MAA: message access agent
MAA client
Bob
UA
LAN or WAN
LAN or WAN
MTA server
MAA server
MTA client
Internet
MTA server
System
System

USER AGENT

The first component of an electronic mail system is the user agent (UA). It provide service to the user to make the process of sending and receiving a message easier. Best known example is Microsoft outlook.

Services Provided by a User Agent

A user agent is a software package (program) that composes, reads, replies to, and forwards messages. It also handles mailboxes. Figure below shows the services of a typical user agent.



User Agent Types

There are two types of user agents

1. **Command-Driven**

•Command-driven user agents belong to the early days of electronic mail. They are still present as the underlying user agents in servers.

•A command-driven user agent normally accepts a one-character command from the keyboard to perform its task. For example, 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.

2. **GUI-Based**

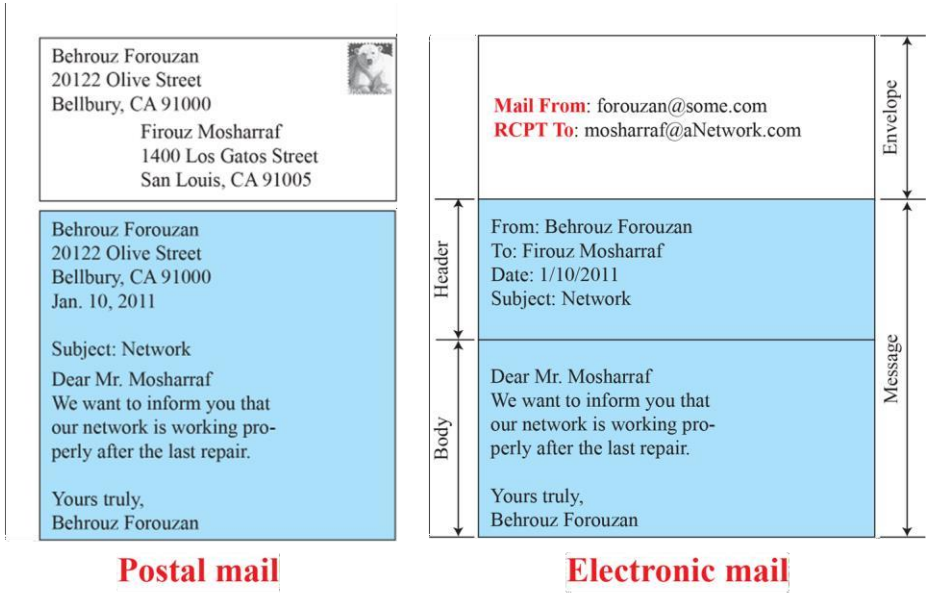
- Modem user agents are GUI-based. They contain graphical-user interface (GUI) components that allow the user to interact with the software by using both the keyboard and the mouse.
- They have graphical components such as icons, menu bars, and windows that make the services easy to access.
- Some examples of GUI-based user agents are Eudora, Microsoft's Outlook, and Netscape.

Sending mail

To send an email, you use a computer program called a User Agent (UA). It's like writing a letter. The email has two parts: an envelope and a message. The envelope holds the sender's address, the receiver's address, and more. The message has a header and a body. The header says who sent it, who gets it, the subject, and more. The body has the main information that the receiver reads.

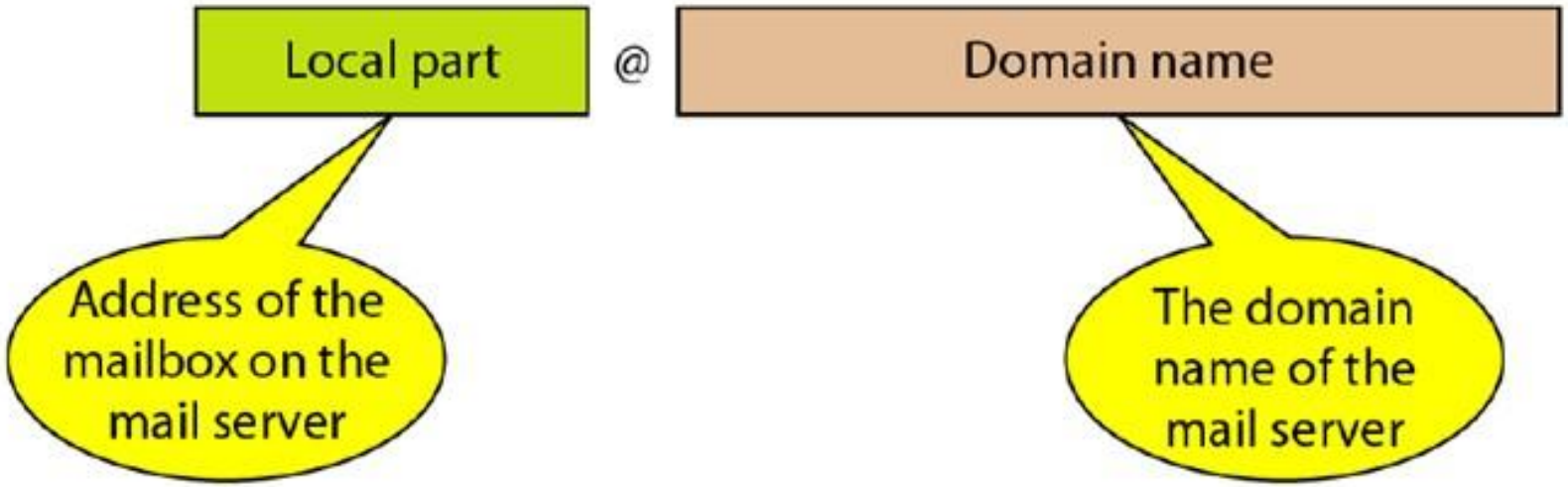
Receiving mail

The user's email program (User Agent or UA) starts when the user wants or at a set time. If there's new email, the UA tells the user. When the user wants to read email, a list appears. Each line in the list shows a quick view of a message: who sent it, what it's about, and when. The user can pick a message to read its full content on the screen.



Addresses

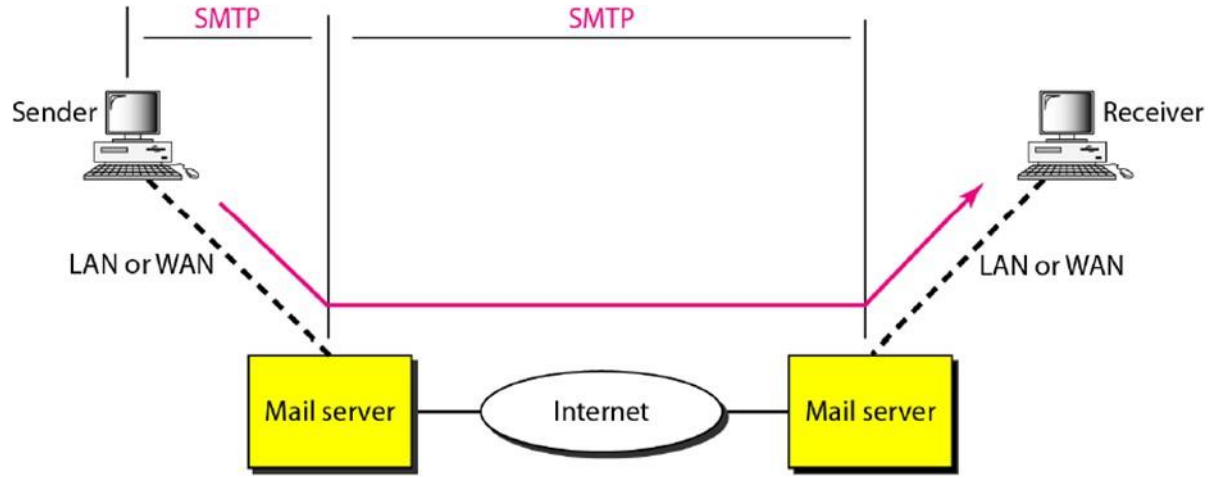
To deliver mail, a mail handling system must use an addressing system with unique addresses. In the Internet, the address consists of two parts: a local part and a domain name, separated by an @ sign (see Figure below).



Message Transfer Agent : SMTP

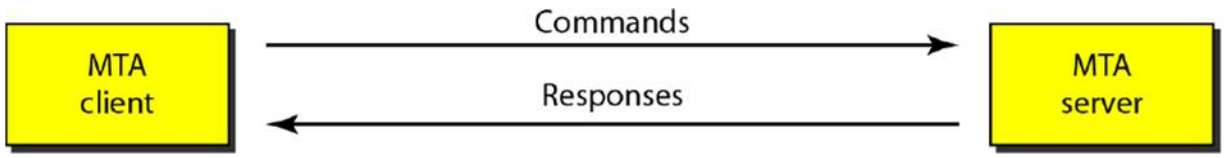
•The actual mail transfer is done through message transfer agents. To send mail, a system must have the client MTA, and to receive mail, a system must have a server MTA.

•The **formal protocol that defines the MTA client and server in the Internet is called the Simple Mail Transfer Protocol (SMTP)**. As we said before, two pairs of MTA client/server programs are used in the most common situation (fourth scenario). Figure below shows the range of the SMTP protocol in this scenario.



- SMTP is used two times, between the sender and the sender's mail server and between the two mail servers.
- It is a connection-oriented protocol and uses **TCP with port number 25**.

•SMTP simply defines how commands and responses must be sent back and forth. MTA client sends the command and server sends the responses (see Figure below).



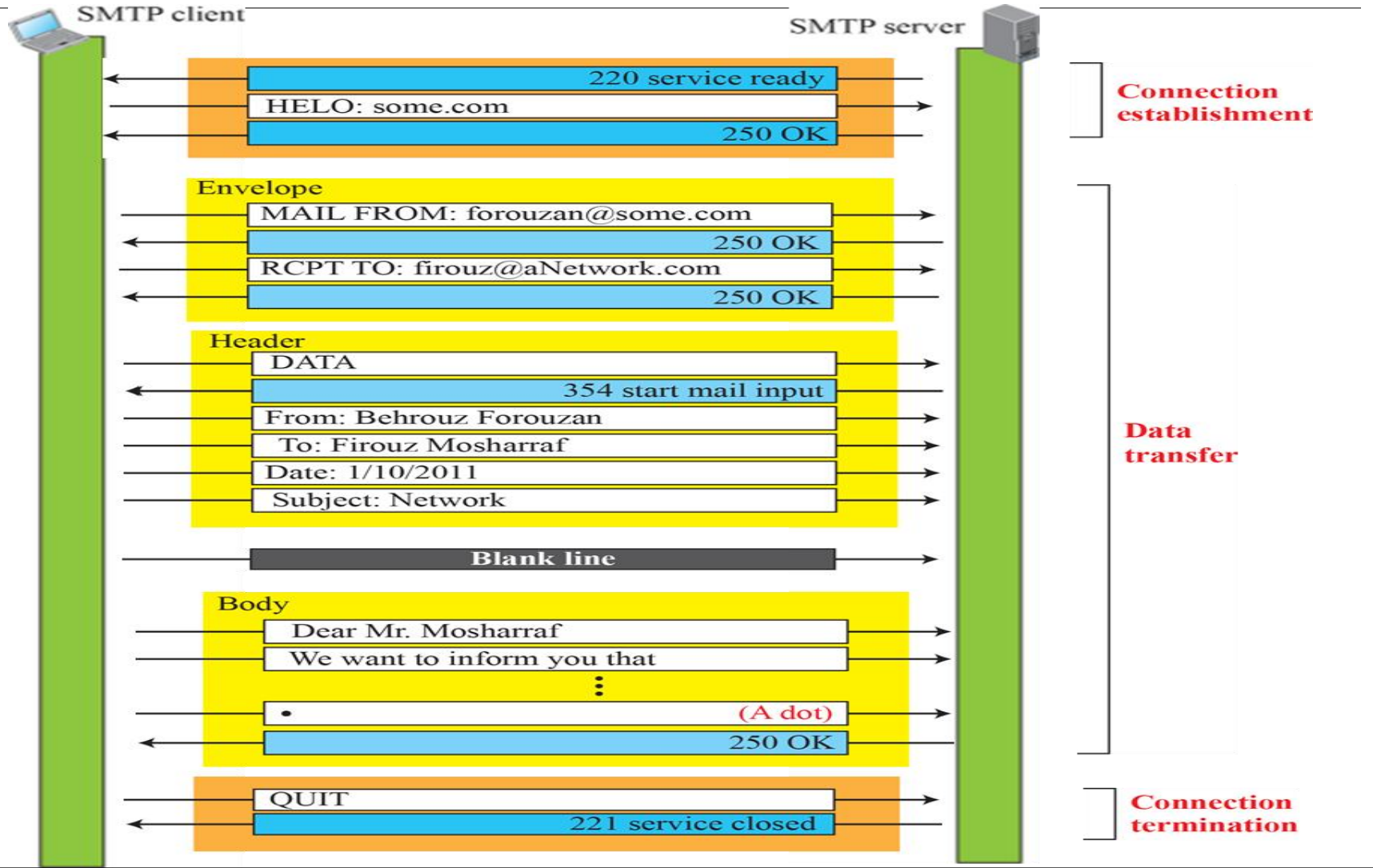
SMTP COMMANDS

HELO. This command is used by the client to identify itself.
MAIL FROM. This command is used by the client to identify the sender of the message.
RCPT TO. This command is used by the client to identify the intended recipient of the message.
DATA. This command is used to send the actual message.
QUIT. This command terminates the message.
RSET. This command aborts the current mail transaction.
VRFY. This command is used to verify the address of the recipient.
NOOP. This command is used by the client to check the status of the recipient
TURN. This command lets the sender and the recipient switch positions, whereby the sender becomes the recipient and vice versa
EXPN. This command can verify the existence of one or more mailboxes on the system.
HELP. This command asks the recipient to send information about the command sent as the argument.
SEND FROM. This command specifies that the mail is to be delivered to the terminal of the recipient, and not the mailbox.
SMOL FROM. This command specifies that the mail is to be delivered to the terminal or the mailbox of the recipient.
SMAL FROM. This command specifies that the mail is to be delivered to the terminal and the mailbox of the recipient.

