Application Layer

- In this section, we introduce the nature of services provided by the Internet and introduce two categories of applications: the traditional one, based on the client server paradigm, and the new one, based on the peer-to-peer paradigm.

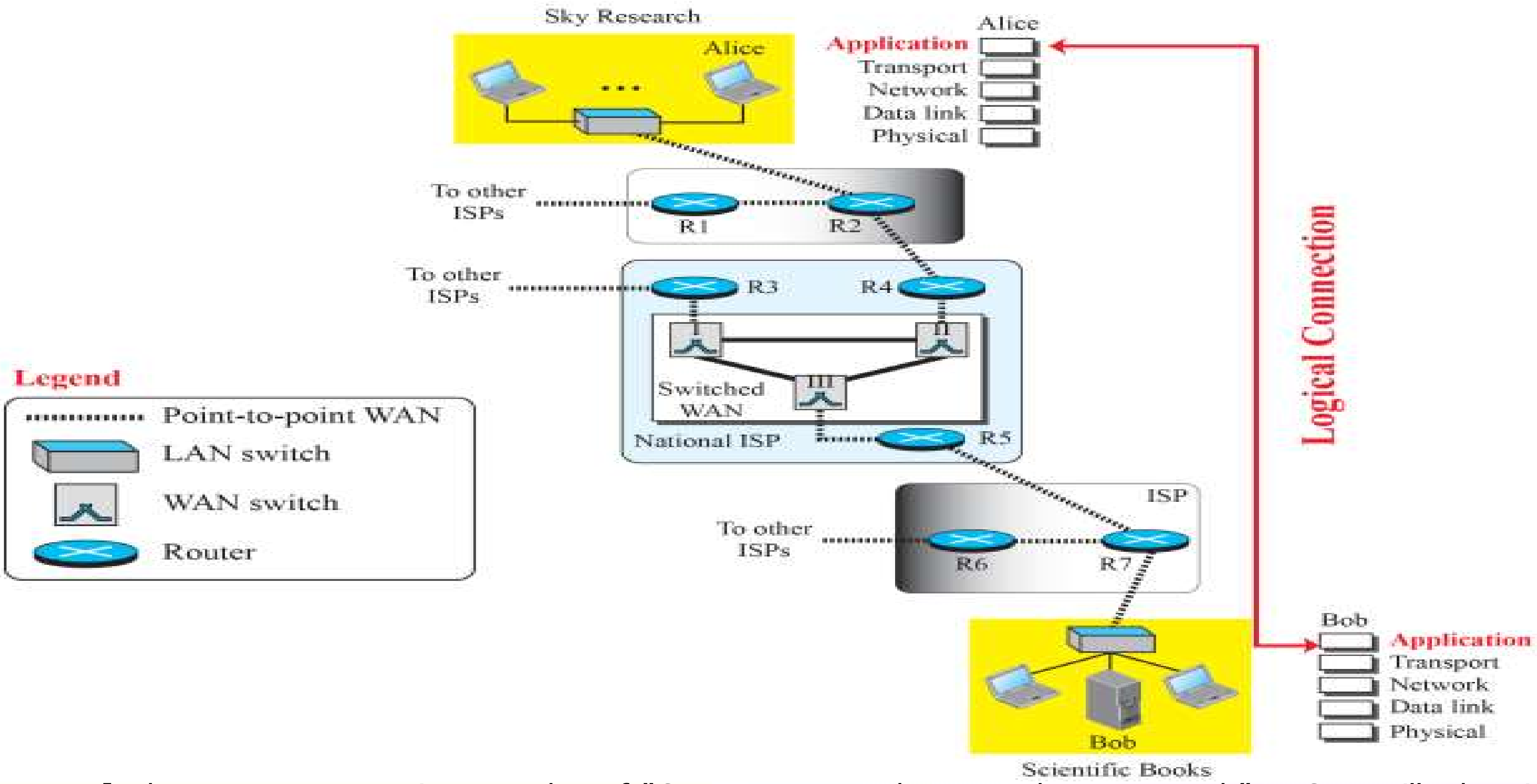
- We also discuss some predefined or standard applications based on the client-server paradigm such as surfing the Web, file transfer, e-mail, and so on.

- We also discuss the concept and protocols in the peer-to-peer paradigm.

Application Layer

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

Figure 1.29 *Logical connection at the application layer*



[Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top down Approach", McGraw Hill Education]

Application Layer

- The application layer is different from other layers in that it is the highest layer in the suite.
- The protocols in this layer do not provide services to any other protocol in the suite; they only receive services from the protocols in the transport layer.
- This means that protocols can be removed from this layer easily.
- New protocols can be also added to this layer as long as the new protocol can use the service provided by one of the transport-layer protocols.

Application Layer

Standard and Nonstandard Protocols

- Standard Application-Layer Protocols are several application-layer protocols that have been standardized and documented by the Internet authority, and we are using them in our daily interaction with the Internet.
- Each standard protocol is a pair of computer programs that interact with the user and the transport layer to provide a specific service to the user.

- Nonstandard Application-Layer Protocols: A programmer can create a nonstandard application-layer program if she can write programs that provide service to the user by interacting with the transport layer.

Application-Layer Paradigms

- To use the Internet we need two application programs to interact with each other: one running on a computer somewhere in the world, the other running on another computer somewhere else in the world.
- The two programs need to send messages to each other through the Internet infrastructure.
- Should both application programs be able to request services and provide services, or should the application programs just do one or the other?
- Two paradigms have been developed during the lifetime of the Internet to answer this question: the client-server paradigm and the peer-to-peer paradigm.

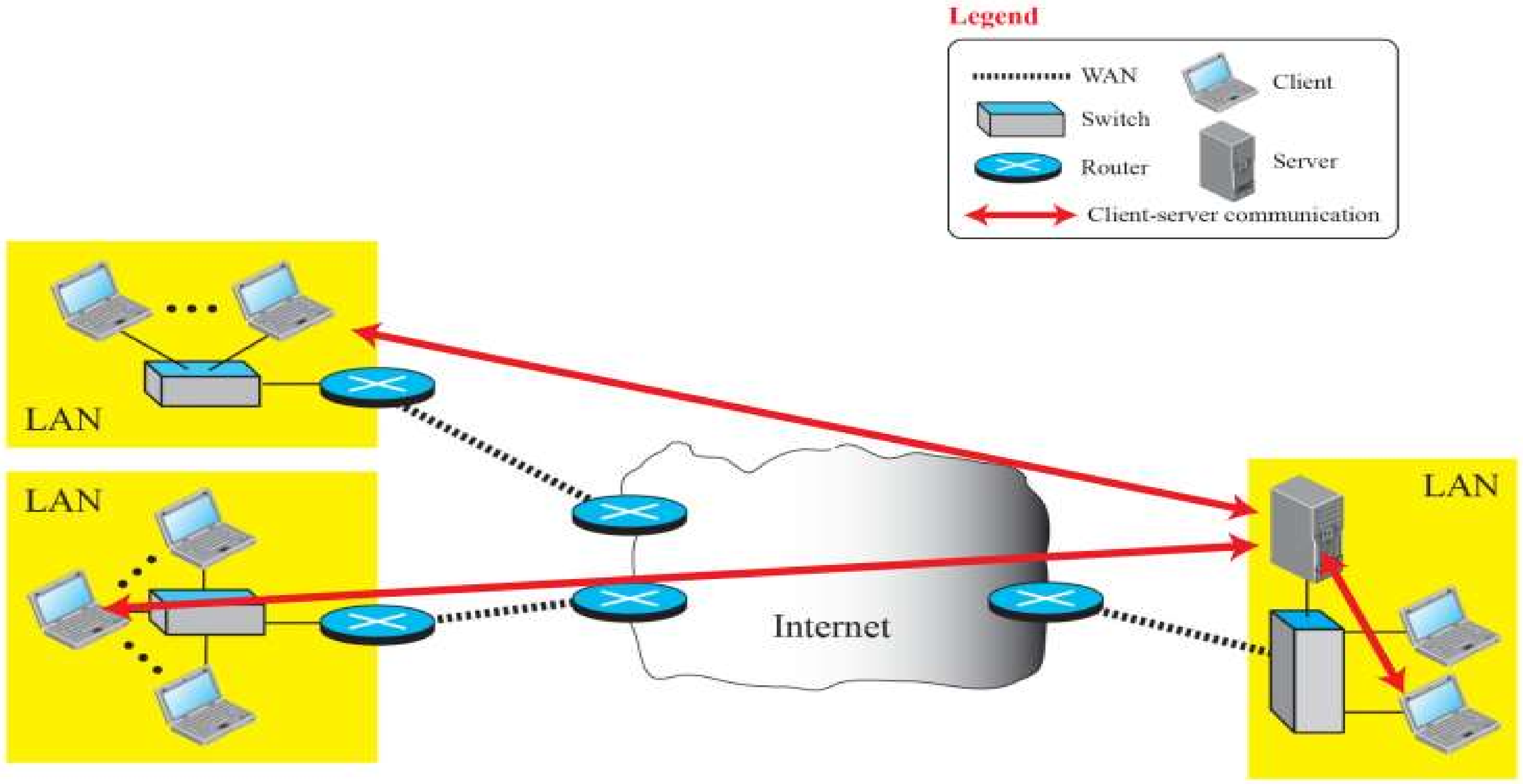
Application-Layer Paradigms

Traditional Paradigm: Client-Server

- The traditional paradigm is called the client-server paradigm.
- In this paradigm, the service provider is an application program, called the server process; it runs continuously, waiting for another application program, called the client process, to make a connection through the Internet and ask for service.
- The server process must be running all the time; the client process is started when the client needs to receive service.

Application-Layer Paradigms

Figure 1.30 Example of a client-server paradigm



[Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top down Approach", McGraw Hill Education]

Application-Layer Paradigms

Traditional Paradigm: Client-Server

- One problem with this paradigm is that the concentration of the communication load is on the shoulder of the server, which means the server should be a powerful computer.
- Even a powerful computer may become overwhelmed if a large number of clients try to connect to the server at the same time.
- Another problem is that there should be a service provider willing to accept the cost and create a powerful server for a specific service, which means the service must always return some type of income for the server in order to encourage such an arrangement.
- Several traditional services are still using this paradigm, including the World Wide Web (WWW) and its vehicle Hyper Text Transfer Protocol (HTTP), File Transfer Protocol (FTP), secure shell (SSH), e-mail, and so on.