SMTP 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

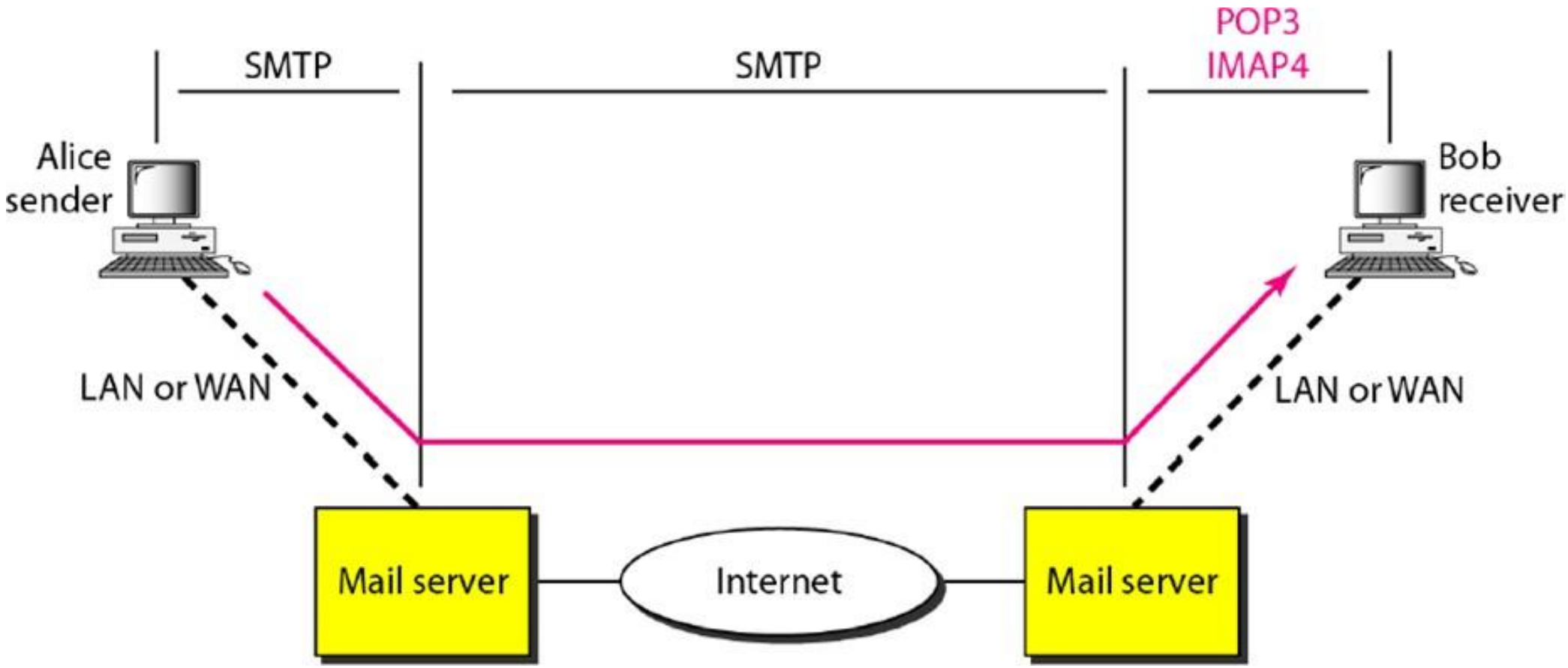
To show the three mail transfer phases, we show all of the steps described above using the information depicted in Figure 2.23. In the figure, we have separated the messages related to the envelope, header, and body in the data transfer section. Note that the steps in this figure are repeated two times in each e-mail transfer: once from the e-mail sender to the local mail server and once from the local mail server to the remote mail server. The local mail server, after receiving the whole e-mail message, may spool it and send it to the remote mail server at another time.



Message Access Agent: POP and IMAP

•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. In other words, the direction of the bulk: data (messages) is 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. 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). Figure below shows the position of these two protocols in the most common situation (fourth scenario).


SMTP
SMTP
POP3
IMAP4
Alice
sender
Bob
receiver
LAN or WAN
LAN or WAN
Mail server
Internet
Mail server

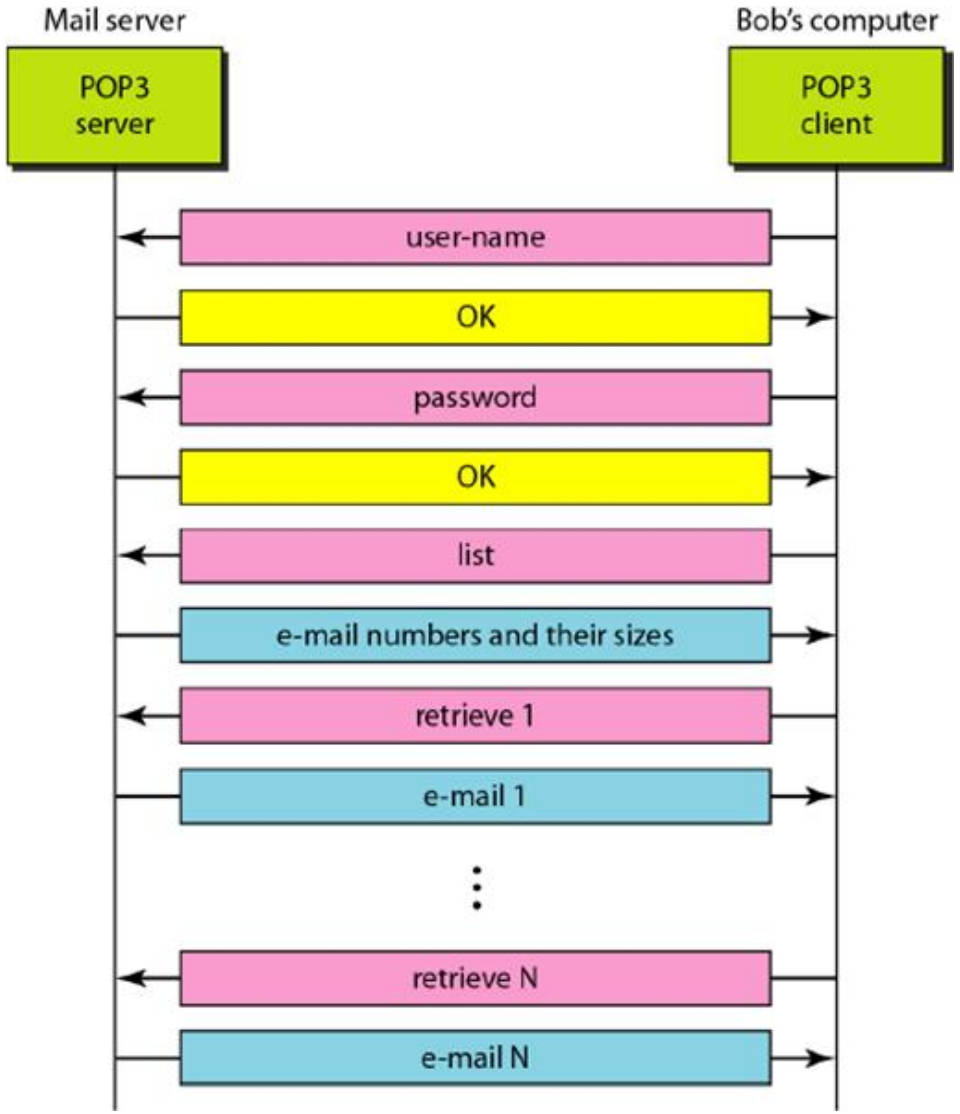
POP3

•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 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 below shows an example of downloading using POP3.


Mail server
Bob's computer
POP3 server
POP3 client
user-name
OK
password
OK
list
e-mail numbers and their sizes
retrieve 1
e-mail 1
retrieve N
e-mail N

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

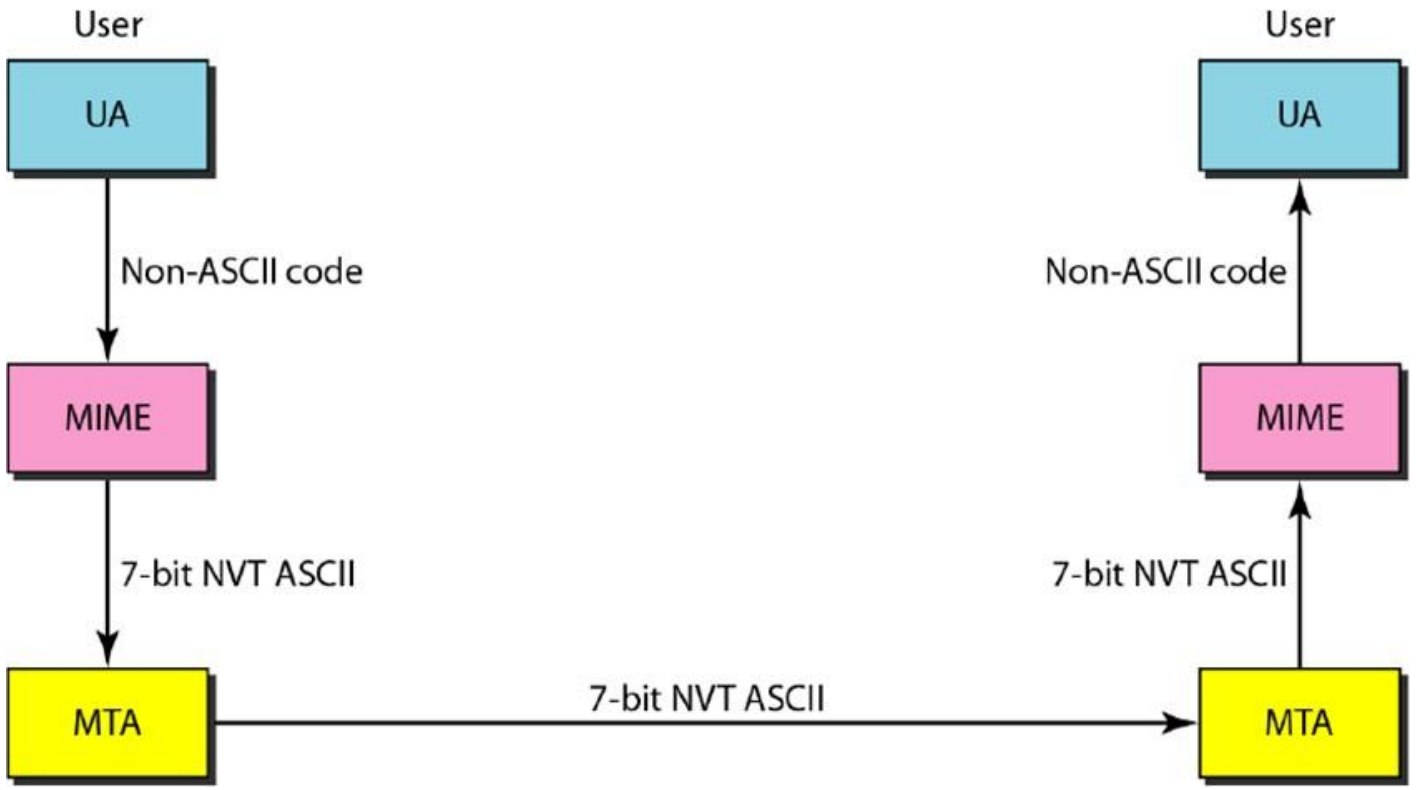
IMAP4

IMAP4 uses TCP port number 143 and **provides the following extra functions which are not provided by POP3**

- 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.

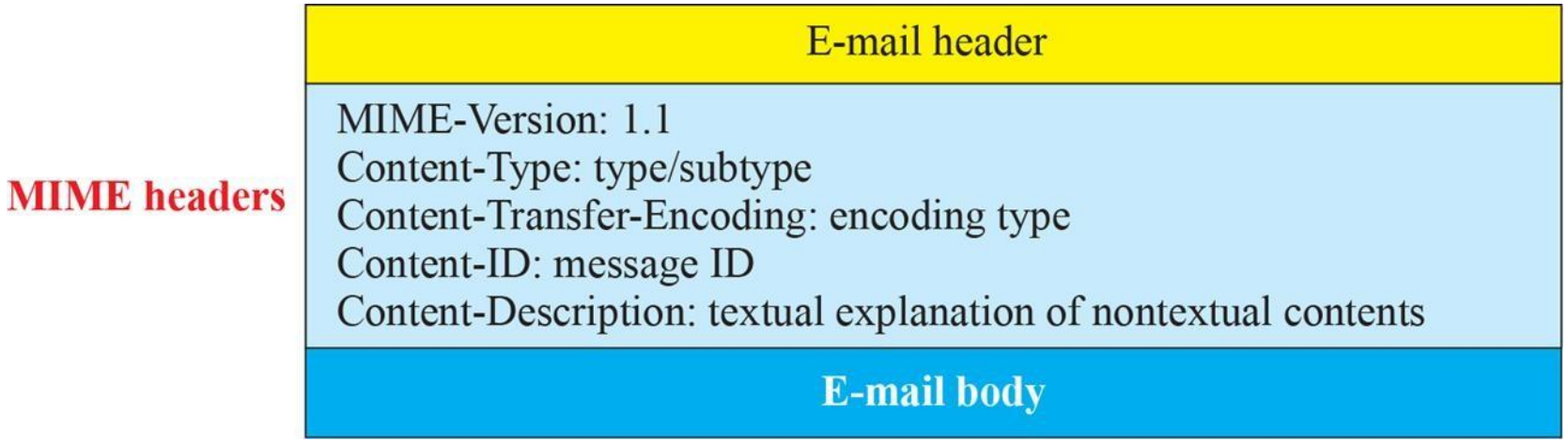
MIME(multipurpose internet mail extension)

- Electronic mail has a simple structure. Its simplicity, however, comes at a price. It can **send messages only in 7-bit ASCII** format.
- In other words, it has some limitations. For example, it cannot be used for languages that are not supported by 7-bit ASCII characters (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 transforms non-ASCII data at the sender site to ASCII data and delivers them to the client MTA to be sent through the Internet. The message at the receiving side is transformed back to the original data.
- We can think of MIME as a set of software functions that transforms non-ASCII data (stream of bits) to ASCII data and vice versa, as shown in Figure below
- MIME allows seven different types of data like audio, video, image etc.

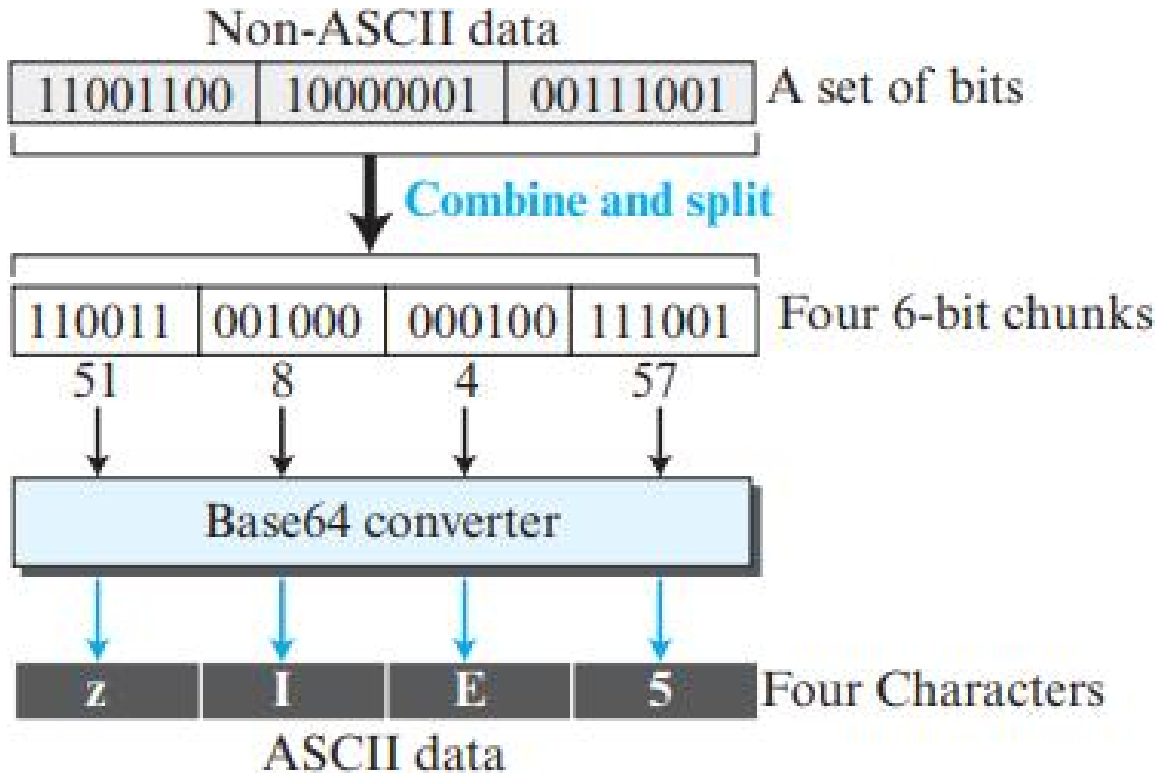

User
UA
Non-ASCII code
MIME
7-bit NVT ASCII
MTA
7-bit NVT ASCII
User
UA
Non-ASCII code
MIME
7-bit NVT ASCII
MTA

MIME Headers

MIME defines five headers, as shown in Figure, which can be added to the original e-mail header section to define the transformation para



Base64 conversion for MINE to ASCII data



HTTP

HyperText Transfer Protocol (HTTP)

- HTTP defines client-server communication for retrieving web pages.
- Client sends requests; server returns responses.
- Server uses port 80; client uses temporary port.
- HTTP relies on TCP for connection-oriented and reliable communication.
- TCP ensures message reliability and manages errors.
- Connection establishment and termination are essential.
- No need to worry about message errors or loss due to TCP's reliability.

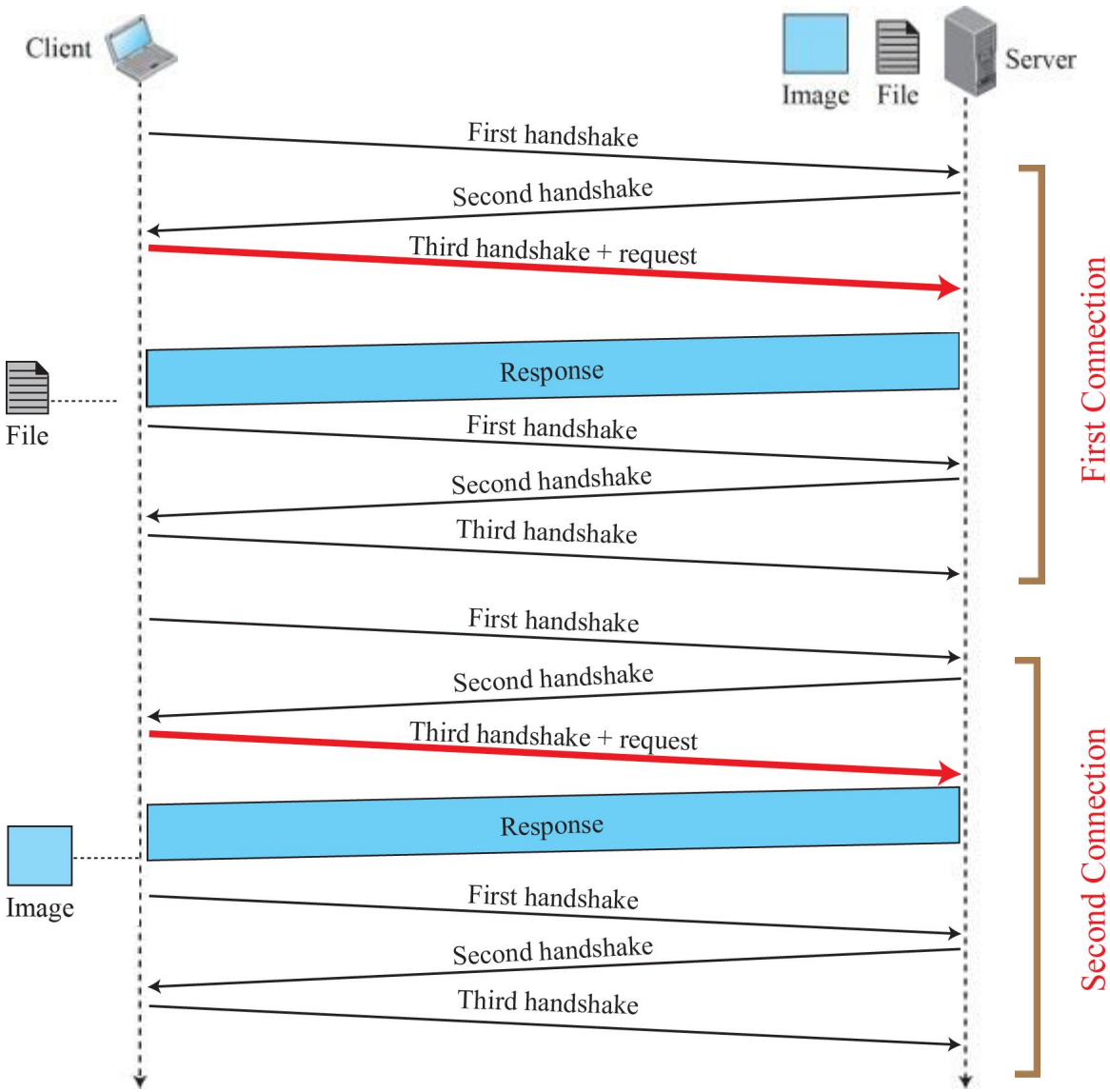
In web pages, multiple requests and responses are often needed due to embedded hypertext. When objects are on different servers, we create new TCP connections for each. If objects are on the same server, we can either use separate connections for each or one connection for all. Separate connections are 'nonpersistent,' while using one connection is 'persistent.' HTTP used to default to nonpersistent (pre-1.1), but version 1.1 made persistent default, user-changeable.

Nonpersistent Connections

In a nonpersistent connection, each request/response uses a new TCP connection:

1. Client opens a connection, sends request.
2. Server sends response, closes connection.
3. Client reads data, closes connection.

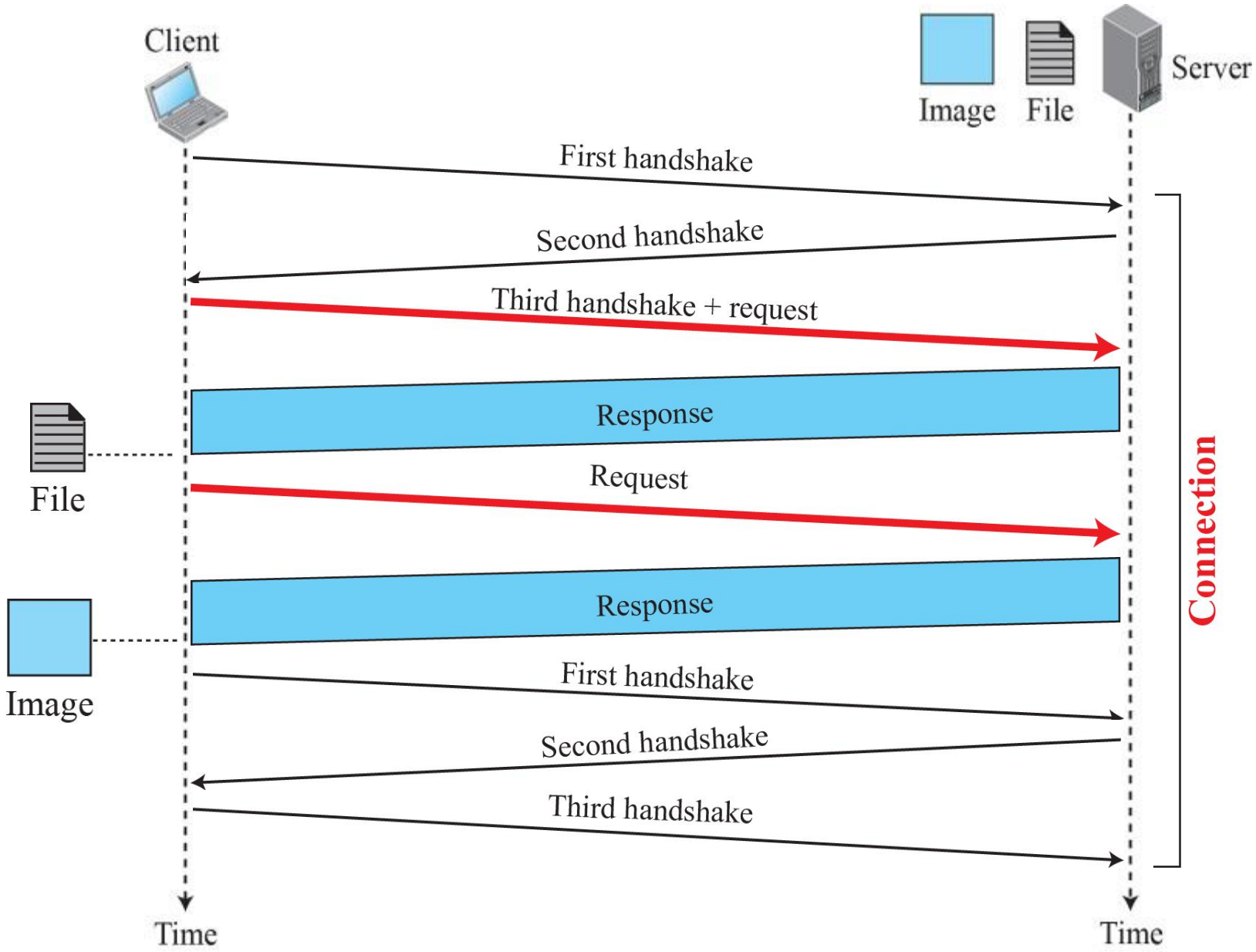
If a file has N picture links on the same server, N + 1 connections open and close. This method causes high server overhead due to N + 1 buffers needed for each connection.


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

Persistent Connection

HTTP 1.1 defaults to persistent connections:

- Server **keeps connection open for more requests after responding.**
- **Connection closed** if client requests or times out.
- Sender usually provides **data length** with each response.
- **Dynamic content** might not have known length. Server tells client if length is unknown, closes after sending.
- Persistent connections **save time, resources, and round trips**.
- **One set of buffers and variables needed** per connection.
- Connection setup and termination time saved.

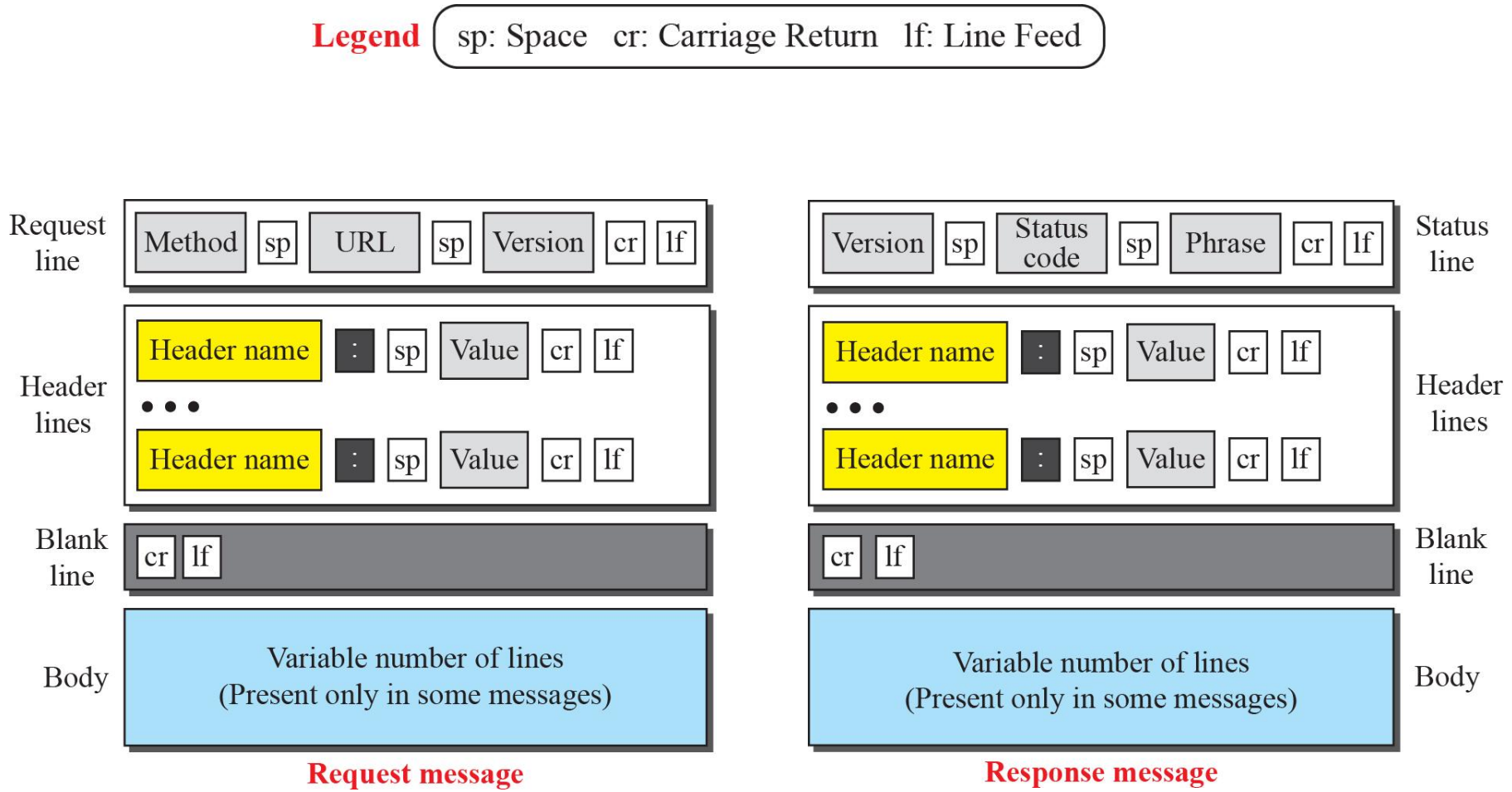

Client
Image
File
Server
First handshake
Second handshake
Third handshake + request
Response
File
Request
Response
Image
First handshake
Second handshake
Third handshake
Connection
Time
Time

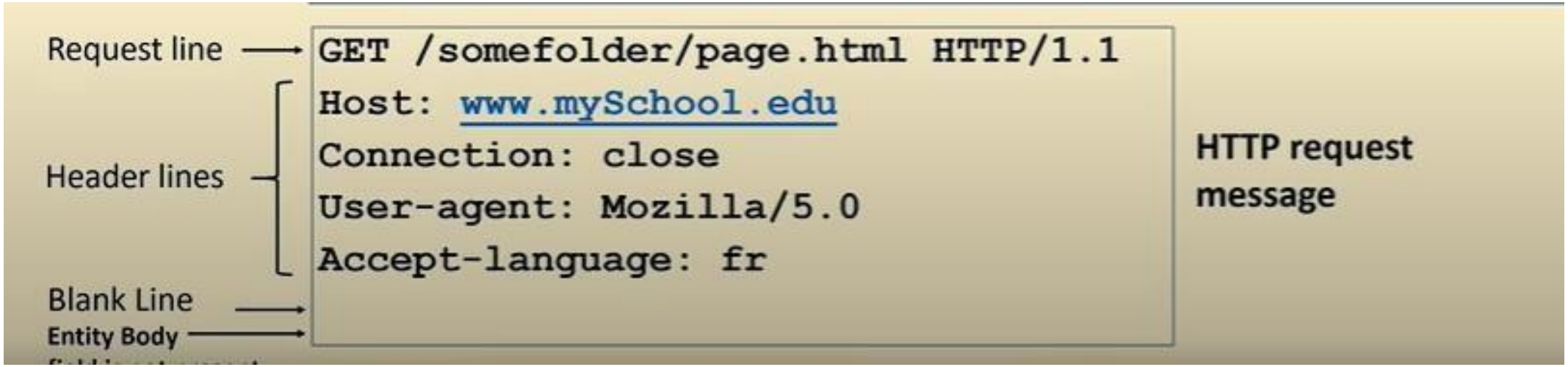
Message Formats

The first section in the request message is called the request line; the first section in the response message is called the status line. The other three sections have the same names in the request and response messages. However, the similarities between these sections are only in the names; they may have different contents.