Application-Layer Paradigms

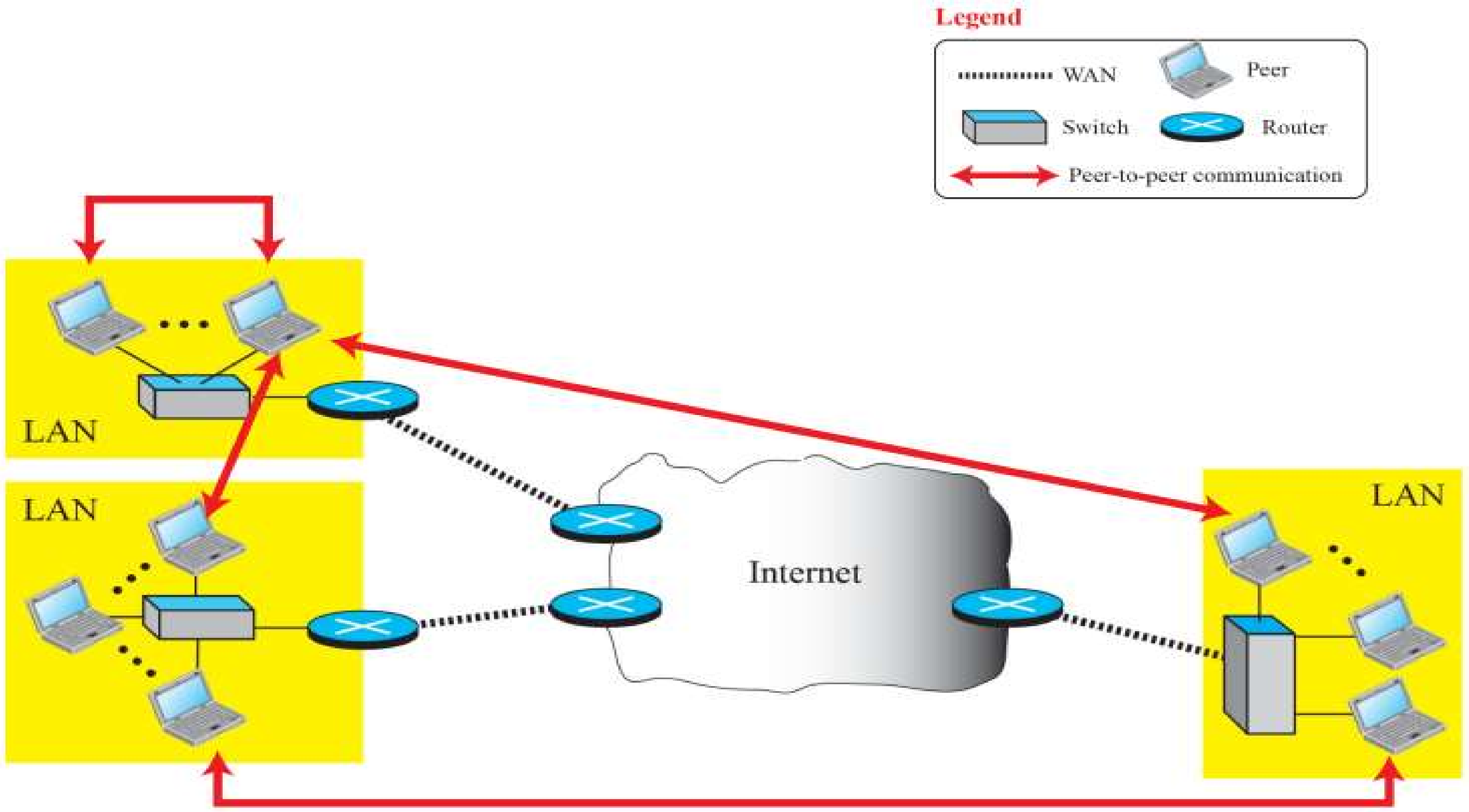
New Paradigm: Peer-to-Peer

- A new paradigm, called the peer-to-peer paradigm (often abbreviated P2P paradigm) has emerged to respond to the needs of some new applications.
- In this paradigm, there is no need for a server process to be running all the time and waiting for the client processes to connect.
- The responsibility is shared between peers.
- A computer connected to the Internet can provide service at one time and receive service at another time.
- A computer can even provide and receive services at the same time.
- One of the areas that really fits in this paradigm is the Internet telephony.

Application-Layer Paradigms

New Paradigm: Peer-to-Peer

Figure 1.31 ***Example of a peer-to-peer paradigm***



[Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top down Approach", McGraw Hill Education]

Application-Layer Paradigms

New Paradigm: Peer-to-Peer

- Peer-to-peer paradigm has been proved to be easily scalable and cost-effective in eliminating the need for expensive servers to be running and maintained all the time, there are also some challenges.
- The main challenge has been security; it is more difficult to create secure communication between distributed services than between those controlled by some dedicated servers.
- The other challenge is applicability; it appears that not all applications can use this new paradigm.
- For example, not many Internet users are ready to become involved, if one day the Web can be implemented as a peer-to-peer service.
- There are some new applications, such as BitTorrent [communication protocol for peer to peer file sharing], Skype, IPTV, and Internet telephony, that use this paradigm.

Application-Layer Paradigms

Mixed Paradigm

- An application may choose to use a mixture of the two paradigms by combining the advantages of both.
- For example, a light-load client-server communication can be used to find the address of the peer that can offer a service. When the address of the peer is found, the actual service can be received from the peer by using the peer-to-peer paradigm.

CLIENT-SERVER PARADIGM

- In a client-server paradigm, communication at the application layer is between two running application programs called processes: a client and a server.
- A client is a running program that initializes the communication by sending a request; a server is another application program that waits for a request from a client.
- The server handles the request received from a client, prepares a result, and sends the result back to the client.
- This definition of a server implies that a server must be running when a request from a client arrives, but the client needs to be run only when it is needed.

CLIENT-SERVER PARADIGM

- If we have two computers connected to each other somewhere, we can run a client process on one of them and the server on the other.
- We need to be careful that the server program is started before we start running the client program.
- In other words, the lifetime of a server is infinite: it should be started and run forever, waiting for the clients.
- The lifetime of a client is finite: it normally sends a finite number of requests to the corresponding server, receives the responses, and stops

CLIENT-SERVER PARADIGM

How can a client process communicate with a server process?

CLIENT-SERVER PARADIGM

Application Programming Interface

How can a client process communicate with a server process?

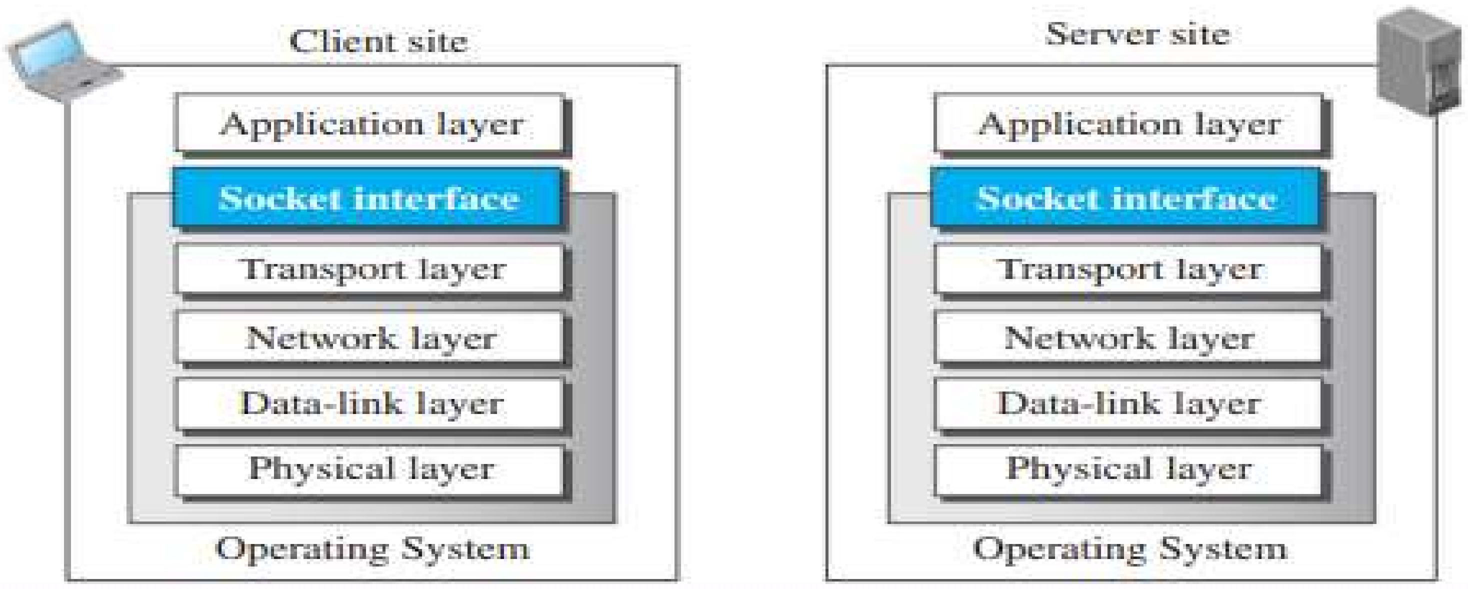
- If we need a process to be able to communicate with another process, we need a new set of instructions to tell the lowest four layers of the TCP/IP suite to open the connection, send and receive data from the other end, and close the connection.
- A set of instructions of this kind is normally referred to as Application Programming Interface (API).
- An interface in programming is a set of instructions between two entities.
- In this case, one of the entities is the process at the application layer and the other is the operating system that encapsulates the first four layers of the TCP/IP protocol suit.
- Several APIs have been designed for communication like socket interface, Transport Layer Interface (TLI), and STREAM.

CLIENT-SERVER PARADIGM

Sockets

- Although a socket is supposed to behave like a terminal or a file, it is not a physical entity like them; it is an abstraction.
- It is a data structure that is created and used by the application program.

Figure 1.32 *Position of the socket interface*



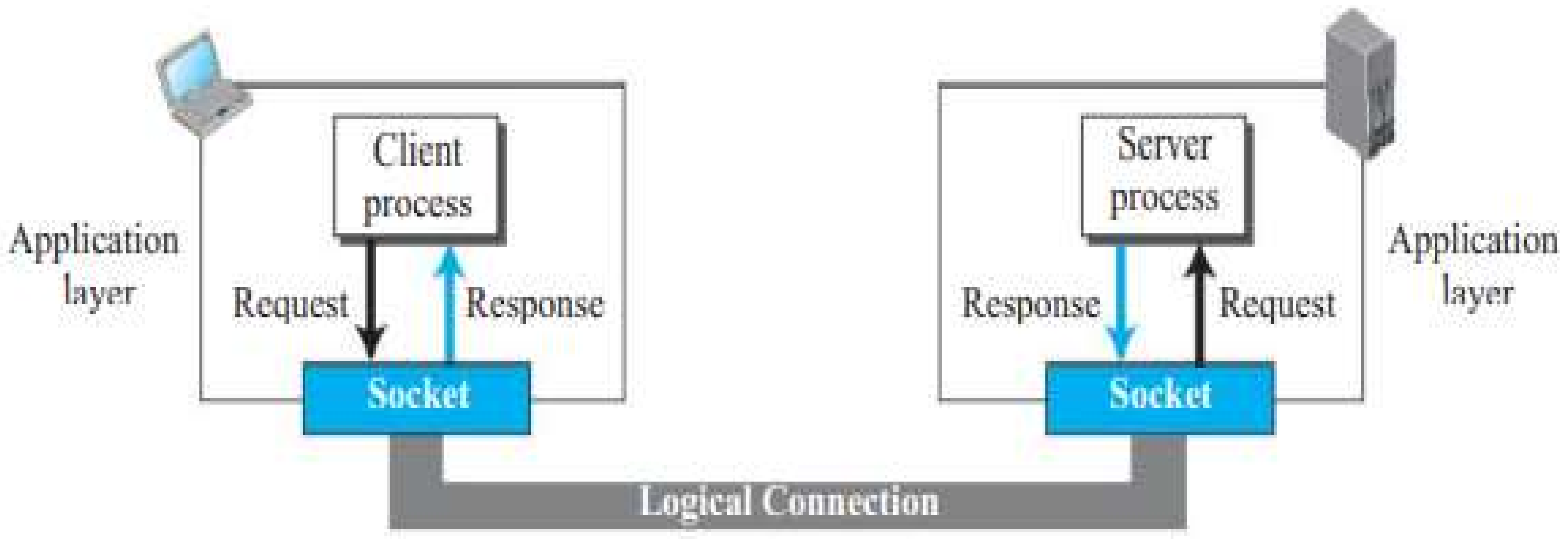
[Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top down Approach", McGraw Hill Education]

CLIENT-SERVER PARADIGM

Sockets

- Communication between a client process and server process is communication between two sockets, created at two ends as shown in the figure.