Request message

Response message


Legend
sp: Space cr: Carriage Return lf: Line Feed
Request line
Method
sp
URL
sp
Version
cr
lf
Header lines
Header name
:
sp
Value
cr
lf
Header name
:
sp
Value
cr
lf
Blank line
cr
lf
Body
Variable number of lines
(Present only in some messages)
Request message
Version
sp
Status code
sp
Phrase
cr
lf
Status line
Header name
:
sp
Value
cr
lf
Header name
:
sp
Value
cr
lf
Header lines
cr
lf
Blank line
Variable number of lines
(Present only in some messages)
Body
Response message


Request line
GET /somefolder/page.html HTTP/1.1
Header lines
Host: www.mySchool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
Blank Line
Entity Body
HTTP request message

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

GET: this method is used when browser request an object with the requested object identified in URL
POST method is used when a browser send data in entity body field.

unlike GET method user-provided search term (or data does) does not concatenate with url rather it goes in the entity-body field

usually, web-based form's data go via POST method

HEAD method is like a GET method. But, when a server receives a request with HEAD method, it responds with HTTP message but it leaves out the requested object.

PUT method allows a user to upload an object to a specific path (or folder) on a web server.

DELETE method allows to delete an object on a web server.

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

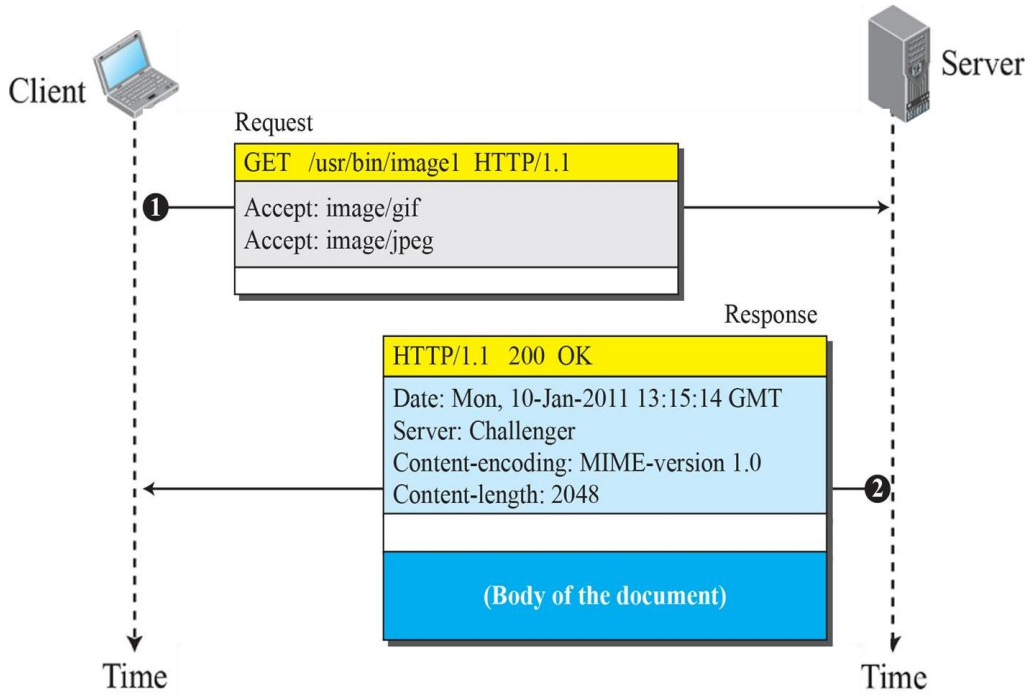
Response message example

Status Codes and Phrases

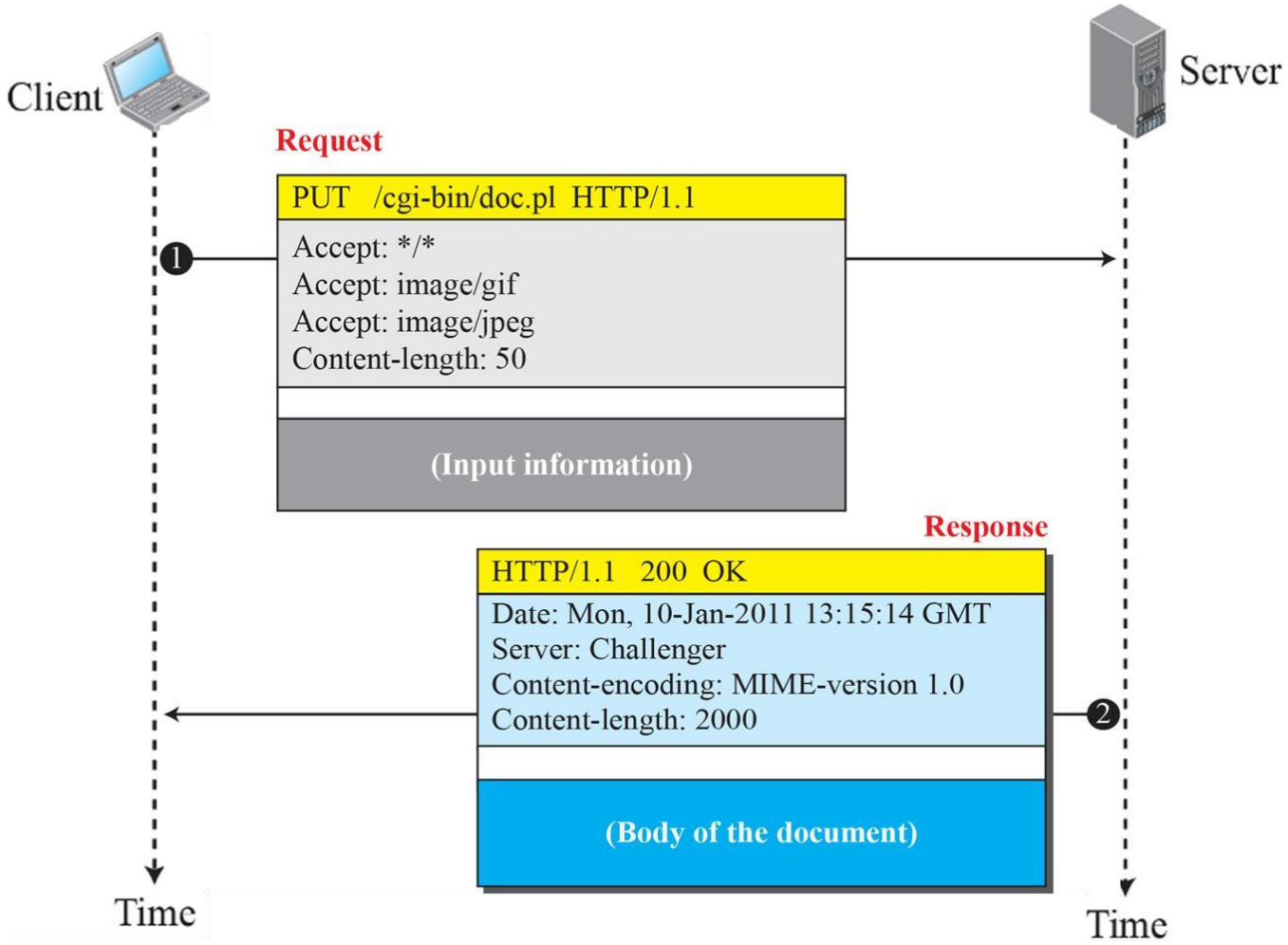
Status Code	Phrase	Description
200	OK	Request succeeded and the information is provided in the response message.
301	Moved Permanently	Requested object has been permanently moved. The new URL is given in `Location:` header field of the response message. The client software automatically retrieves the new URL.
400	Bad Request	Request could not be understood by the server.
404	Not Found	Requested object does not exist on the server.
505	HTTP Version Not Supported	Requested HTTP version is not supported by the server.

Example1- This example retrieves a document (see Figure 2.13). We use the GET method to retrieve an image with the path /usr/bin/image1. The request line shows the method (GET), the URL, and the HTTP version (1.1). The header has two lines that show that the client can accept images in the GIF or JPEG format. The request does not have a body. The response message contains the status line and four lines of header. The header lines define the date, server, content encoding and length of the document. The body of the document follows the header.

Figure 2.13: Example 2.6



Example 2- In this example, the client wants to send a web page to be posted on the server. We use the PUT method. The request line shows the method (PUT), URL, and HTTP version (1.1). There are four lines of headers. The request body contains the web page to be posted. The response message contains the status line and four lines of headers. The created document, which is a CGI document, is included as the body (see Figure 2.14).



Conditional Request

A client can add a condition in its request. In this case, the server will send the requested web page if the condition is met or inform the client otherwise. **One of the most common conditions imposed by the client is the time and date the web page is modified**. The client can send the header line If-Modified-Since with the request to tell the server that it **needs the page only if it is modified after a certain point in time.**

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

GET http://www.commonServer.com/information/file1 HTTP/1.1	Request line
If-Modified-Since: Thu, Sept 04 00:00:00 GMT	Header line
	Blank line

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

HTTP/1.1 304 Not Modified	Status line
Date: Sat, Sept 06 08 16:22:46 GMT	First header line
Server: commonServer.com	Second header line
	Blank line
(Empty Body)	Empty body

Cookies

The World Wide Web was originally designed as a **stateless entity**. A client sends a request; a server responds. Their relationship is over. The original purpose of the Web, retrieving publicly available documents, exactly fits this design. Today the Web has other functions that need to remember some information about the clients

- Websites are being used as electronic stores that allow users to browse through the store, select wanted items, put them in an electronic cart, and pay at the end with a credit card.
- Some websites need to allow access to registered clients only.
- Some websites are used as portals: the user selects the web pages he wants to see.
- Some websites are just advertising agency.

Creating and Storing Cookies

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

1.When a server receives a request from a client, it stores information about the client in a file or a string. The information may include the domain name of the client, the contents of the cookie (information the server has gathered about the client such as name, registration number, and so on), a timestamp, and other information depending on the implementation.

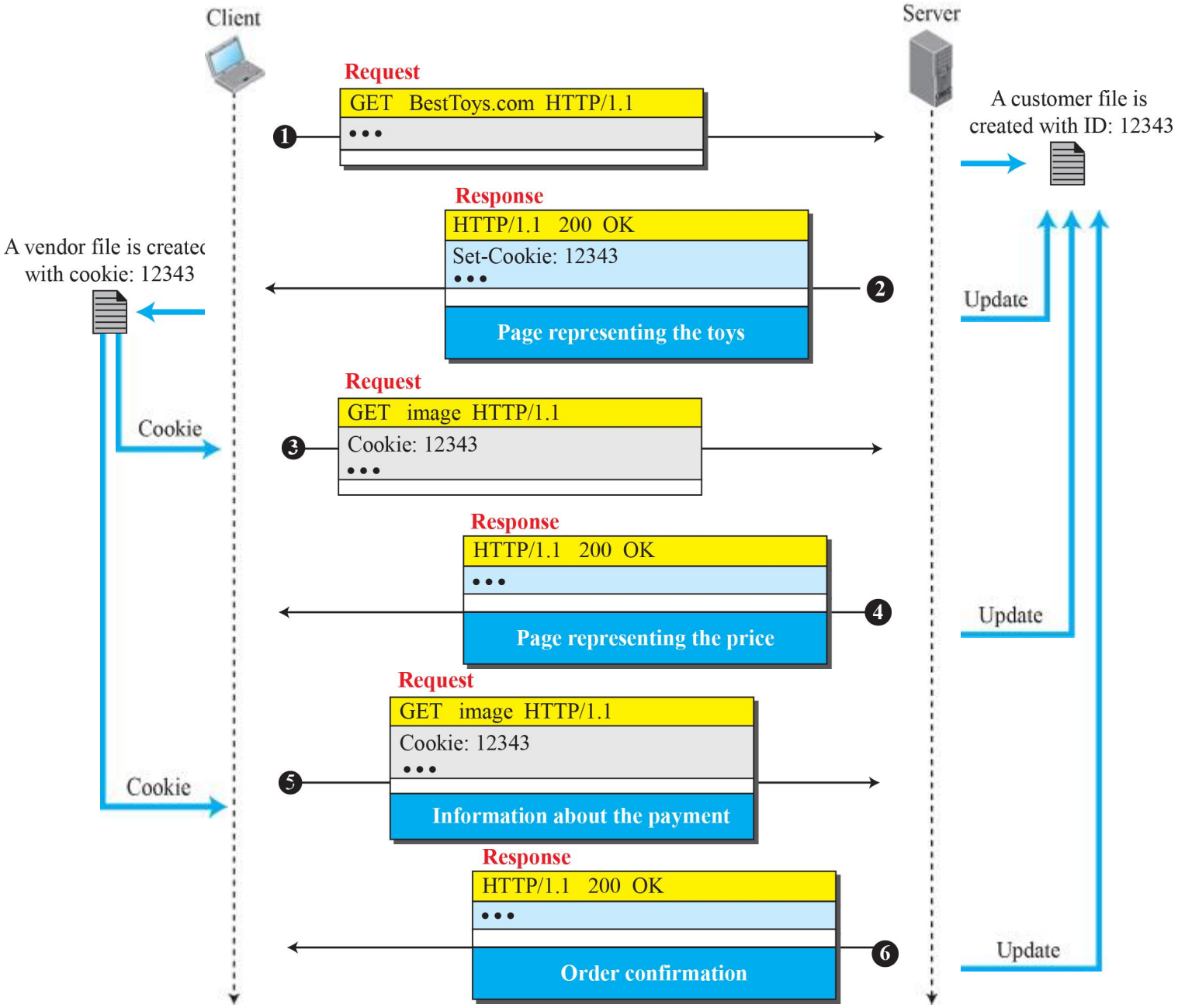
2. The server includes the cookie in the response that it sends to the client.

3.When the client receives the response, the browser stores the cookie in the cookie directory, which is sorted by the server domain name.