Figure 1.33 *Use of sockets in process-to-process communication*



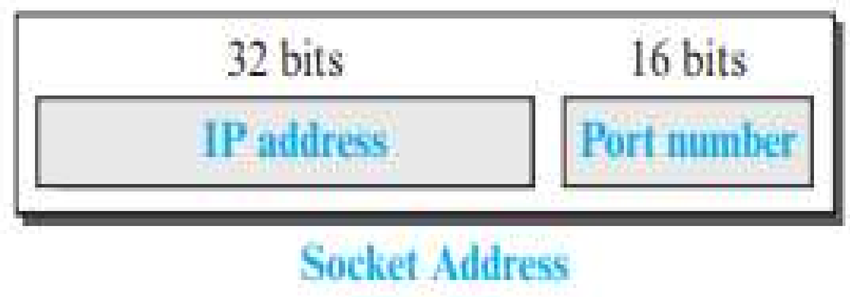
[Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top down Approach", McGraw Hill Education]

CLIENT-SERVER PARADIGM

Socket Addresses

- Since communication in the client-server paradigm is between two sockets, we need a pair of socket addresses for communication: a local socket address and a remote socket address.

Figure 1.34 *A socket address*



CLIENT-SERVER PARADIGM

Server Site

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

Local Socket Address

- The local (server) socket address is provided by the operating system.
- The operating system knows the IP address of the computer on which the server process is running.
- The port number of a server process needs to be assigned.
- If the server process is a standard one defined by the Internet authority, a port number is already assigned to it.
- If the server process is not standard, the designer of the server process can choose a port number and assign it to the process.
- When a server starts running, it knows the local socket address.

CLIENT-SERVER PARADIGM

Server Site

Remote Socket Address

- The remote socket address for a server is the socket address of the client that makes the connection.
- Since the server can serve many clients, it does not know beforehand the remote socket address for communication.
- The server can find this socket address when a client tries to connect to the server.
- The client socket address, which is contained in the request packet sent to the server, becomes the remote socket address that is used for responding to the client.
- In other words, although the local socket address for a server is fixed and used during its lifetime, the remote socket address is changed in each interaction with a different client.

CLIENT-SERVER PARADIGM

Client Site

- The client also needs a local (client) and a remote (server) socket address for communication.

Local Socket Address

- The local (client) socket address is also provided by the operating system. The operating system knows the IP address of the computer on which the client is running.
- The port number is a 16-bit temporary integer that is assigned to a client process each time the process needs to start the communication.
- The port number needs to be assigned from a set of integers defined by the Internet authority and called the ephemeral (temporary) port numbers.
- The operating system needs to guarantee that the new port number is not used by any other running client process.

CLIENT-SERVER PARADIGM

Client Site

- The client also needs a local (client) and a remote (server) socket address for communication.

Remote Socket Address

- Finding the remote (server) socket address for a client needs more work. When a client process starts, it should know the socket address of the server it wants to connect to.
- We will have two situations in this case.

    1. The user who starts the client process knows both the server port number and IP address of the computer on which the server is running.
        - This usually occurs in situations when we have written client and server applications and we want to test.

CLIENT-SERVER PARADIGM

Client Site

Remote Socket Address

2. Each standard application has a well-known port number, most of the time, we do not know the IP address. This happens in situations such as when we need to contact a Webpage, send an e-mail to a friend, copy a file from a remote site, and so on.

- In these situations, an identifier that uniquely defines the server process. The client process should now change this identifier.
- Examples of these identifiers are URLs, such as www.xxx.yyy, or e-mail addresses, such as xxxx@yyyy.com.
- The client process should now change this identifier (name) to the corresponding server socket address.
- The client process normally knows the port number because it should be a well-known port number, but the IP address can be obtained using another client server application called the Domain Name System (DNS). [Domain Name System, translates human readable domain names (for example, www.amazon.com) to machine readable IP addresses (for example, 192.0.2.44)]

CLIENT-SERVER PARADIGM

Using Services of the Transport Layer

- There are three common transport layer protocols in the TCP/IP suite: UDP, TCP, and SCTP(Stream Control Transmission Protocol).
- Most standard applications have been designed to use the services of one of these protocols.
- When we write a new application, we can decide which protocol we want to use.
- The choice of the transport layer protocol seriously affects the capability of the application processes.

CLIENT-SERVER PARADIGM

Using Services of the Transport Layer

UDP Protocol

- UDP provides connectionless, unreliable, datagram service.
- UDP has an advantage: it is message-oriented. It gives boundaries to the messages exchanged.
- We can compare the connectionless and unreliable service to the regular service provided by the post office.
- An application program may be designed to use UDP if it is sending small messages and the simplicity and speed is more important for the application than reliability.
- For example, some management and multimedia applications fit in this category.

CLIENT-SERVER PARADIGM

Using Services of the Transport Layer

TCP Protocol

- TCP provides connection-oriented, reliable, byte-stream service.
- By numbering the bytes exchanged, the continuity of the bytes can be checked.
- if some bytes are lost or corrupted, the receiver can request the resending of those bytes, which makes TCP a reliable protocol. TCP also can provide flow control and congestion control.
- We can compare the service provided by TCP to the service provided by the telephone company.
- Most of the standard applications that need to send long messages and require reliability may benefit from the service of the TCP.

CLIENT-SERVER PARADIGM

Using Services of the Transport Layer

SCTP Protocol

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

STANDARD CLIENT-SERVER APPLICATIONS

World Wide Web and HTTP

World Wide Web (abbreviated WWW or Web)

- The idea of the Web was first proposed by Tim Berners-Lee in 1989 at CERN, the European Organization for Nuclear Research, to allow several researchers at different locations throughout Europe to access each others' researches.
- The commercial Web started in the early 1990s.
- The Web today is a repository of information in which the documents, called Web pages, are distributed all over the world and related documents are linked together.
- The popularity and growth of the Web can be related to two terms in the above statement: distributed and linked.
- The linking of web pages was achieved using a concept called hypertext, now it is known as hypermedia.
- The purpose of the Web has gone beyond the simple retrieving of linked documents. [The Web is used to provide electronic shopping and gaming.]

STANDARD CLIENT-SERVER APPLICATIONS

World Wide Web

Architecture

- The WWW today is a distributed client-server service, in which a client using a browser can access a service using a server.
- The service provided is distributed over many locations called sites.
- Each site holds one or more documents, referred to as web pages.
- Each web page, however, can contain some links to other web pages in the same or other sites.
- In other words, a web page can be simple or composite. A simple web page has no links to other web pages; a composite web page has one or more links to other web pages.
- Each web page is a file with a name and address.

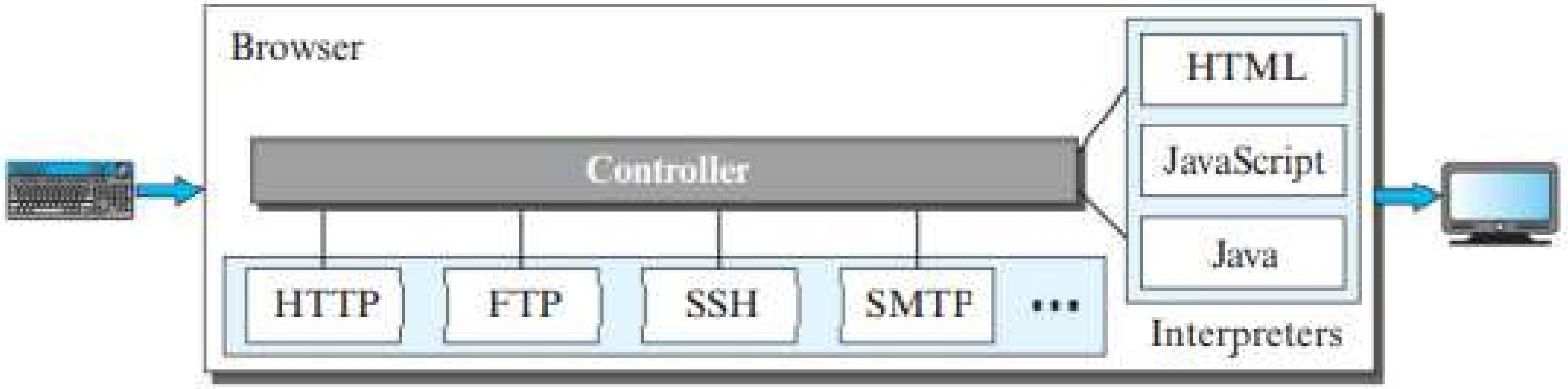
STANDARD CLIENT-SERVER APPLICATIONS

World Wide Web

Web Client (Browser)

- A variety of vendors offer commercial browsers that interpret and display a web page, and all of them use nearly the same architecture.
- Each browser usually consists of three parts: a controller, client protocols, and interpreters.

Figure 1.35 *Browser*



STANDARD CLIENT-SERVER APPLICATIONS

World Wide Web

Web Client (Browser)

- The controller receives input from the keyboard or the mouse and uses the client programs to access the document.
- After the document has been accessed, the controller uses one of the interpreters to display the document on the screen.
- The client protocol can be one of the protocols such as HTTP or FTP.
- The interpreter can be HTML, Java, or JavaScript, depending on the type of document.
- Some commercial browsers include Internet Explorer, Netscape Navigator, and Firefox.

STANDARD CLIENT-SERVER APPLICATIONS

World Wide Web

Web Server

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

STANDARD CLIENT-SERVER APPLICATIONS

World Wide Web

Uniform Resource Locator (URL) -Web Address

- A web page, as a file, needs to have a unique identifier to distinguish it from other web pages.
- To define a web page, we need three identifiers: host, port, and path.
- Before defining the web page, we need to tell the browser what client server application we want to use, which is called the protocol.
- This means we need four identifiers to define the web page. The first is the type of vehicle to be used to fetch the web page; the last three make up the combination that defines the destination object (web page).

STANDARD CLIENT-SERVER APPLICATIONS

World Wide Web

- Protocol:
  - The first identifier is the abbreviation for the client-server program that we need in order to access the web page.
  - Although most of the time the protocol is HTTP (HyperText Transfer Protocol), we can also use other protocols such as FTP (File Transfer Protocol).

- Host:
  - The host identifier can be the IP address of the server or the unique name given to the server.
  - IP addresses can be defined in dotted decimal notations (such as 64.23.56.17); the name is normally the domain name that uniquely defines the host, such as forouzan.com.

STANDARD CLIENT-SERVER APPLICATIONS

World Wide Web

- Port:
  - The port, a 16-bit integer, is normally predefined for the client-server application.
  - For example, if the HTTP protocol is used for accessing the web page, the well-known port number is 80.
- Path:
  - The path identifies the location and the name of the file in the underlying operating system.
  - The format of this identifier normally depends on the operating system.
  - In UNIX, a path is a set of directory names followed by the file name, all separated by a slash.
  - For example, /top/next/last/myfile is a path that uniquely defines a file named myfile, stored in the directory last, which itself is part of the directory next, which itself is under the directory top.

STANDARD CLIENT-SERVER APPLICATIONS

World Wide Web

To combine these four pieces together, the uniform resource locator (URL) has been designed; it uses three different separators between the four pieces as shown below:

protocol://host/path Used most of the time

protocol://host:port/path Used when port number is needed

- Example: The URL http://www.mhhe.com/compsci/forouzan/ defines the web page.
- The string www.mhhe.com is the name of the computer in the McGraw-Hill company (the three letters www are part of the host name and are added to the commercial host).
- The path is compsci/forouzan/, which defines Forouzan's web page under the directory compsci (computer science).

STANDARD CLIENT-SERVER APPLICATIONS

World Wide Web

Web Documents

- The documents in the WWW can be grouped into three broad categories: static, dynamic, and active.

Static Documents

- Static documents are fixed-content documents that are created and stored in a server. Static documents are prepared using one of the several languages: Hypertext Markup Language (HTML), Extensible Markup Language (XML), Extensible Style Language (XSL), and Extensible Hypertext Markup Language (XHTML).

STANDARD CLIENT-SERVER APPLICATIONS

World Wide Web

Web Documents

- The documents in the WWW can be grouped into three broad categories: static, dynamic, and active.