Using Cookies

1. **E-commerce Cookie Usage:** An online store uses cookies for shoppers. When a shopper adds an item to their cart, a cookie with item details like number and price is sent. If they add more items, the cookie updates. At checkout, the last cookie calculates the total charge.
2. **Registered Clients Cookie:** Some websites allow only registered users. A first-time registered client gets a cookie. Later visits require the right cookie to access.
3. **Web Portal Cookie:** A web portal uses cookies similarly. A user's favorite pages are saved in a cookie. Returning to the site sends the cookie to show the user's preferences.
4. **Advertising Agency Cookie:** Advertising agencies also use cookies. They place ads on popular sites, sending only a URL. Clicking the ad sends a request to the agency with a cookie containing the user's ID. This builds a profile of the user's web behavior for targeted advertising, which has sparked privacy concerns.

Figure 2.15 shows a scenario in which an electronic store can benefit from the use of cookies. Assume a shopper wants to buy a toy from an electronic store named BestToys. The shopper browser (client) sends a request to the BestToys server. The server creates an empty shopping cart (a list) for the client and assigns an ID to the cart (for example, 12343). The server then sends a response message, which contains the images of all toys available, with a link under each toy that selects the toy if it is being clicked. This response message also includes the Set-Cookie header line whose value is 12343. The client displays the images and stores the cookie value in a file named BestToys.


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

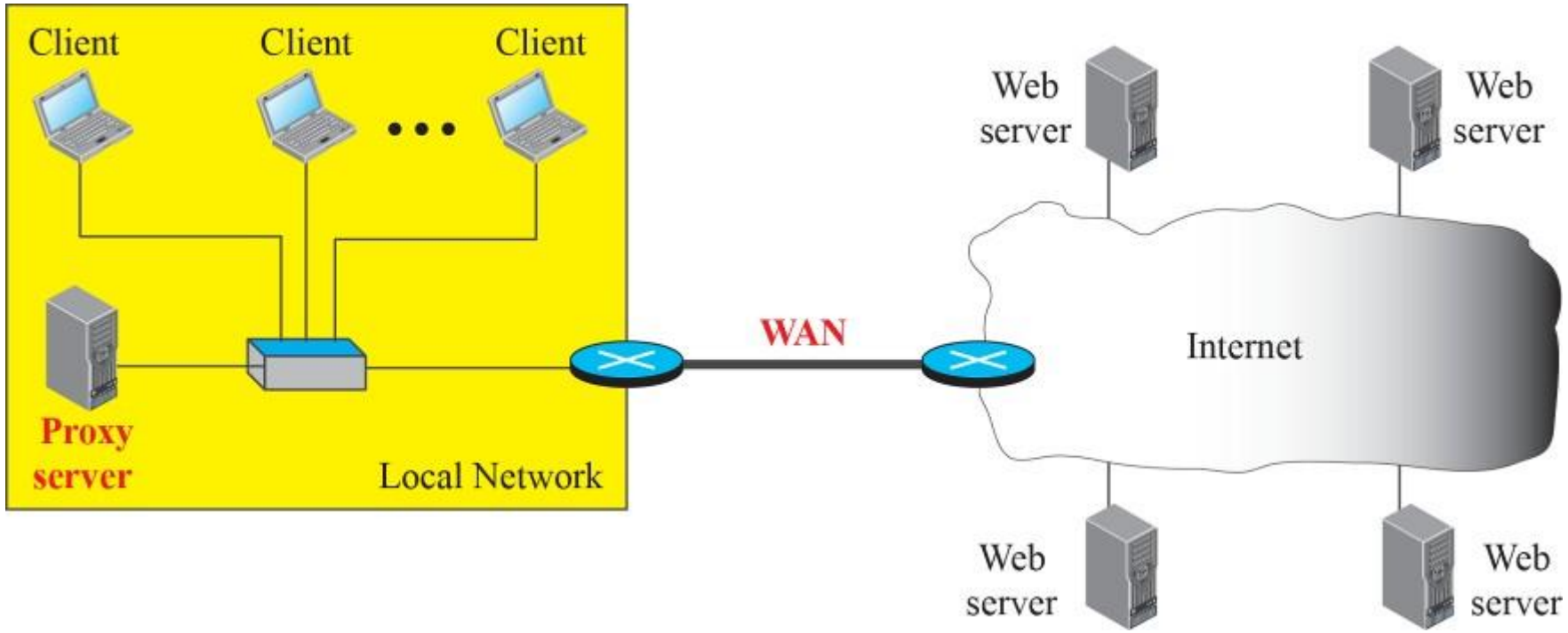
Web Caching: Proxy Server

- HTTP has a feature for proxy servers. These servers store recent response copies. The client sends a request to the proxy server, which checks if the response is cached. If not, it sends the request to the main server. Incoming responses are stored by the proxy server for future requests from other clients.
- The **proxy server lessens the load on the original server, cuts down on traffic, and improves speed**. However, clients must be configured to use the proxy server instead of the main server.
- It's important to know that the proxy server acts as both a server and a client. It **serves responses to clients** when it has them, but it **also acts as a client** to get responses from the main server when it doesn't have them.

Proxy servers are often placed at the client's location, creating a hierarchy:

1. Clients themselves can work as small proxy servers, storing frequently requested responses.
2. Within a company, a proxy server on the LAN reduces incoming and outgoing load.
3. Internet Service Providers (ISPs) with many customers can use proxy servers to ease network load.

Figure 2.16 shows an example of a use of a proxy server in a local network, such as the network on a campus or in a company. The proxy server is installed in the local network. When an HTTP request is created by any of the clients (browsers), the request is first directed to the proxy server. If the proxy server already has the corresponding web page, it sends the response to the client. Otherwise, the proxy server acts as a client and sends the request to the web server in the Internet. When the response is returned, the proxy server makes a copy and stores it in its cache before sending it to the requesting client.



Cashe Update

An important question is how long a response should stay in the proxy server before being removed and replaced. Different strategies are used for this. One idea is to keep a list of websites that don't change often. For example, **a news site might update its news each morning. So, a proxy could get the news in the morning and keep it until the next day**. Another suggestion is to **add headers that show when the information was last changed**. The proxy can use this information to guess how long the data will stay valid.

HTTP Security

- 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

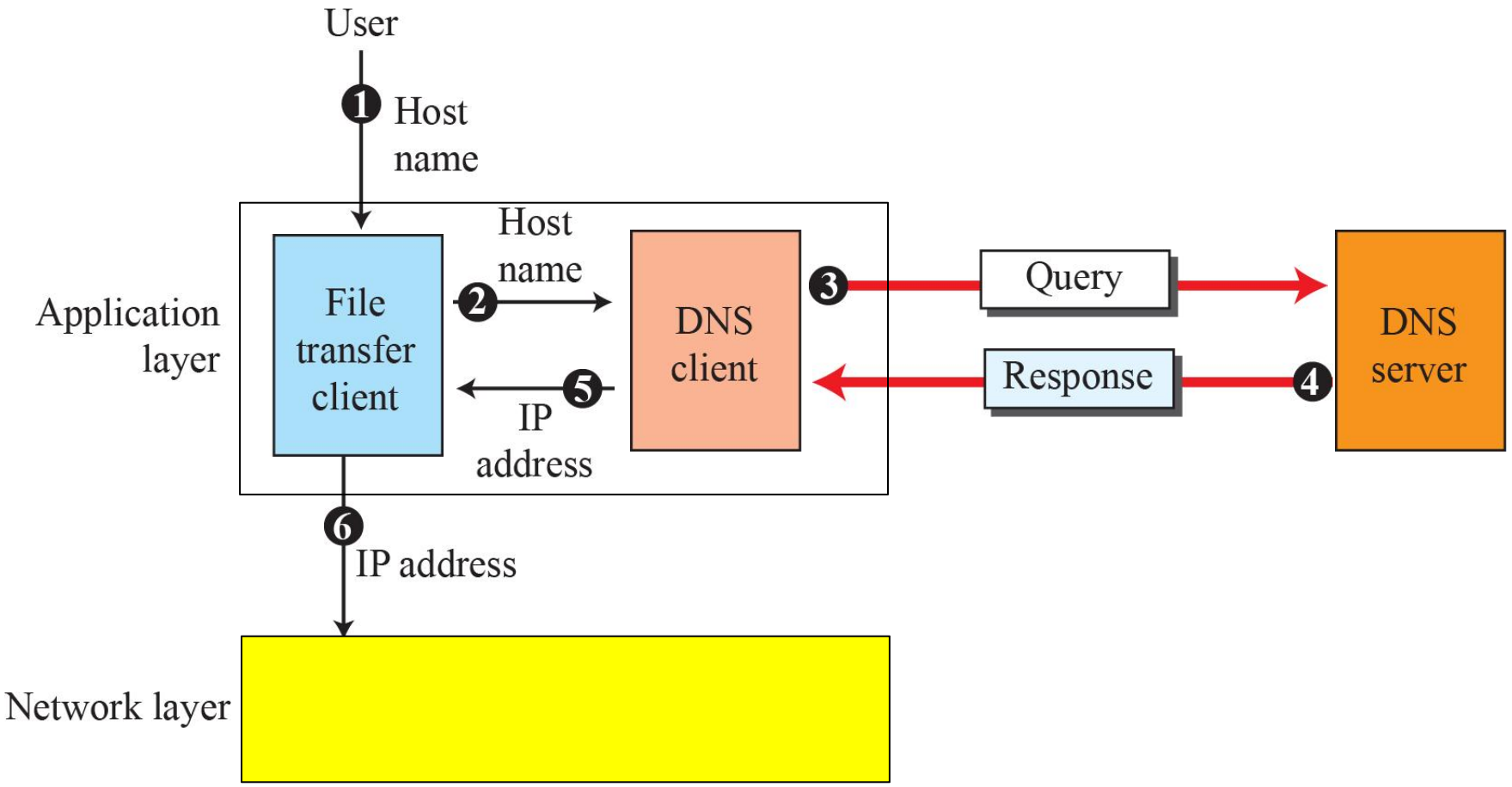
DNS

Domain name system

Domain Name System (DNS)

1. The client/server programs can be divided into two categories: those that can be directly used by the user, such as e-mail, and those that support other application programs.
2. The Domain Name System (DNS) is a supporting program that is used by other programs such as e-mail.
3. A user of an e-mail program may know the e-mail address of the recipient; however, the IP protocol needs the IP address.
4. The DNS client program sends a request to a DNS server to map the e-mail address to the corresponding IP address.
5. To identify an entity, TCP/IP protocols use the IP address, which uniquely identifies the connection of a host to the Internet.
6. 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.

Purpose of DNS



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:
    1. flat name space
    2. hierarchical.

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 main disadvantage of a flat name space is that it cannot be used in a large system such as the Internet because it must be centrally controlled to avoid ambiguity and duplication.

Hierarchical Name Space

In a hierarchical name space, each name is made of several parts.

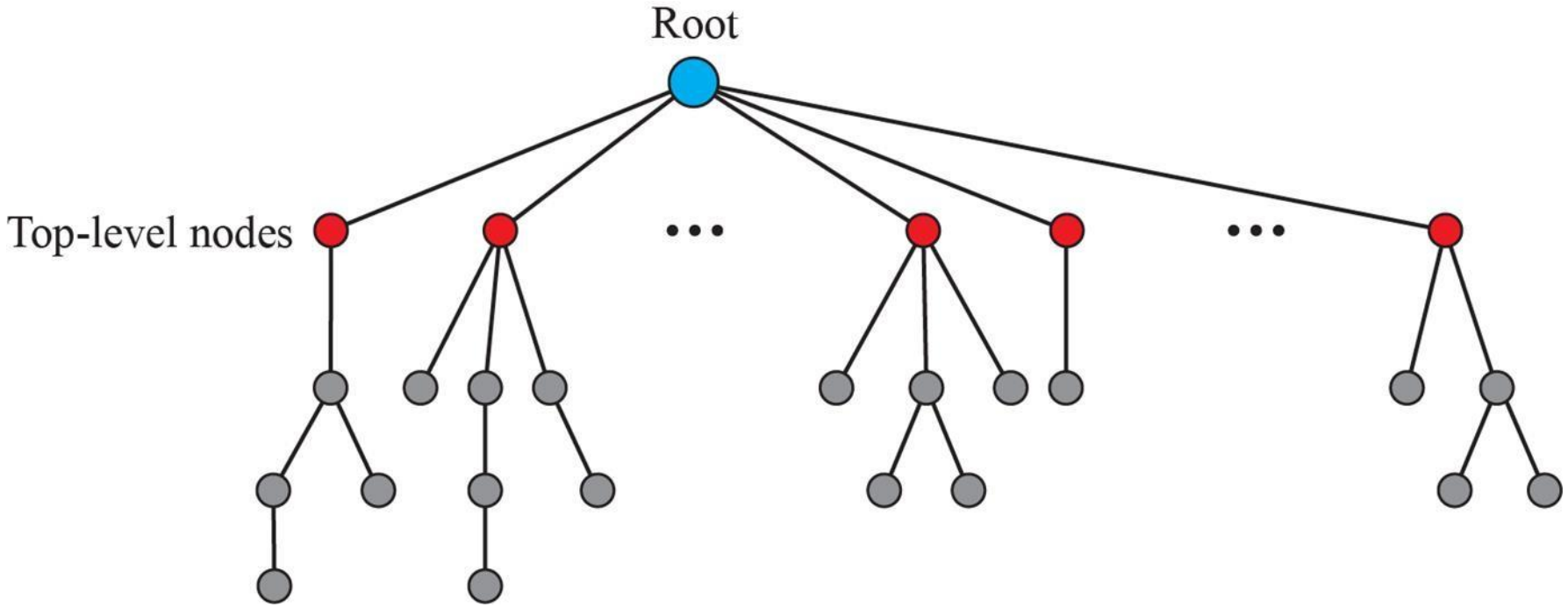
The first part can define the nature of the organization, the second part can define the name of an organization, the third part can define departments in the organization, and so on.

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.



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

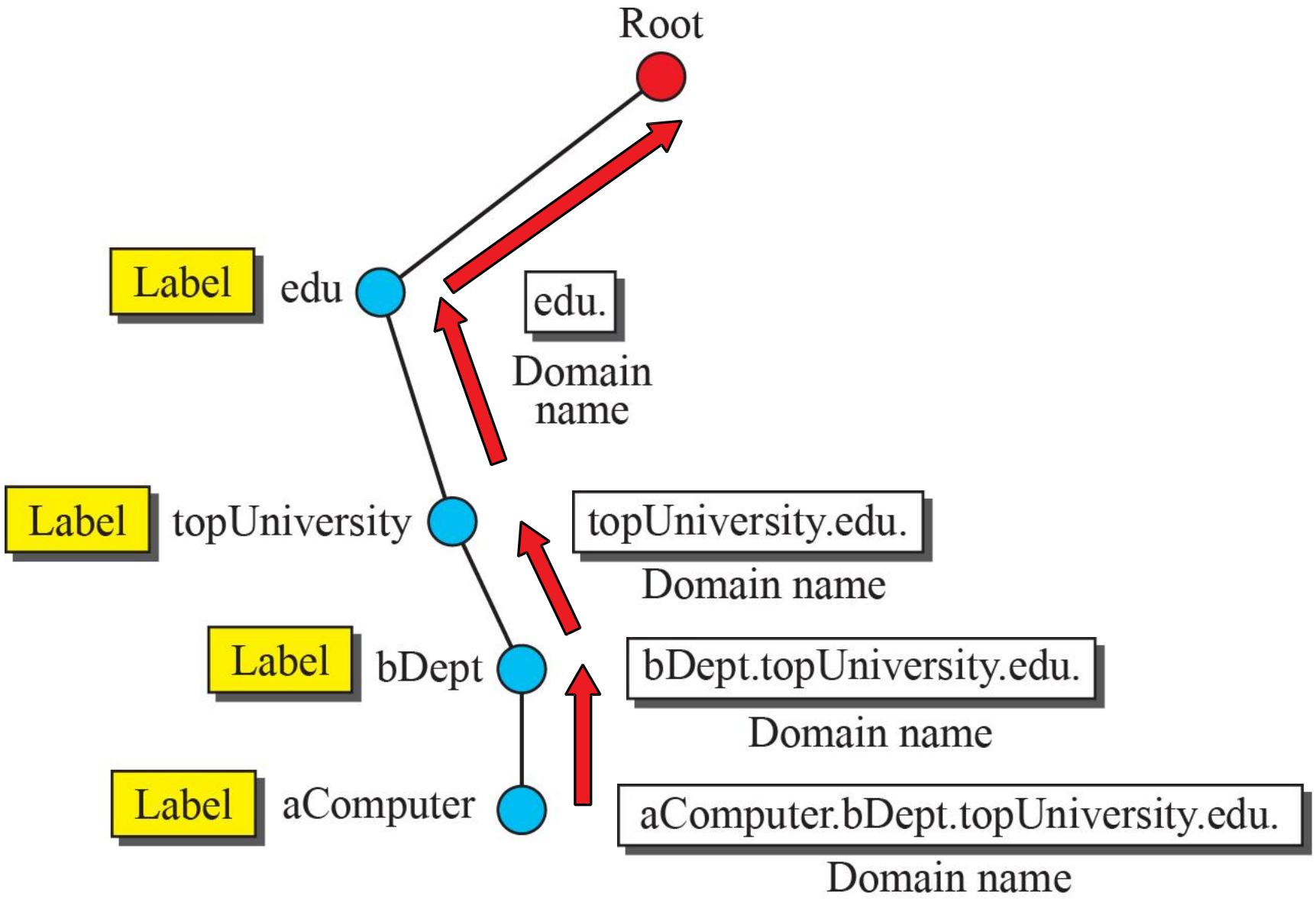
- Each node in the tree has a domain name.
- A full domain name is a sequence of labels

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


Root
Label
edu
edu.
Domain
name
Label
topUniversity
topUniversity.edu.
Domain name
Label
bDept
bDept.topUniversity.edu.
Domain name
Label
aComputer
aComputer.bDept.topUniversity.edu.
Domain name

Fully Qualified Domain Name:

- If a label is terminated by a null string, it is called a fully qualified domain name (FQDN).

An FQDN is a domain name that contains the full name of a host.

- It contains all labels, from the most specific to the most general, that uniquely define the name of the host. For example, the domain name
  - challenger.atc.tbda.edu.
- is the FQDN of a computer named challenger installed at the Advanced Technology Center (ATC)
- at De Anza College.
- A DNS server can only match an FQDN to an address.
- Note that the name must end with a null label, but because null means nothing, the label ends with a dot (.).

Partially Qualified Domain Name:

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.

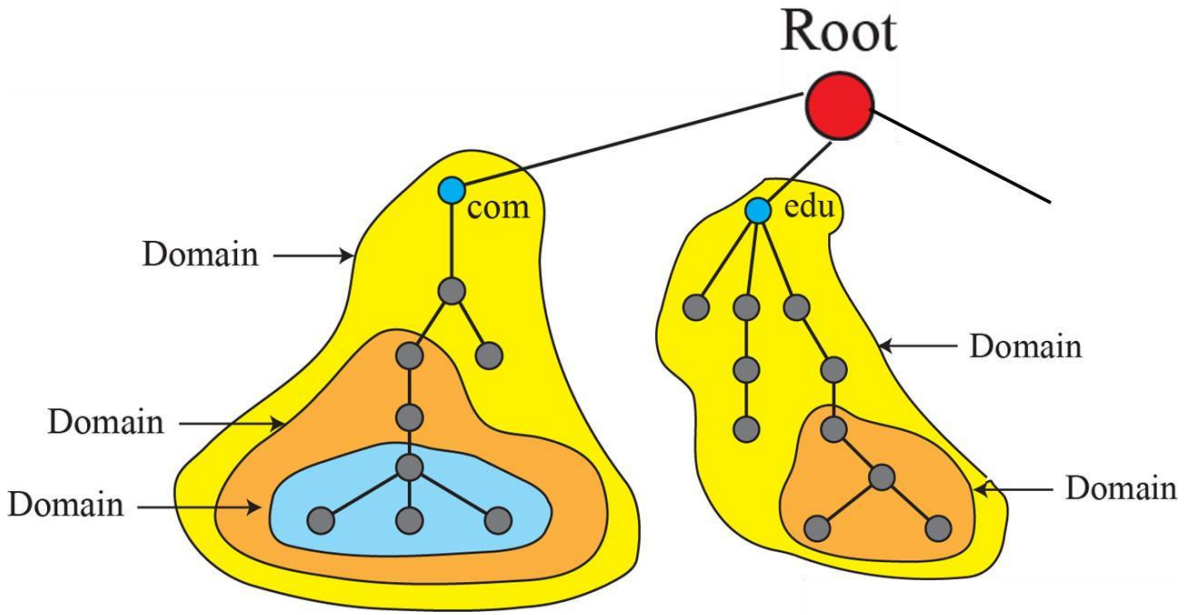
Here the resolver can supply the missing part, called the suffix, to create an FQDN. For example, if a user at the jhda.edu. site wants to get the IP address of the challenger computer, he or she can define the partial name

challenger

The DNS client adds the suffix atc.jhda.edu. before passing the address to the DNS server.

Domain:

1. A **domain is a sub-tree of the domain name space.**
2. The name of the domain is the domain name of the node at the top of the sub-tree.
3. Note that a domain may itself be divided into domains (or sub-domains).



DISTRIBUTION OF NAME SPACE

The information contained in the domain name space must be stored.

However, it is very inefficient and also unreliable 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 unreliable because any failure makes the data inaccessible.

Hierarchy of Name Servers

The solution to these problems is to distribute the information among many computers called DNS servers.

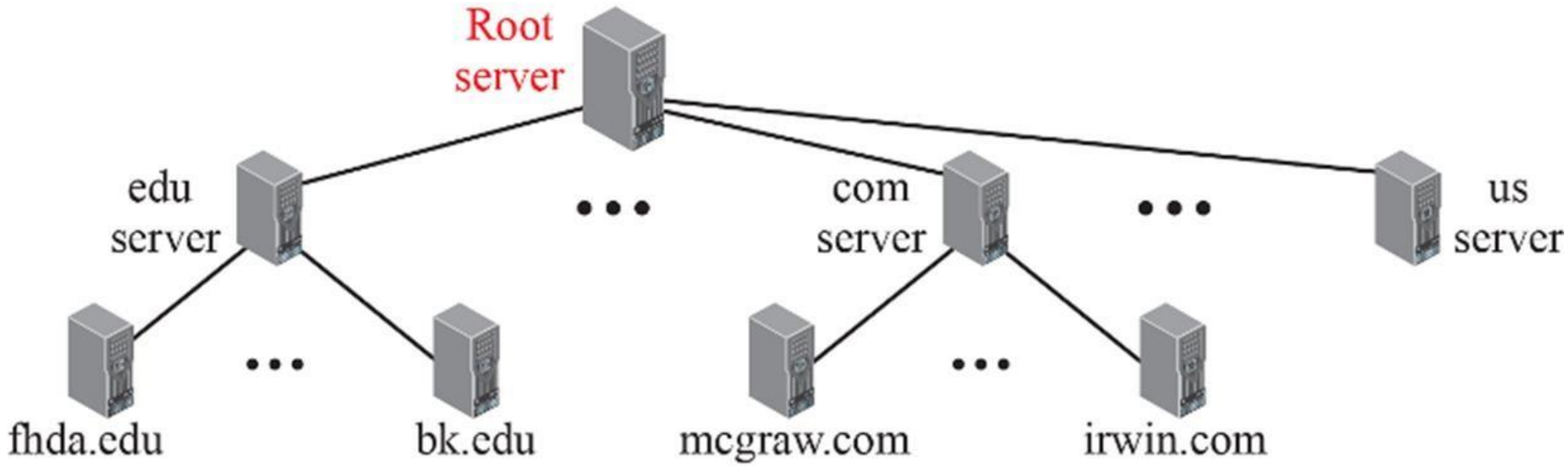
One way to do this is to divide the whole space into many domains based on the first level.

In other words, we let the root stand alone and create as many domains (sub-trees) as there are first-level nodes.

Because a domain created in this way could be very large, DNS allows domains to be divided further into smaller domains (sub-domains).

Each server can be responsible (authoritative) for either a large or a small domain.

In other words, we have **a hierarchy of servers in the same way that we have a hierarchy of names**.


Root
server
edu
server
com
server
us
server
fhda.edu
bk.edu
mcgraw.com
irwin.com

Zone:

Since the complete domain name hierarchy cannot be stored on a single server, it is divided among many servers.

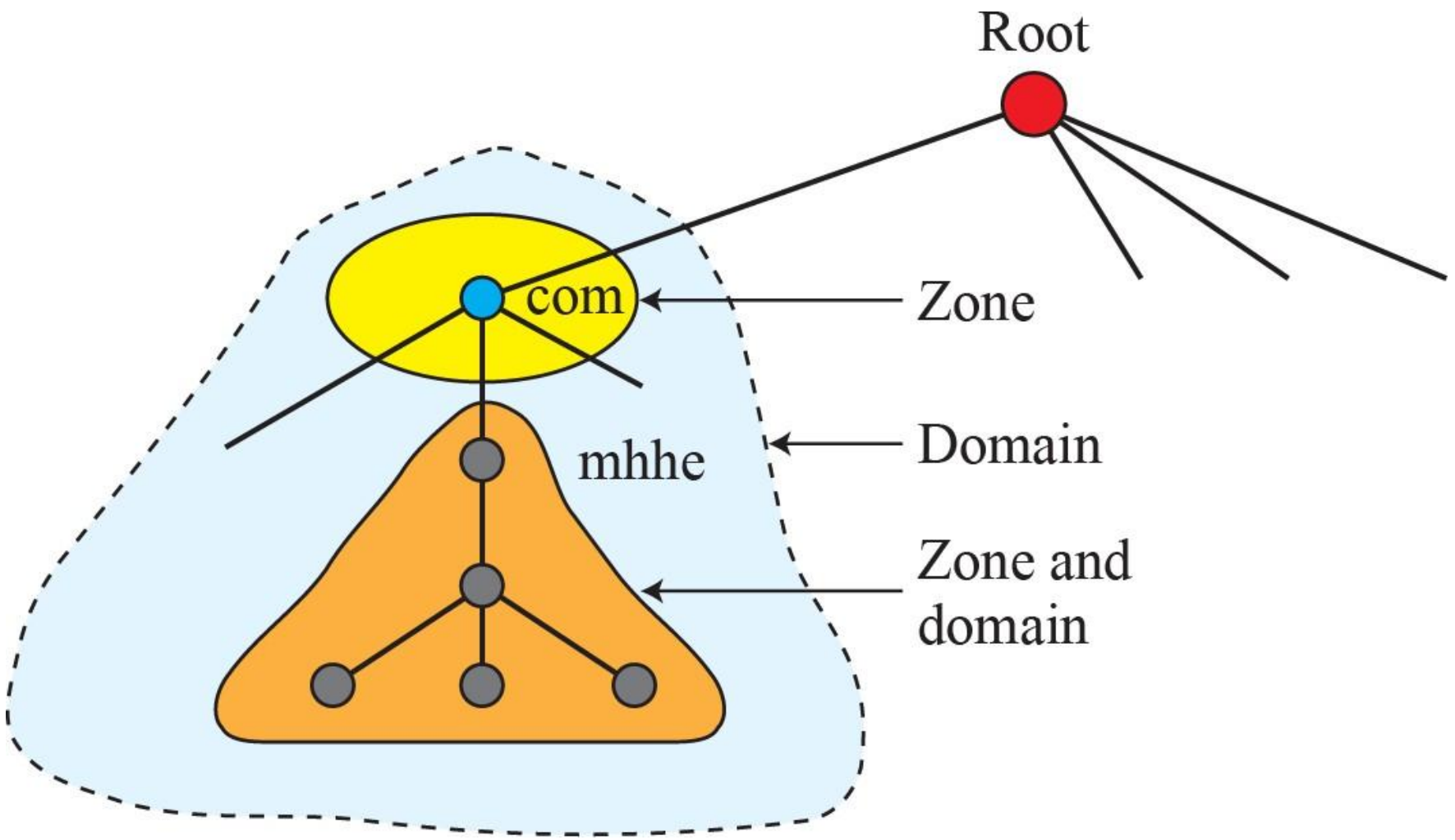
What a server is responsible for or has authority over is called a zone.

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.

However, if a server divides its domain into sub-domains and delegates part of its authority to other servers, domain and zone refer to different things.

The information about the nodes in the sub-domains is stored in the servers at the lower levels, with the original server keeping some sort of reference to these lower-level servers.


Root
com
Zone
mhhe
Domain
Zone and
domain

Root Server

- A root server is a server whose zone consists of the whole tree.
- A **root server usually does not store any information** about domains but delegates its authority
- to other servers, **keeping references** to those servers.
- There are several root servers, each covering the whole domain name space. The 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.
- Note also that a server can be a primary server for a specific zone and a secondary server for another zone

DNS IN THE INTERNET

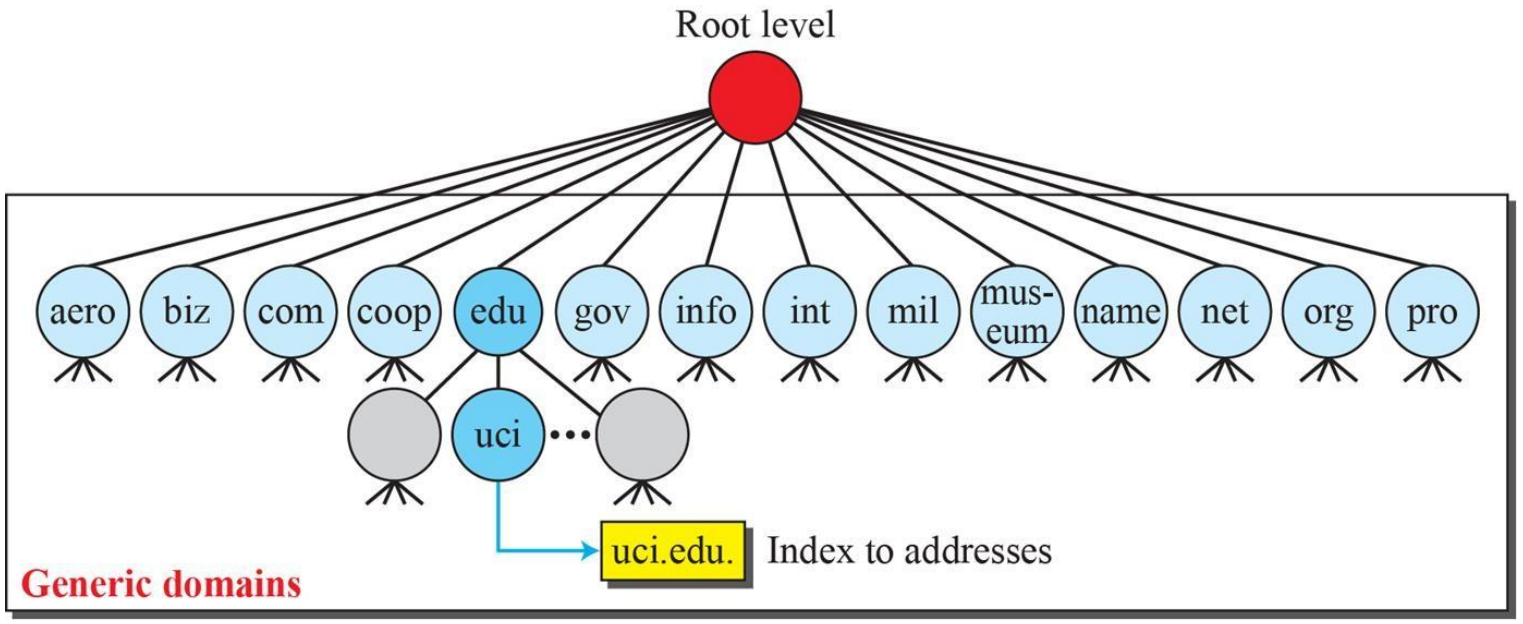
DNS is a protocol that can be used in different platforms. In the Internet, the domain name space (tree) is divided into three different sections: **generic domains, country domains, and the inverse domain.**

Generic Domains:

It defines the registered hosts according to their generic behavior.