Dynamic Documents

- A dynamic document is created by a web server whenever a browser requests the document.
- Although the Common Gateway Interface (CGI) was used to retrieve a dynamic document in the past, today's options include one of the scripting languages such as Java Server Pages (JSP), which uses the Java language for scripting, or Active Server Pages (ASP), a Microsoft product that uses Visual Basic language for scripting, or ColdFusion, which embeds queries in a Structured Query Language (SQL) database in the HTML document.

STANDARD CLIENT-SERVER APPLICATIONS

World Wide Web

Web Documents

- The documents in the WWW can be grouped into three broad categories: static, dynamic, and active.

Active Documents

- For many applications, we need a program or a script to be run at the client site. These are called active documents.
- For example, suppose we want to run a program that creates animated graphics on the screen or a program that interacts with the user. The program definitely needs to be run at the client site where the animation or interaction takes place.
- When a browser requests an active document, the server sends a copy of the document or a script. The document is then run at the client (browser) site.
- One way to create an active document is to use Java applets, a program written in Java on the server. Another way is to use JavaScripts.

STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

- The HyperText Transfer Protocol (HTTP) is a protocol that is used to define how the client-server programs can be written to retrieve web pages from the Web.
- An HTTP client sends a request; an HTTP server returns a response.
- The server uses the port number 80; the client uses a temporary port number.
- HTTP uses the services of TCP, which is a connection-oriented and reliable protocol.
- The client and server do not need to worry about errors in messages exchanged or loss of any message, because the TCP is reliable and will take care of this matter.

STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

Nonpersistent versus Persistent Connections

- The hypertext concept embedded in web page documents may require several requests and responses.
- If the web pages, objects to be retrieved, are located on different servers, we do not have any other choice than to create a new TCP connection for retrieving each object.
- If some of the objects are located on the same server, we have two choices: to retrieve each object using a new TCP connection or to make a TCP connection and retrieve them all.
- The first method is referred to as nonpersistent connections, the second as persistent connections.
- HTTP specified nonpersistent connections, while persistent connections is the default in version 1.1, but it can be changed by the user.

STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

Nonpersistent Connections

- In a nonpersistent connection, one TCP connection is made for each request/response.
- The following lists the steps in this strategy:

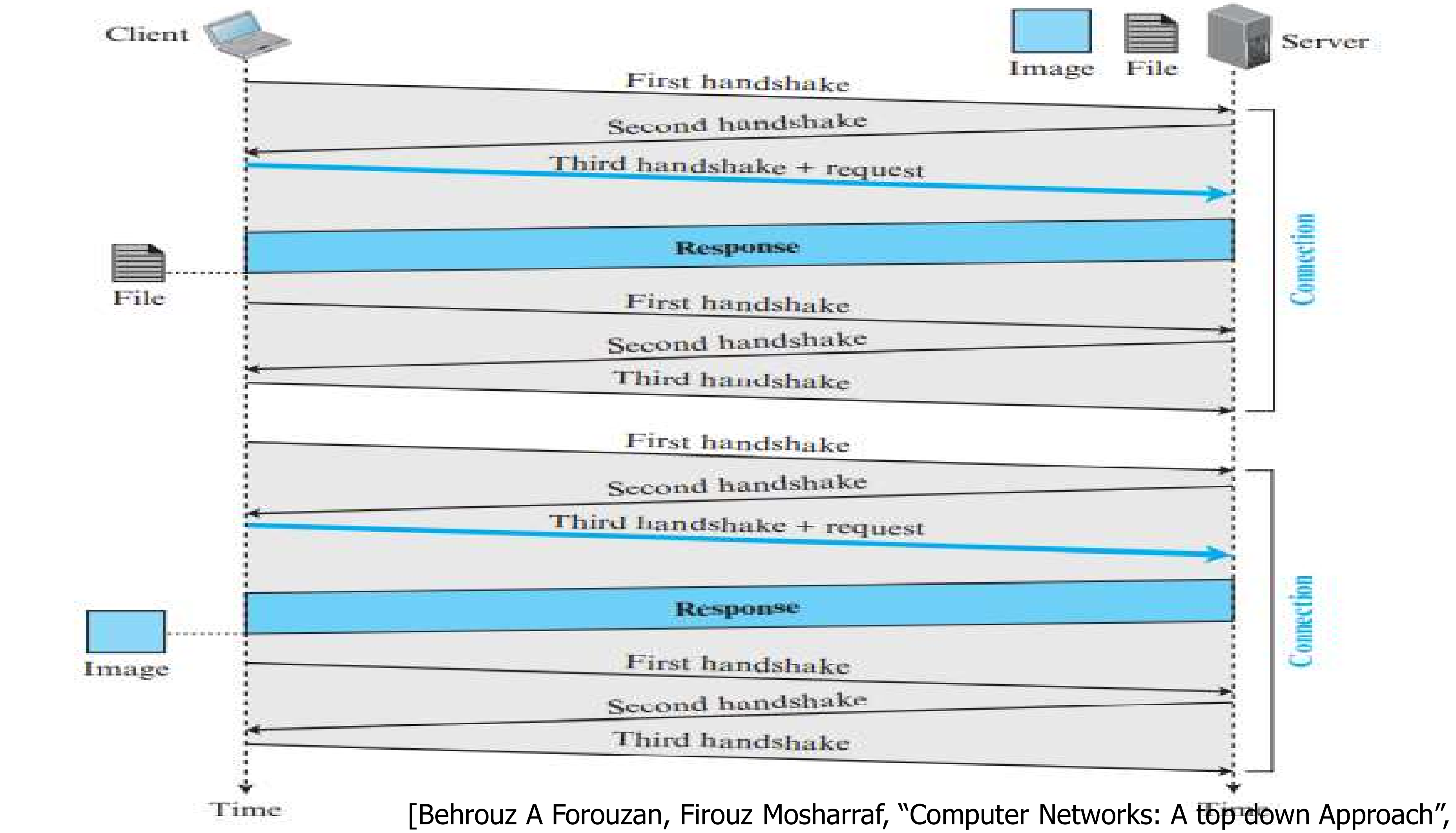
  1. The client opens a TCP connection and sends a request.
  2. The server sends the response and closes the connection.
  3. The client reads the data until it encounters an end-of-file marker; it then closes the connection.

- In this strategy, if a file contains links to N different pictures in different files (all located on the same server), the connection must be opened and closed N + 1 times.
- The nonpersistent strategy imposes high overhead on the server because the server needs N + 1 different buffers each time a connection is opened.

STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

Figure 1.36 *Example* Nonpersistent Connections



[Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top-down Approach", McGraw Hill Education]

- The client needs to access a file that contains one link to an image.

STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

- Figure 1.36 shows an example of a nonpersistent connection.
- The client needs to access a file that contains one link to an image. The text file and image are located on the same server.
- Here we need two connections.
- For each connection, TCP requires at least three handshake messages to establish the connection.
- After the connection is established, the object can be transferred.
- After receiving an object, another three handshake messages are needed to terminate the connection.
- This means that the client and server are involved in two connection establishments and two connection terminations.
- If the transaction involves retrieving 10 or 20 objects, the round trip times spent for these handshakes add up to a big overhead

Persistent Connections

- HTTP version 1.1 specifies a persistent connection by default.
- In a persistent connection, the server leaves the connection open for more requests after sending a response.
- The server can close the connection at the request of a client or if a time-out has been reached.
- The sender usually sends the length of the data with each response.

STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

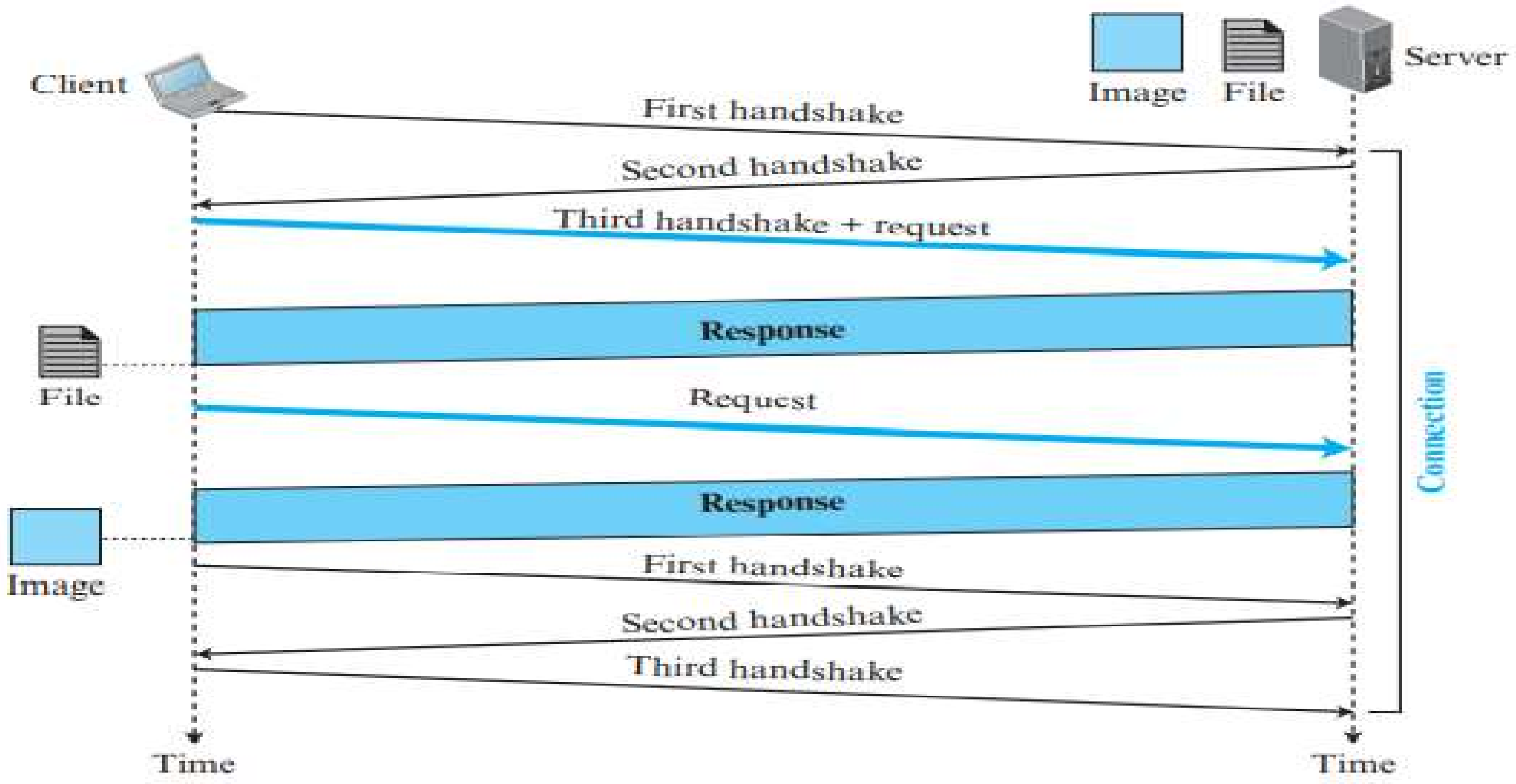
Persistent Connections

- There are some occasions when the sender does not know the length of the data. This is the case when a document is created dynamically or actively.
- In these cases, the server informs the client that the length is not known and closes the connection after sending the data so the client knows that the end of the data has been reached.
- Time and resources are saved using persistent connections.
- Only one set of buffers and variables needs to be set for the connection at each site.
- The round trip time for connection establishment and connection termination is saved.

STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

Figure 1.37 *Example* Persistent Connections



[Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top down Approach", McGraw Hill Education]

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

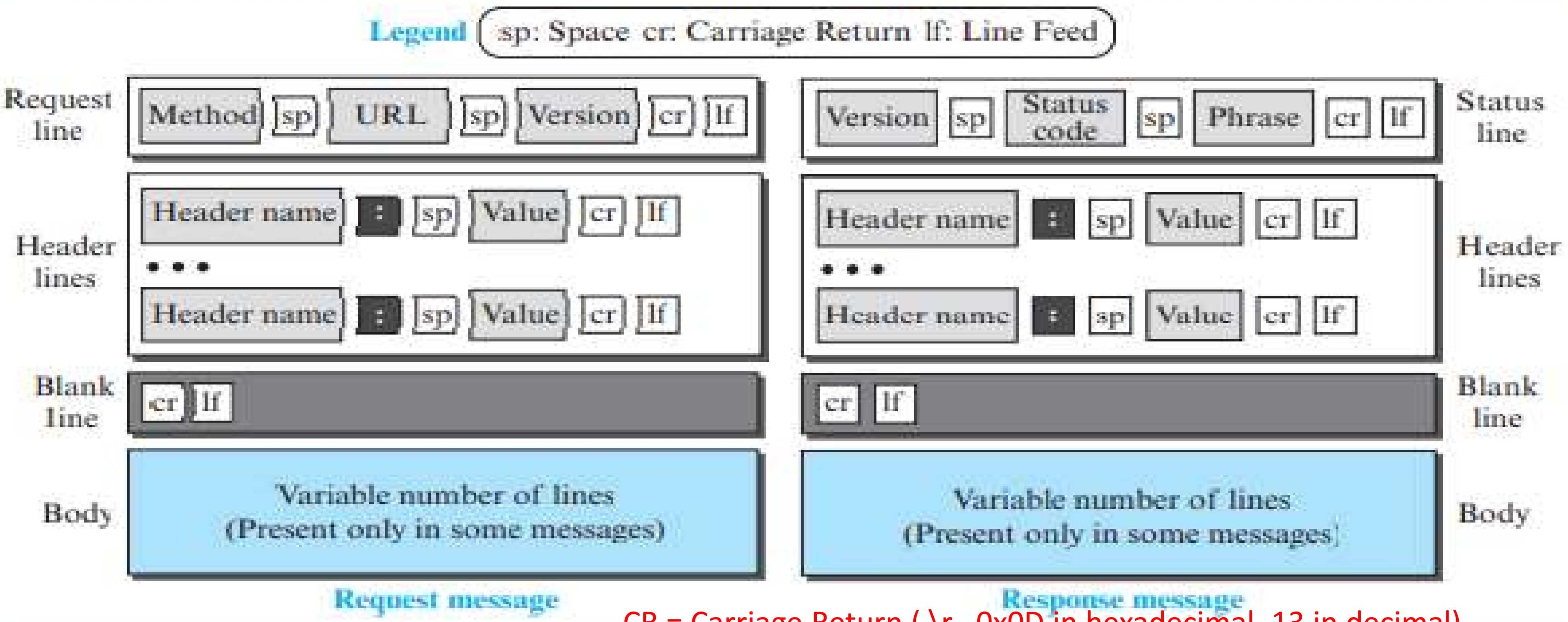
STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

Message Formats

- The HTTP protocol defines the format of the request and response messages.

Figure 1.38 *Formats of the request and response messages*



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
Status line
Version sp Status code sp Phrase 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)
Response message

CR = Carriage Return (\r , 0x0D in hexadecimal, 13 in decimal)

LF = Line Feed (\n , 0x0A in hexadecimal, 10 in decimal)

[Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top down Approach", McGraw Hill Education]

STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

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.

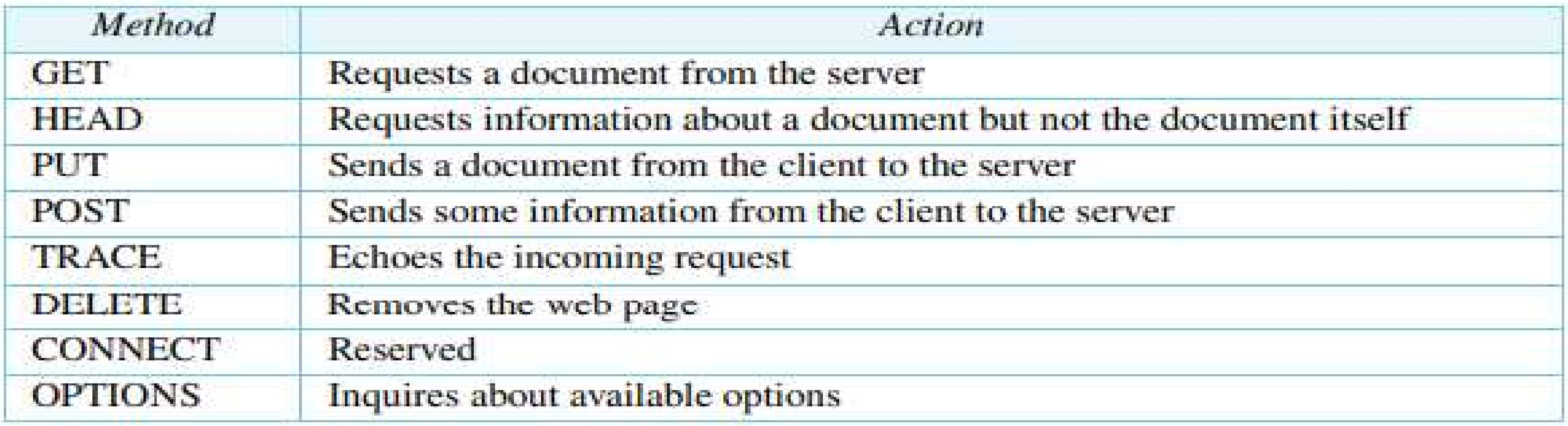
STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

Request Message

- The first line in a request message is called a request line.
- There are three fields in this line separated by one space and terminated by two characters (carriage return and line feed).
- The fields are called method, URL, and version.
- The method field defines the request types.
- In version 1.1 of HTTP, several methods are defined, as shown in Table 1.2.

Table 1.2 *Methods*



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

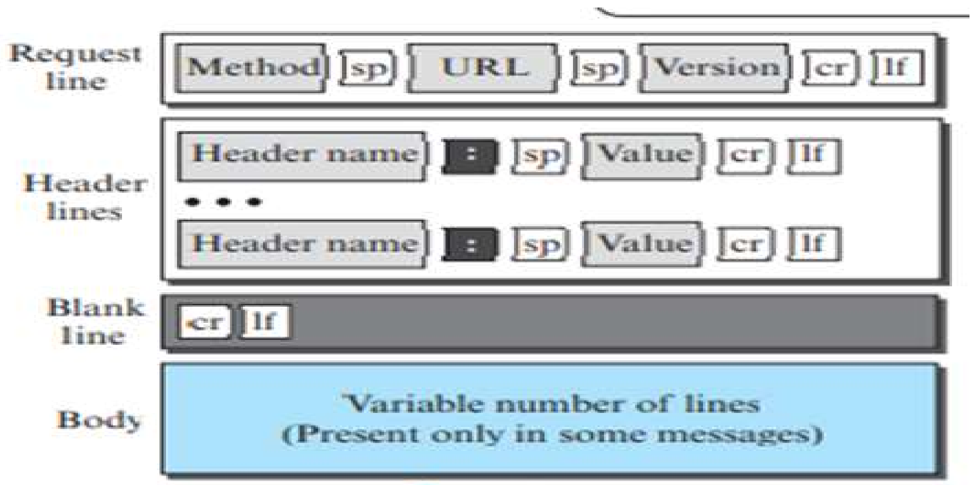
[Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top down Approach", McGraw Hill Education]

STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

Request Message

- The second field, URL defines the address and name of the corresponding web page.
- The third field, version, gives the version of the protocol; the most current version of HTTP is 1.1. (HTTP has four versions — HTTP/0.9, HTTP/1.0, HTTP/1.1, and HTTP/2.0. Today the version in common use is HTTP/1.1 and the future will be HTTP/2.0.) [HTTP/3]



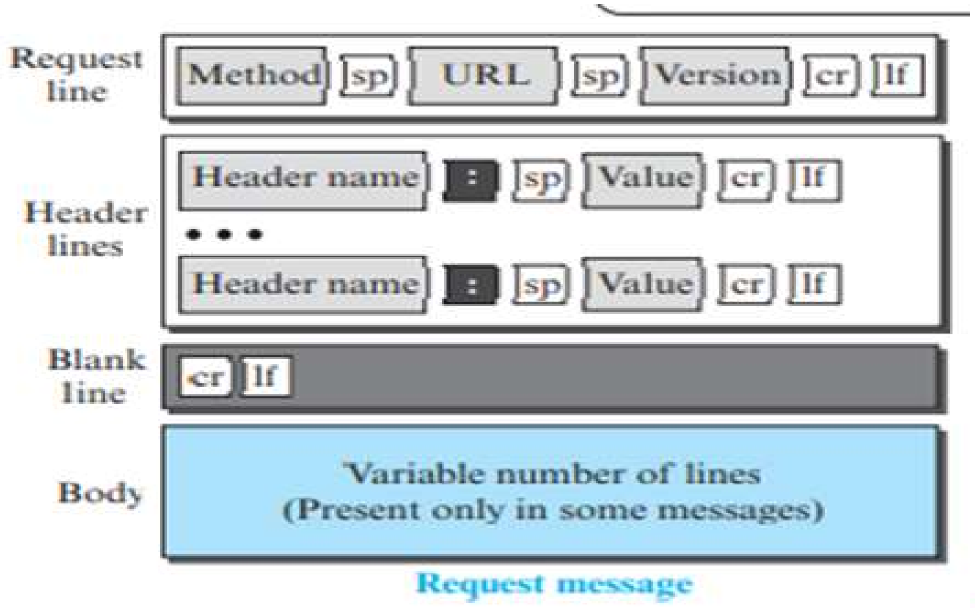
Request message

STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

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.
- Each header line has a header name, a colon, a space and a header value.



STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

Request Message

- Table 1.3 shows some header names commonly used in a request.
- The value field defines the values associated with each header name.
- The list of values can be found in the corresponding RFCs.
- The body can be present in a request message. Usually, it contains the comment to be sent or the file to be published on the website when the method is PUT or POST.