Each node in a tree defines the domain name, which is an index to the DNS database.

It uses three-character labels, and these labels describe the **organization type**.



Cont..

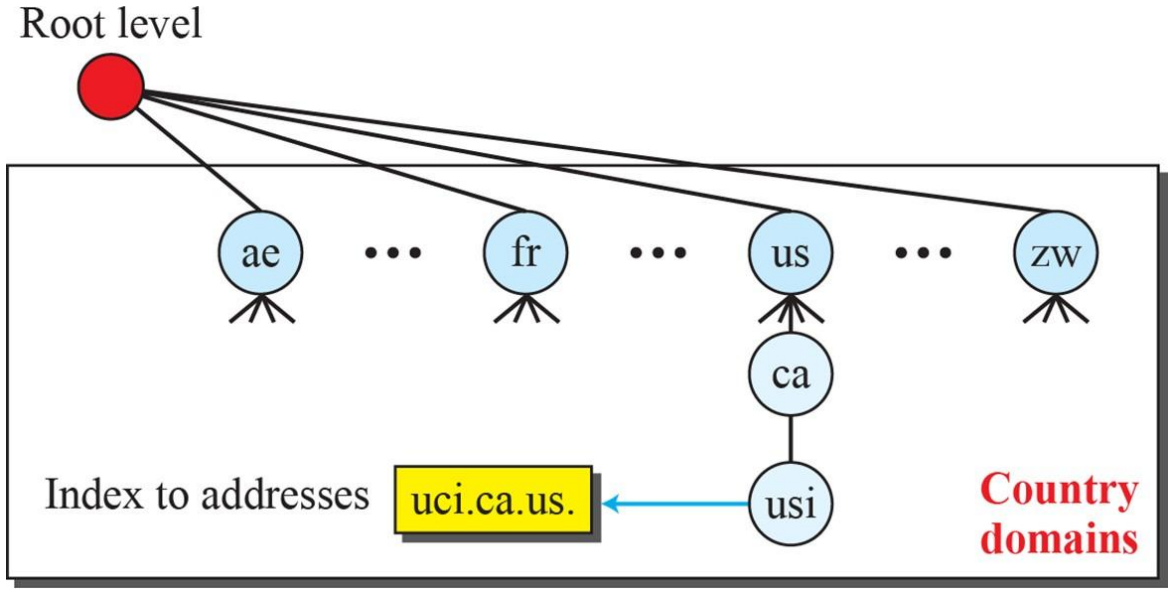
Label	*Description*
aero	Airlines and aerospace companies
biz	Businesses or firms (similar to "com")
com	Commercial organizations
coop	Cooperative business organizations
edu	Educational institutions
gov	Government institutions
info	Information service providers
int	International organizations
mil	Military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

Generic domain labels

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.).



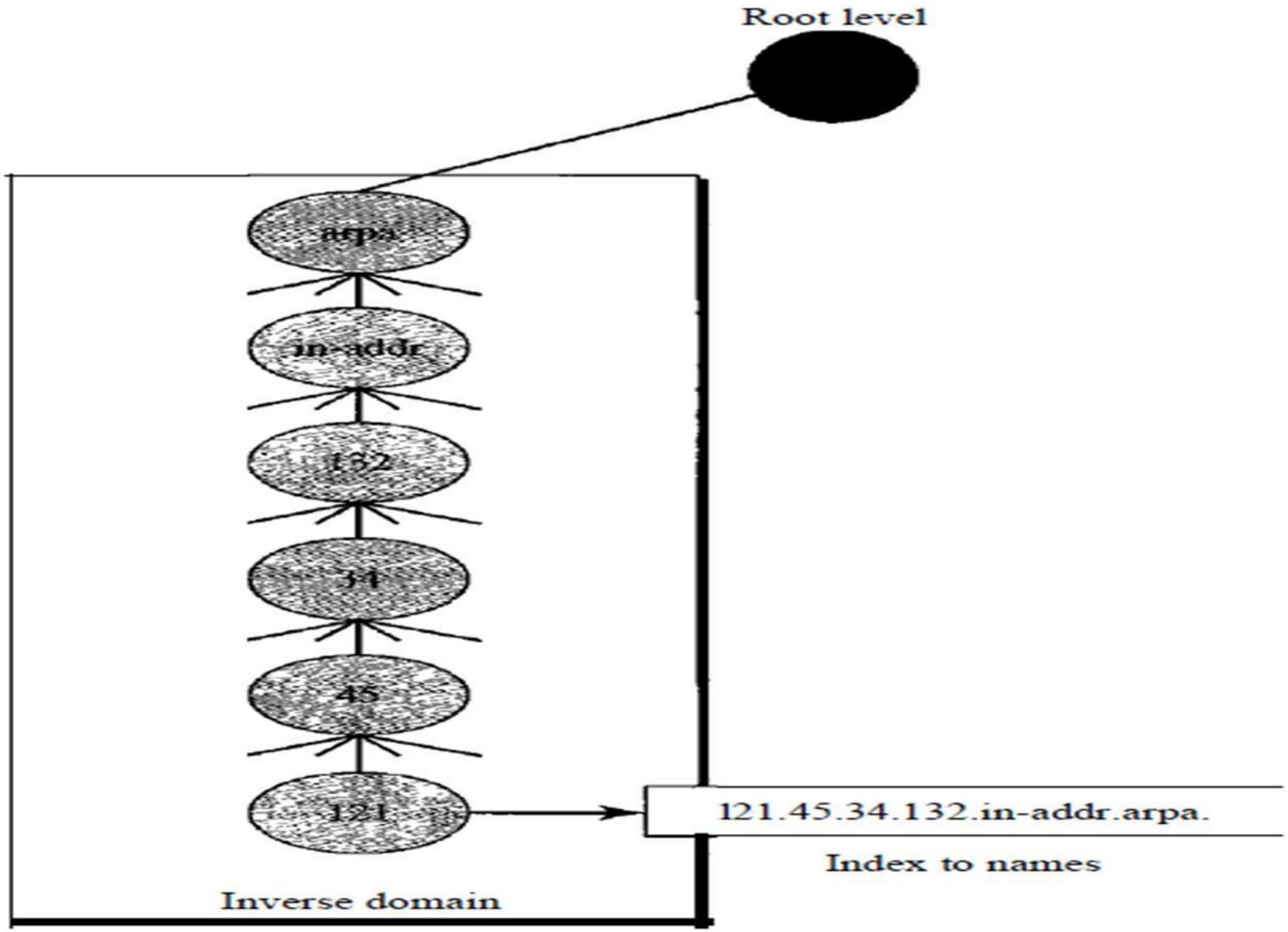
Inverse Domain

It is used for mapping an address to a name.

When a client requests to the server, **the server has a list of authorized clients**.

The server asks its resolver to send the query to the DNS server to verify the client belongs to the list of authorized clients and sends a query to the DNS server to map an address to the name.

This type of query is called an inverse or pointer (PTR) query. To handle a pointer query, the inverse domain is added to the domain name space with the first-level node called arpa. The second level is also one single node named in-addr (for inverse address). The rest of the domain defines IP addresses.


Root level
arpa
in-addr
132
34
45
121
121.45.34.132.in-addr.arpa.
Index to names
Inverse domain

RESOLUTION

Mapping a name to an address or an address to a name is called name-address resolution.

Resolver:

1. 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 or a resolver.
2. The resolver accesses the **closest DNS server with a mapping request**.
3. If the server has the information, it satisfies the resolver; otherwise, it either refers the **resolver to other servers** or **asks other servers to provide** the information.
4. 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.

Mapping Names to Addresses

Most of the time, the resolver gives a domain name to the server and asks for the corresponding address.

In this case, the server checks the generic domains or the country domains to find the mapping.

In both the cases, the query is sent by the resolver to the local DNS server for resolution. If the local server cannot resolve the query, it either refers the resolver to other servers or asks other servers directly.

Mapping Addresses to Names:

A client can send an IP address to a server to be mapped to a domain name. As mentioned before, this is called a PTR query.

To answer queries of this kind, DNS uses the inverse domain. However, in the request, the IP address is reversed and the two labels in-addr and arpa are appended to create a domain acceptable by the inverse domain section.

For example, if the resolver receives the IP address 132.34.45.121, **the resolver first inverts the address** and then adds the two labels before sending.

The domain name sent is "121.45.34.132.in-addr.arpa." which is received by the local DNS and resolved.

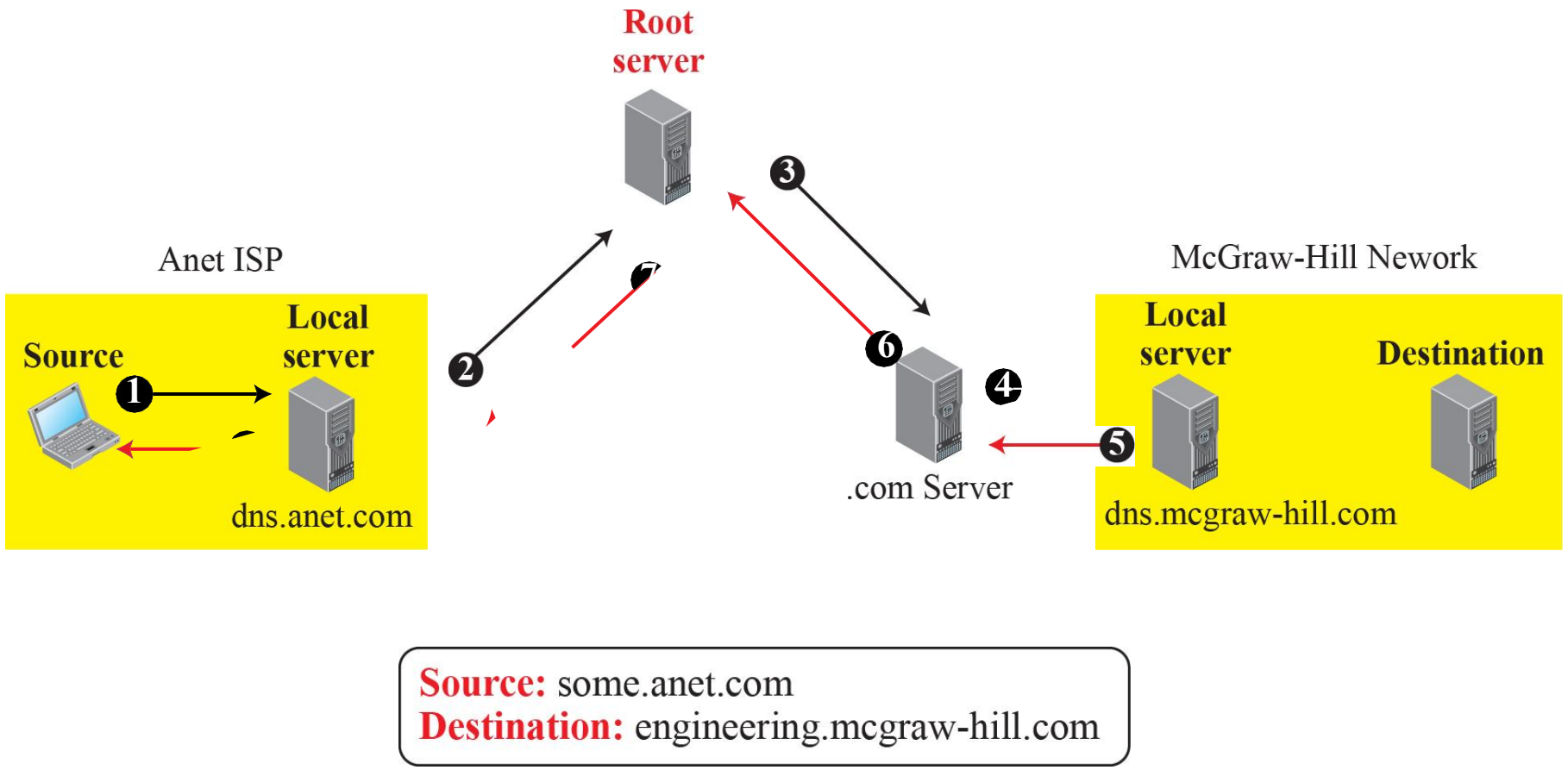
Recursive Resolution

The client (resolver) can ask for a recursive answer from a name server. This means that the resolver expects the server to supply the final answer.

If the server is the authority for the domain name, it checks its database and responds. If the server is not the authority, it sends the request to another server (the parent usually) and waits for the response.

If the parent is the authority, it responds; otherwise, it sends the query to yet another server.

When the query is finally resolved, the response travels back until it finally reaches the requesting client. This is called recursive resolution


Root
server
Anet ISP
McGraw-Hill Nework
Source
Local
server
Local
server
Destination
1
2
3
4
5
6
7
dns.anet.com
.com Server
dns.mcgraw-hill.com
Source: some.anet.com
Destination: engineering.mcgraw-hill.com

Iterative Resolution

If the client does not ask for a recursive answer, the mapping can be done iteratively.

If the server is an authority for the name, it sends the answer. If it is not, it returns (to the client) the IP address of the server that it thinks can resolve the query.

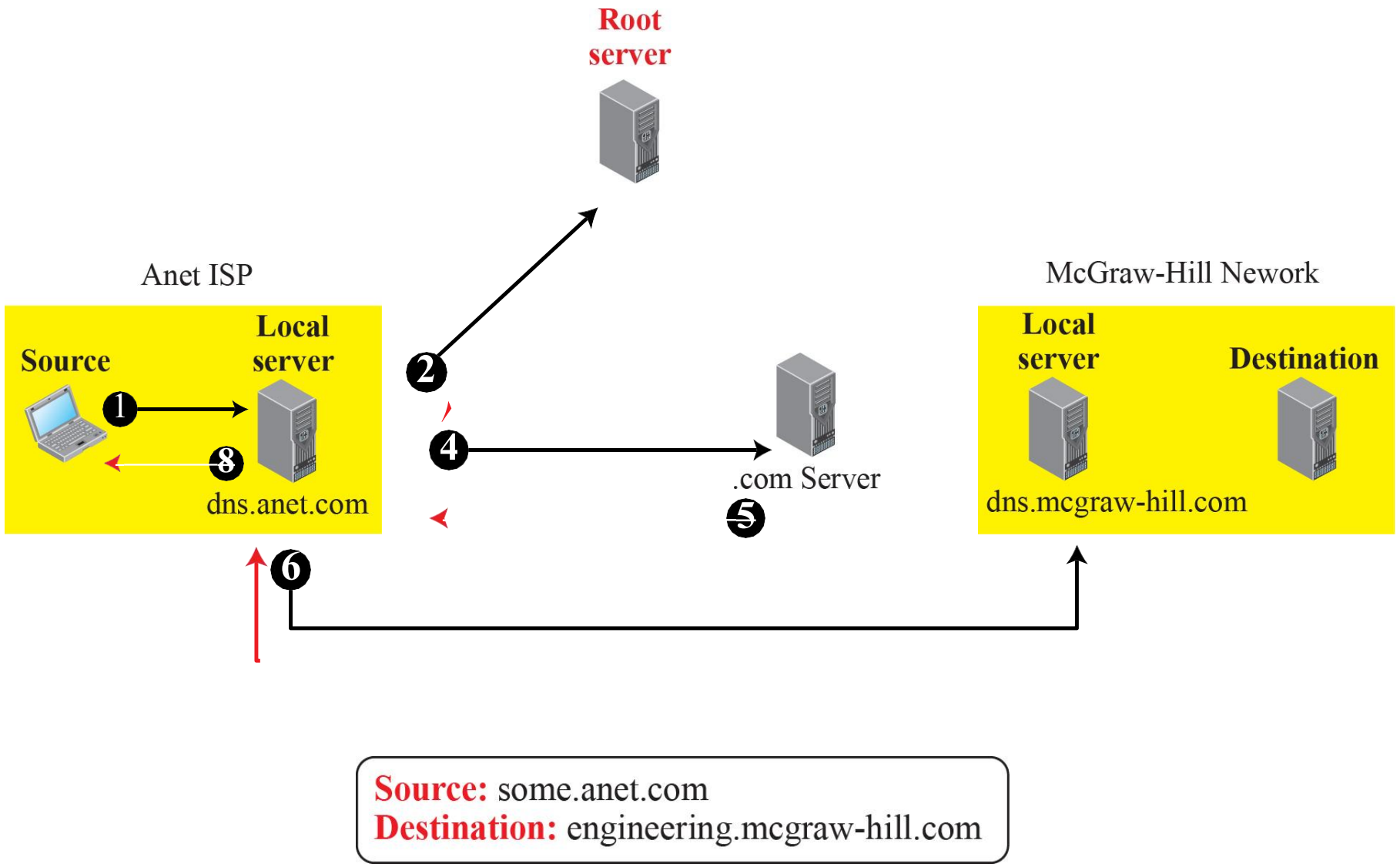
The client is responsible for repeating the query to this second server.

If the newly addressed server can resolve the problem, it answers the query with the IP address; otherwise, it returns

the IP address of a new server to the client.

Now the client must repeat the query to the third server.

This process is called iterative resolution because the client repeats the same query to multiple servers.


Root
server
Anet ISP
Source
Local
server
1
8
dns.anet.com
2
4
.com Server
5
6
McGraw-Hill Nework
Local
server
Destination
dns.mcgraw-hill.com
Source: some.anet.com
Destination: engineering.mcgraw-hill.com

Caching

Each time a server receives a query for a name that is not in its domain, it needs to search its database for a server IP address.

Reduction of this search time would increase efficiency.

DNS handles this with a mechanism called caching.

Caching is used to speeds up resolution.

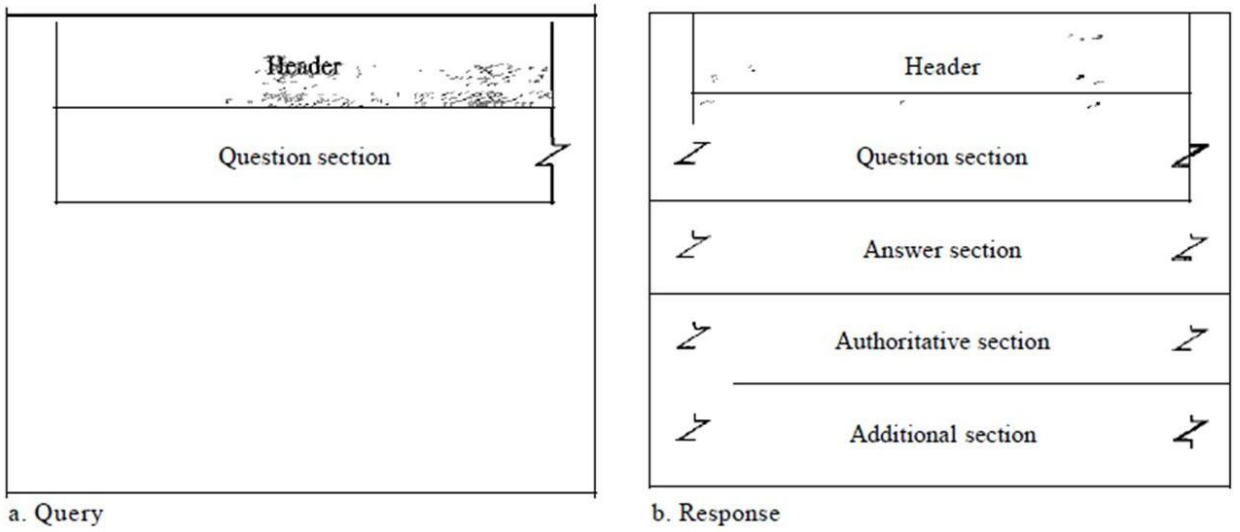
When a server asks for a mapping from another server and receives the response, it stores this information in its cache memory before sending it to the client.

If the same or another client asks for the same mapping, it can check its cache memory and solve 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.

DNS MESSAGES

- DNS has two types of messages: **query and response**. Both types have the same format. The **query message** consists of **a header and question records**.
- The **response message** consists of **a header, question records, answer records, authoritative records, and additional records.**



Question Section:

This is a section consisting of one or more question records.

It is present on both query and response messages.

A question record is **used by the client to get information from a server**. This contains the domain name.

Answer Section:

This is a section consisting of one or more resource records.

It is present only on response messages.

This section includes the **answer from the server to the client** (resolver).

Each domain name (each node on the tree) is associated with a record called the resource record. The server database consists of resource records. Resource records are also what is returned by the server to the client.

Authoritative Section:

This is a section consisting of one or more resource records.

It is present only on response messages.

This section **gives information** (domain name) **about one or more authoritative servers** for the query.

Additional Information Section:

This is a section consisting of one or more resource records.

It is present only on response messages.

This section provides **additional information that may help the resolver**.

For example, a server may give the domain name of an authoritative server to the resolver in the authoritative section, and include the IP address of the same authoritative server in the additional information section.

Header

The **identification** subfield is used by the client to match the response with the query.

The client uses a different identification number each time it sends a query. The server duplicates this number in the corresponding response.

Identification	Flags
Number of question records	Number of answer records (all 0s in query message)
Number of authoritative records (all 0s in query message)	Number of additional records (all 0s in query message)

The **flags** subfield is a collection of subfields that define the type of the message, the type of answer requested, **the type of desired resolution (recursive or iterative)**, and so on.

The **number of question records** subfield contains the number of queries in the question section of the message. The **number of answer** records subfield contains the number of answer records in the answer section of the response message. Its value is zero in the query message.

The number of authoritative records subfield contains the number of authoritative records in the authoritative section of a response message. Its value is zero in the query message.

Finally, the **number of additional records** subfield contains the number additional records in the additional section of a response message.

DNS Record types

DNS record types are records that provide important information about a hostname or domain.

These records include the current IP address for a domain.

Also, DNS records are stored in text files (zone files) on the authoritative DNS server.

.

1. A record

The A record is the most important DNS record type. The "A" in A record stands for "address." An A record shows the IP address for a specific hostname or domain

For example, a DNS record lookup for the domain example.com returns the following result:

Type	Domain Name	IP Address	TTL
A	example.com	93.184.216.34 Verizon Business (AS15133)	24 hrs

The A record only supports IPV4 addresses.

The main use of A record is for IP address lookup. Using an A record, a web browser is able to load a website using the domain name. As a result, we can access websites on the internet without knowing their IP addresses.

Another use of A record is in the domain name system-based blackhole list (DNSBL). Here, the A record is **used to block mail from known spam sources**.

2. AAAA record

AAAA record, just like A record, point to the IP address for a domain. However, this DNS record type is different in the sense that it **points to IPV6 addresses**.

3. **CNAME record:**

A CNAME (Canonical Name) **points one domain or sub-domain to another domain name**, allowing you to update one A Record each time you make a change, regardless of how many Host Records need to resolve to that IP address.

These records point to www.example.com to example.com, imap.example.com to mail.example.com, and docs.example.com to ghs.google.com.

The first record allows the domain to resolve to the same server with or without the www sub-domain. The second record allows you to use an alternative sub-domain for email hosting and delivery. The third record allows you to use the docs.example.com sub-domain with G Suite, where you can use Google's document management system.

4. NS record

A nameserver (NS) record specifies the authoritative DNS server for a domain. In other words, **the NS record helps point to where internet applications like a web browser can find the IP address for a domain name**. Usually, multiple nameservers are specified for a domain.

5. **MX record**

A mail exchange (MX) record, is a DNS record type that shows where emails for a domain should be routed to. In other words, an **MX record makes it possible to direct emails to a mail server**.

Delay and throughput in Packet-switched Network

Delay in Packet- switched Network

Delay in a packet-switched network refers to the time it takes for a packet of data to travel from the source to the destination. There are several components of delay:

- Transmission Delay
- Propagation Delay
- Processing Delay
- Queueing Delay
- Total Delay

- Transmission Delay: This is the time taken to transmit the entire packet onto the communication medium. It depends on the packet's size and the transmission rate of the link.
    - $Delay_{tr}$ = (Packet length) / (Transmission rate).
- Propagation Delay: This is the time it takes for a signal to travel from the source to the destination. It is determined by the physical distance between the two points and the speed of the signal through the transmission medium.
    - $Delay_{pg}$ = (Distance) / (Propagation speed).
- Processing Delay: When a packet arrives at a router or a network device, there is a small processing delay as the device inspects the packet headers and makes forwarding decisions.
    - $Delay_{pr}$ = Time required to process a packet in a router or a destination host

- Queueing Delay: If there are other packets ahead of the current packet waiting to be transmitted on the same link, the packet will experience a delay in the queue before it gets its turn for transmission.
  - $Delay_{qu}$ = The time a packet waits in input and output queues in a router
- Total Delay: assuming equal delays for the sender, routers, and receiver, the total delay (source-to-destination delay) a packet encounters can be calculated if we know the number of routers, n, in the whole path.

$$\text{Total delay} = (n + 1)\ (Delay_{tr} + Delay_{pg} + Delay_{pr}\ ) + (n)\ (Delay_{qu}\ )$$

- Delay can have a significant impact on real-time applications such as voice and video, where low delay is essential to maintain smooth communication.

Throughput in Packet- switched Network

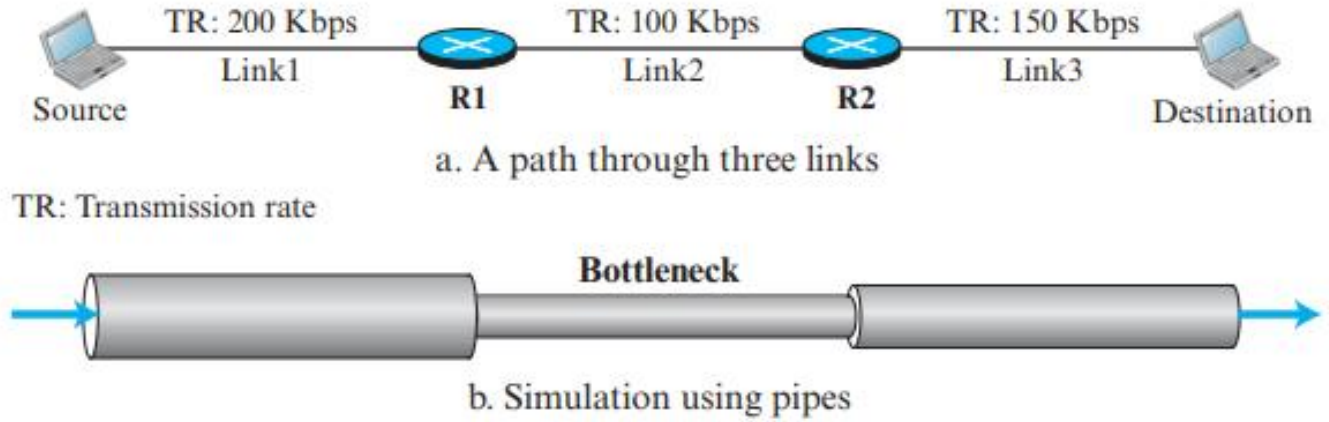
Throughput in a packet-switched network refers to the rate at which data can be transmitted over the network link. It is usually measured in bits per second (bps) or packets per second (pps). Throughput depends on various factors:

- Link Capacity
- Network Congestion
- Protocol Overheads
- Routing Efficiency

- Link Capacity: The maximum data rate that the physical link can support. It is determined by the link's transmission rate and bandwidth.
- Network Congestion: High levels of network congestion can reduce throughput as packets experience more queueing delay and compete for available bandwidth.
- Protocol Overheads: Packet-switched networks use various protocols to manage and control data transmission. These protocols add some overhead, which can slightly reduce the actual throughput.

- Routing Efficiency: The efficiency of the routing algorithms and protocols used in the network can impact how quickly packets reach their destination.
- Throughput is crucial for applications that require high data transfer rates, such as file downloads, streaming, and data-intensive processes.

Figure 4.10 *Throughput in a path with three links in a series*



In this figure, the data can flow at the rate of 200 Kbps in Link1. However, when the data arrives at router R1, it cannot pass at this rate. Data needs to be queued at the router and sent at 100 Kbps. When data arrives at router R2, it could be sent at the rate of 150 Kbps, but there is not enough data to be sent. In other words, the average rate of the data flow in Link3 is also 100 Kbps. We can conclude that the average data rate for this path is 100 Kbps, the minimum of the three different data rates. The figure also shows that we can simulate the behavior of each link with pipes of different sizes; the average throughput is determined by the bottleneck, the pipe with the smallest diameter.
In general, in a path with n links in series, we have

$$\text{Throughput} = \text{minimum } \{TR1, TR2, \ldots TRn\}.$$