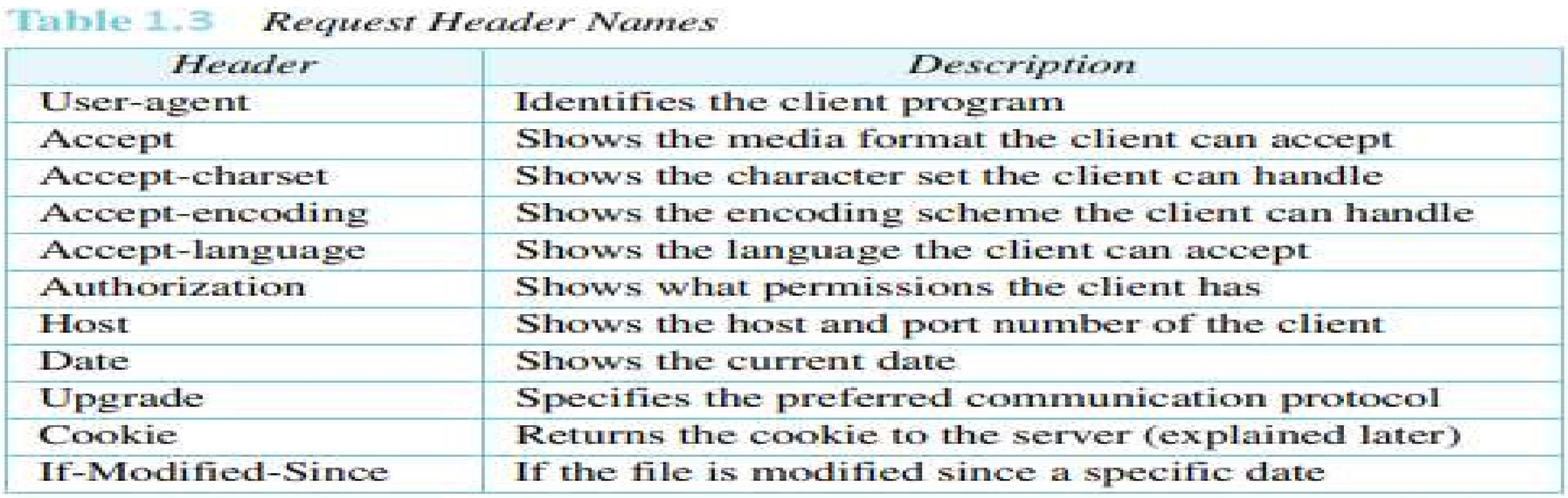


Table 1.3 *Request Header Names*

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

[Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top down Approach", McGraw Hill Education]

STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

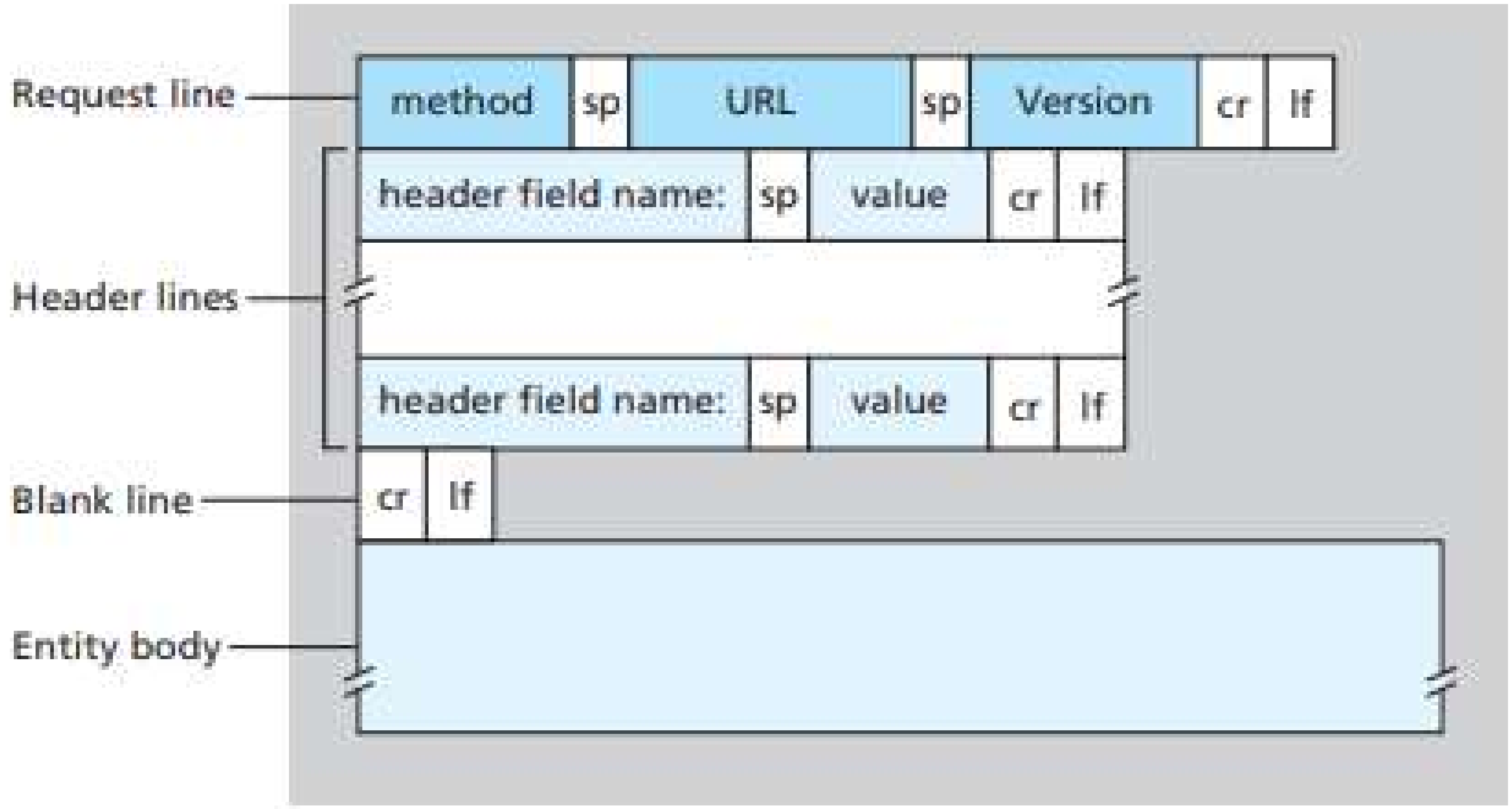


Figure 1.39 General format of an HTTP request message

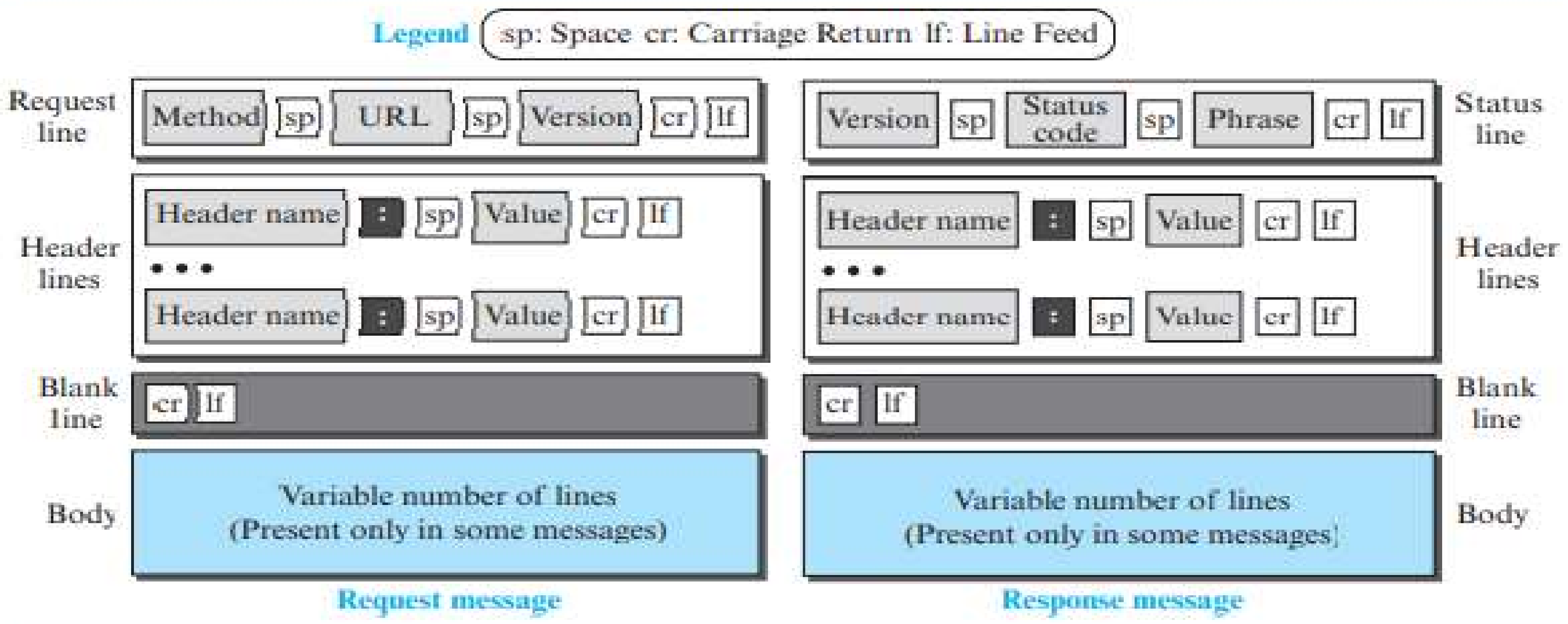
STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

Response Message

The format of the response message is also shown in Figure 1.38.

Figure 1.38 *Formats of the request and response messages*



[Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top down Approach", McGraw Hill Education]

STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

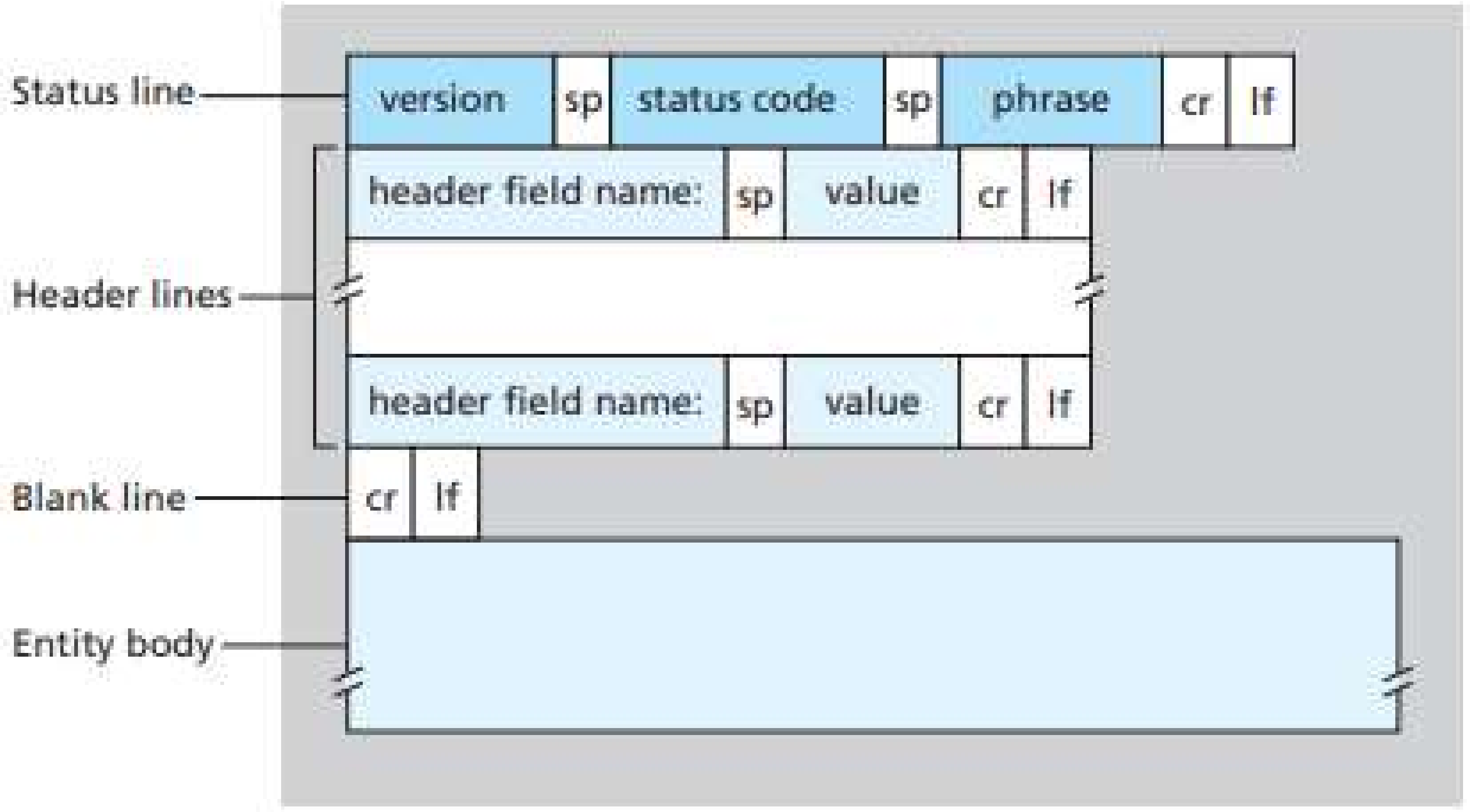


Figure 1.40 General format of an HTTP response message

HyperText Transfer Protocol (HTTP)

Response Message

- A response message consists of a status line, header lines, a blank line, and sometimes a body.
- 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.

STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

Response Message

- The status code field defines the status of the request. 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.

STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

Response Message

- After the status line, we can have zero or more response header lines.
- Each header line sends additional information from the server to the client.
- For example, the sender can send extra information about the document.
- Each header line has a header name, a colon, a space, and a header value.

STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

Response Message

- Table 1.4 shows some header names commonly used in a response message.
- The body contains the document to be sent from the server to the client.
- The body is present unless the response is an error message.

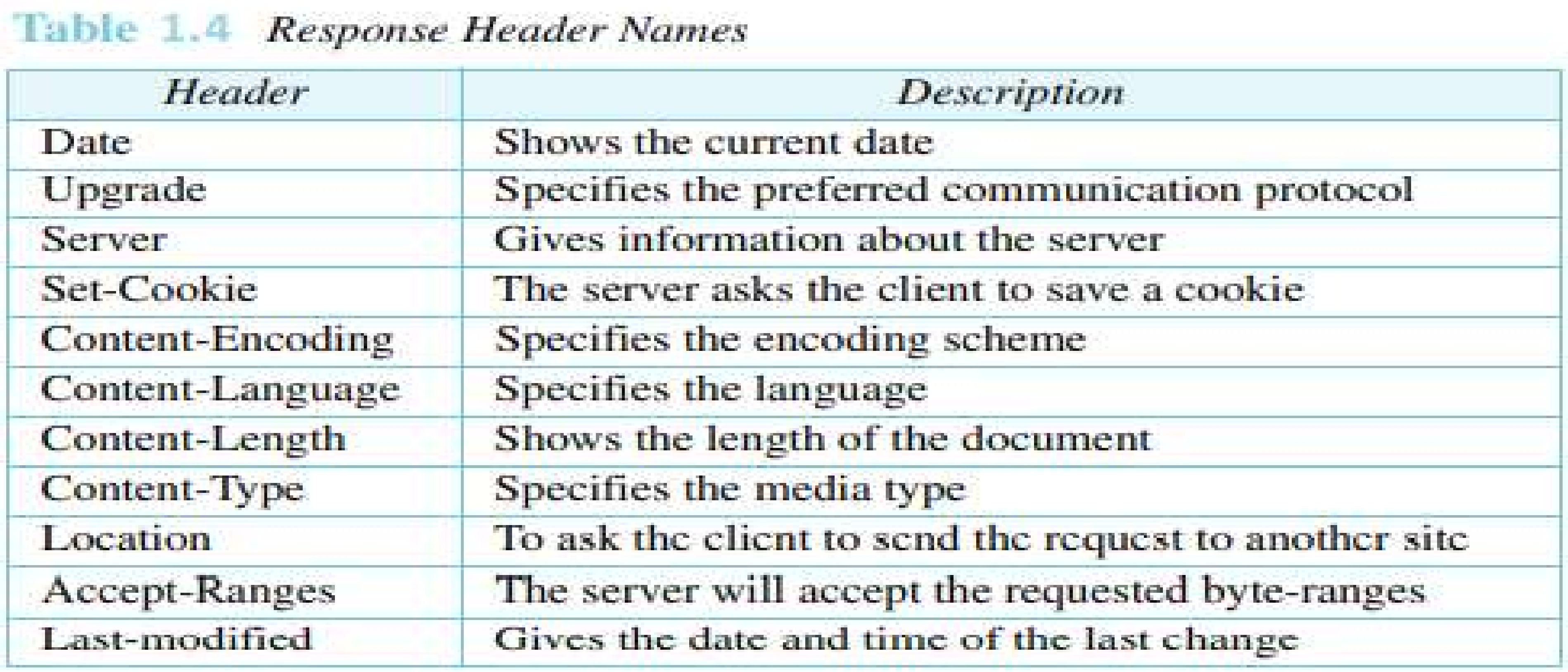


Table 1.4 *Response Header Names*

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

[Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top down Approach", McGraw Hill Education]

STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

Question

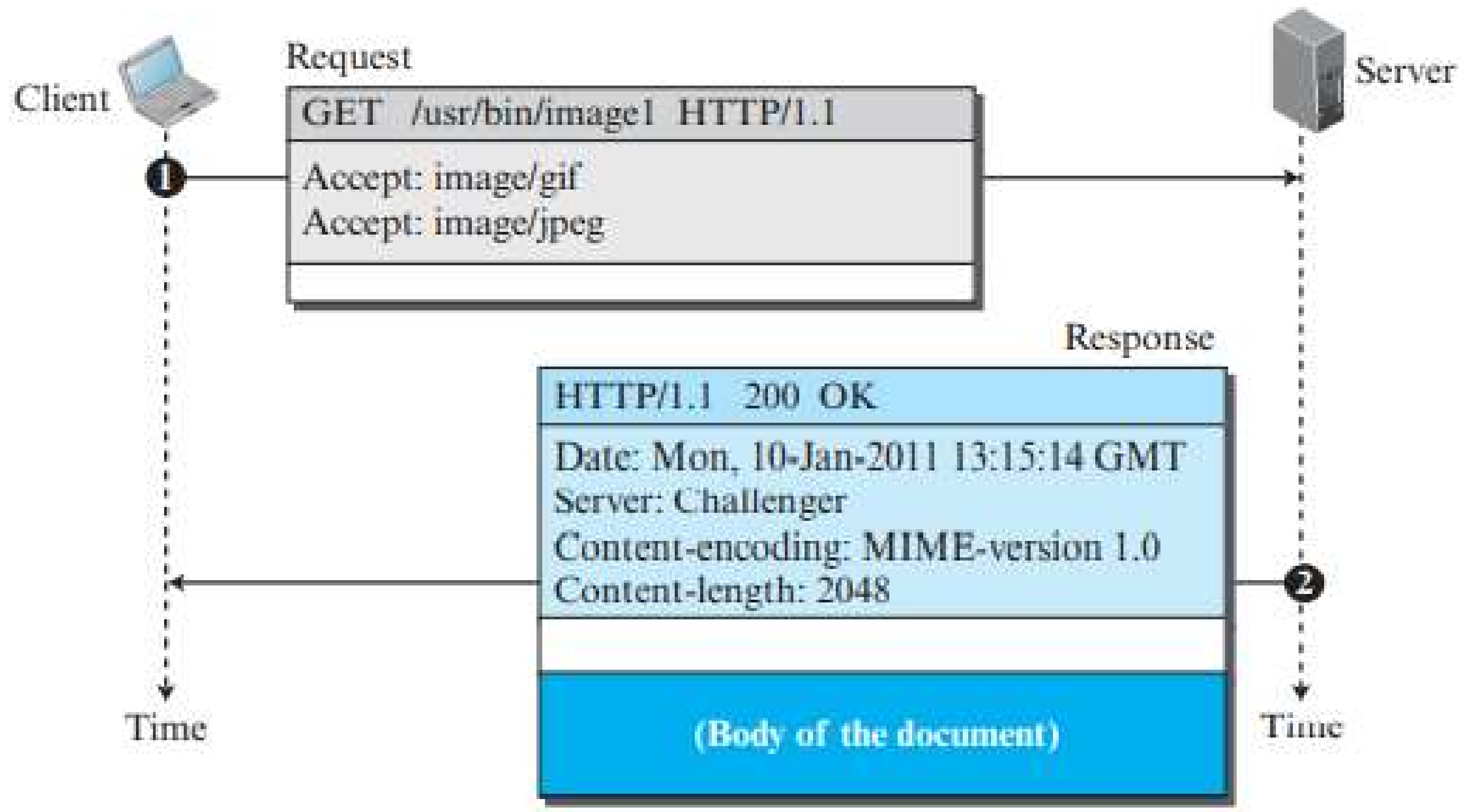
- How does a client retrieve of a document using HTTP with the help of an example.

STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

- An example retrieves a document (see Figure 1.41)

Figure1.41 *Example*



```
GET   /usr/bin/image1  HTTP/1.1
Accept: image/gif
Accept: image/jpeg
```

```
HTTP/1.1  200  OK
Date: Mon, 10-Jan-2011 13:15:14 GMT
Server: Challenger
Content-encoding: MIME-version 1.0
Content-length: 2048
```

[Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top down Approach", McGraw Hill Education]

STANDARD CLIENT-SERVER APPLICATIONS

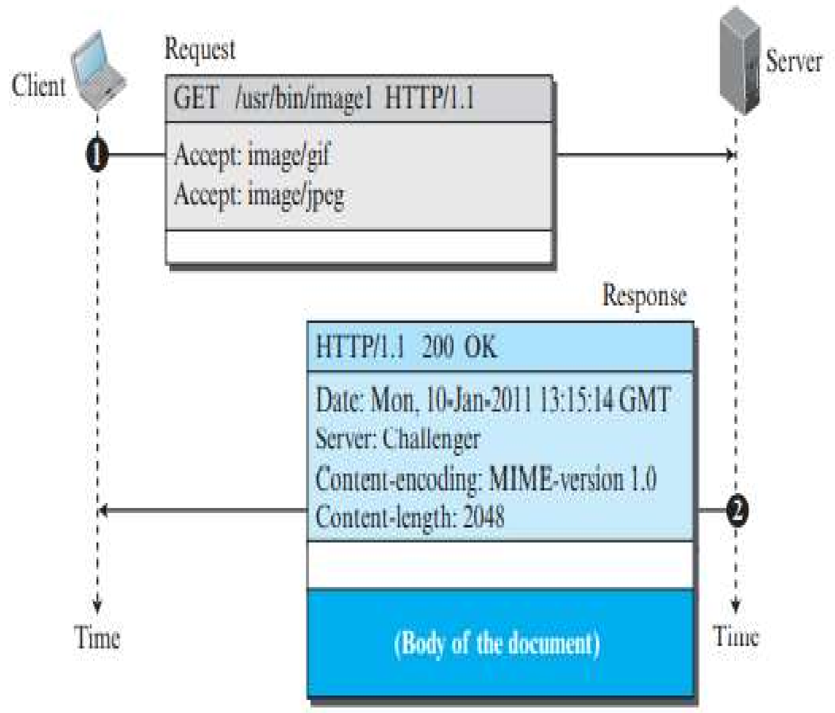
HyperText Transfer Protocol (HTTP)

- An example retrieves a document (see Figure 1.41)

We use the GET method to retrieve an image with the path /usr/bin/image1. The request line shows the method (GET), the URL, and the HTTP version (1.1). The header has two lines that show that the client can accept images in the GIF or JPEG format. The request does not have a body.

The response message contains the status line and four lines of header. The header lines define the date, server, content encoding (MIME version, which will be described in electronic mail), and length of the document. The body of the document follows the header.

Figure1.41 *Example*



```
GET  /usr/bin/image1  HTTP/1.1
Accept: image/gif
Accept: image/jpeg
```

```
HTTP/1.1  200  OK
Date: Mon, 10-Jan-2011 13:15:14 GMT
Server: Challenger
Content-encoding: MIME-version 1.0
Content-length: 2048
```

[Multipurpose Internet Mail Extensions (MIME) is an Internet standard that extends the format of email messages]

[Behrouz A Forouzan, Firouz Mosharraf, “Computer Networks: A top down Approach”, McGraw Hill Education]

STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

Question

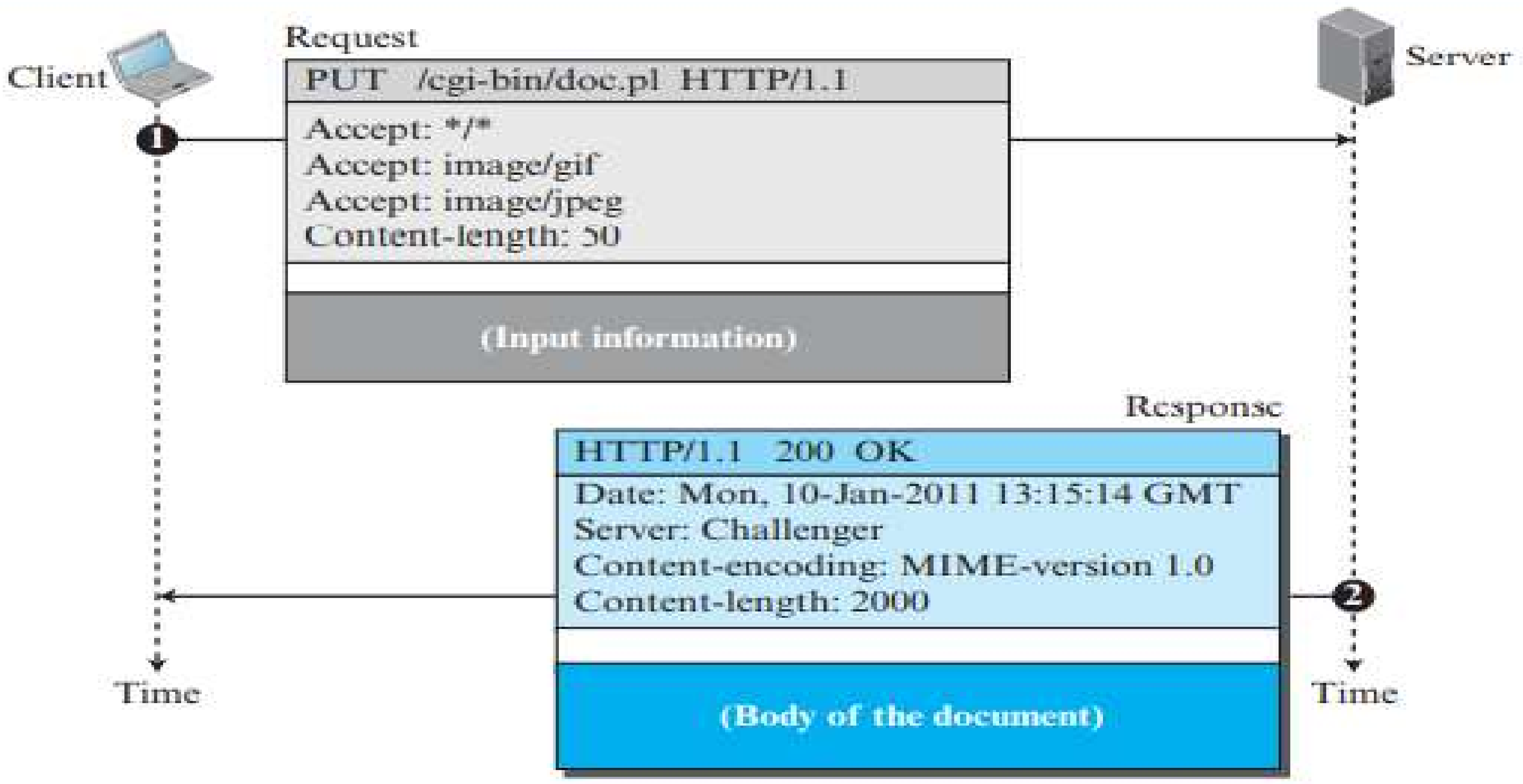
- How does a client send a web page that has to be posted on the server with the help of an example.

STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

- The client wants to send a web page to be posted on the server(see Figure 1.42)

Figure 1.42 *Example*



```
PUT   /cgi-bin/doc.pl  HTTP/1.1
Accept: */*
Accept: image/gif
Accept: image/jpeg
Content-length: 50
```

```
HTTP/1.1  200  OK
Date: Mon, 10-Jan-2011 13:15:14 GMT
Server: Challenger
Content-encoding: MIME-version 1.0
Content-length: 2000
```

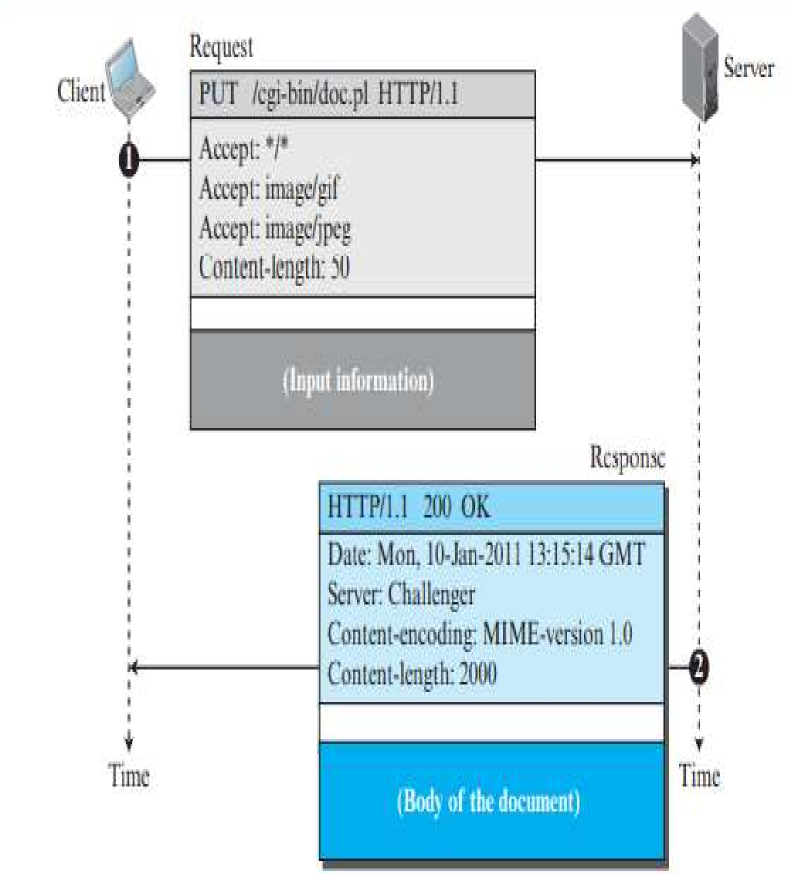
[Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top down Approach", McGraw Hill Education]

STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

- The client wants to send a web page to be posted on the server(see Figure 1.42)

Figure 1.42 *Example*



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.

[A **CGI file** is known as a Common Gateway Interface script that is used by a web server to run an external program to process user requests.]

[Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top down Approach", McGraw Hill Education]

STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

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 like:
  - 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. For these purposes, the cookie mechanism was devised.

STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

Cookies

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.

STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

Cookies

Using Cookies

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

STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

Web Caching: Proxy Server

- HTTP supports proxy servers.
- A proxy server is a computer that keeps copies of responses to recent requests.
- The HTTP client sends a request to the proxy server.
- The proxy server checks its cache.
- If the response is not stored in the cache, the proxy server sends the request to the corresponding server.
- Incoming responses are sent to the proxy server and stored for future requests from other clients.
- The proxy server reduces the load on the original server, decreases traffic, and improves latency.
- However, to use the proxy server, the client must be configured to access the proxy instead of the target server. Note that the proxy server acts as both server and client

STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

Web Caching: Proxy Server

Proxy Server Location

- The proxy servers are normally located at the client site.
- This means that we can have a hierarchy of proxy servers as shown below:

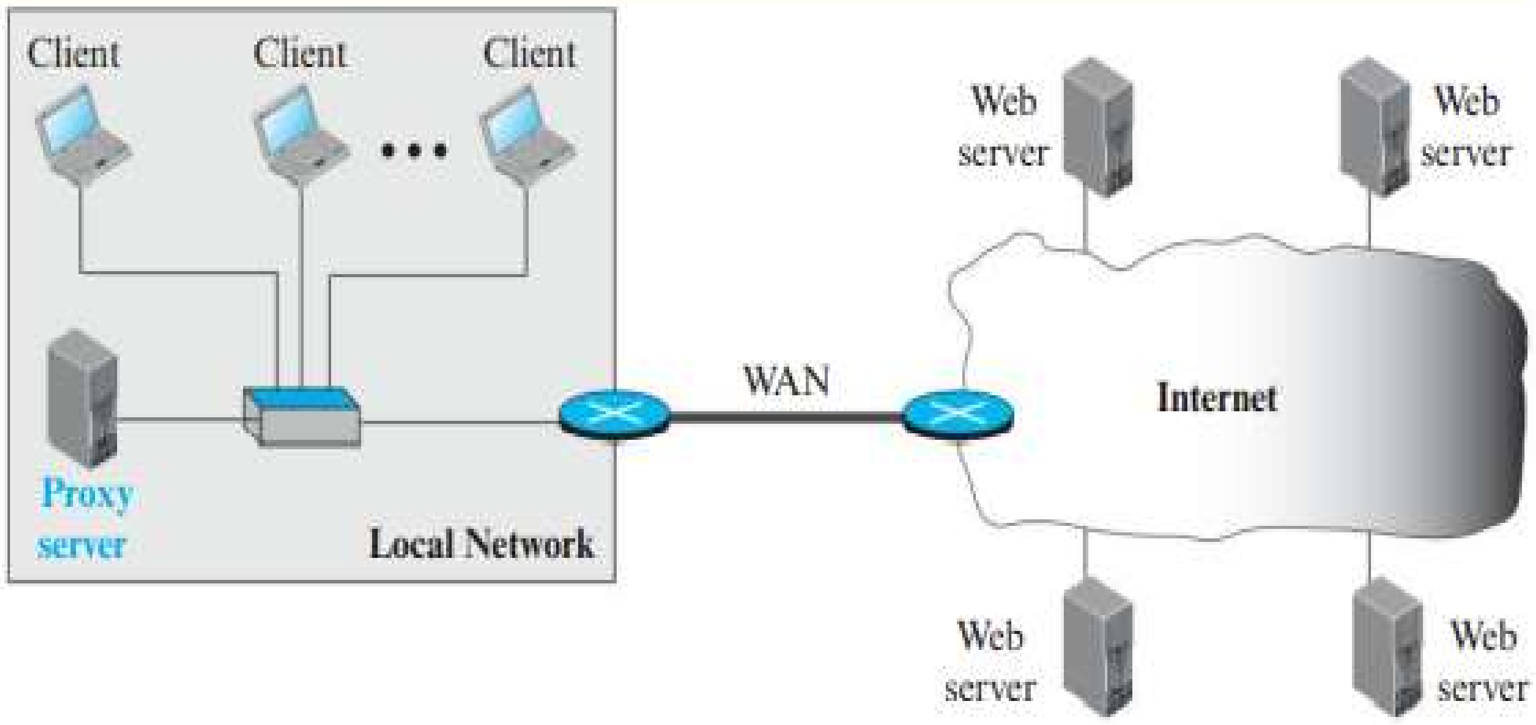
  1. A client computer can also be used as a proxy server, in a small capacity, that stores responses to requests often invoked by the client.
  2. In a company, a proxy server may be installed on the computer LAN to reduce the load going out of and coming into the LAN.
  3. An ISP with many customers can install a proxy server to reduce the load going out of and coming into the ISP network.

STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

Web Caching: Proxy Server

Figure 1.43 *Example of a proxy server*



[Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top down Approach", McGraw Hill Education]

STANDARD CLIENT-SERVER APPLICATIONS

HyperText Transfer Protocol (HTTP)

HTTP Security

- HTTP per se does not provide security.
- However, HTTP can be run over the Secure Socket Layer (SSL).
- In this case, HTTP is referred to as HTTPS.