TCP/IP Protocol Suite: Application Layer

Faculty: Dr. Asif Uddin Khan
Assistant Professor
KIIT University

Acknowledgement

- These slides are taken (as it is or with some modifications) from various resources including -
  - Computer Networking: A top-down approach (Book and Slides) by Jim Kurose and Keith Ross
  - Data Communication and Networking with TCP/IP Protocol Suite (Book and Slides) by Behrouz A. Forouzan

What we have studied so far...

- What is a Computer Network?
- Network Criteria
- Network Topologies
- Types of Computer Networks
- Structure of the Internet
- Circuit Switching
- Packet Switching
- Delays in Packet Switched Networks
- Throughput
- TCP / IP Protocol Suite
- OSI Model
- Analog and Digital Signals
- Attributes of Analog Signals
- Composite Signals
- Attributes of Digital Signals
- Transmission of Digital Signals
- Multiplexing
- Digital Transmission
  - Digital to Digital Conversion
  - Analog to Digital Conversion
- Analog Tranmission
  - Digital to Analog Conversion
  - Analog to Analog Conversion

Application layer

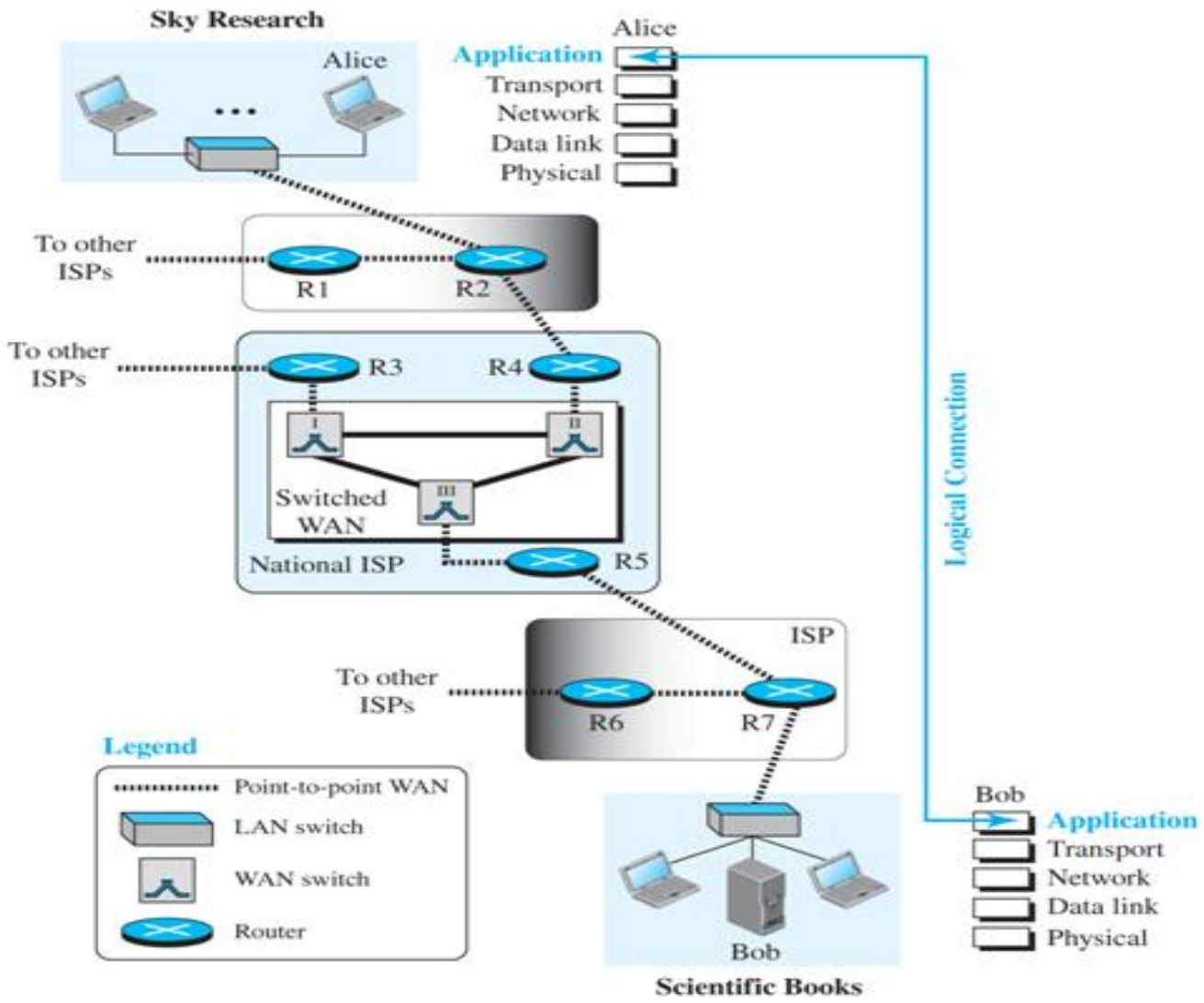


Figure 10.1 Logical connection at the application layer

- 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 10.1 shows the idea behind this logical connection.

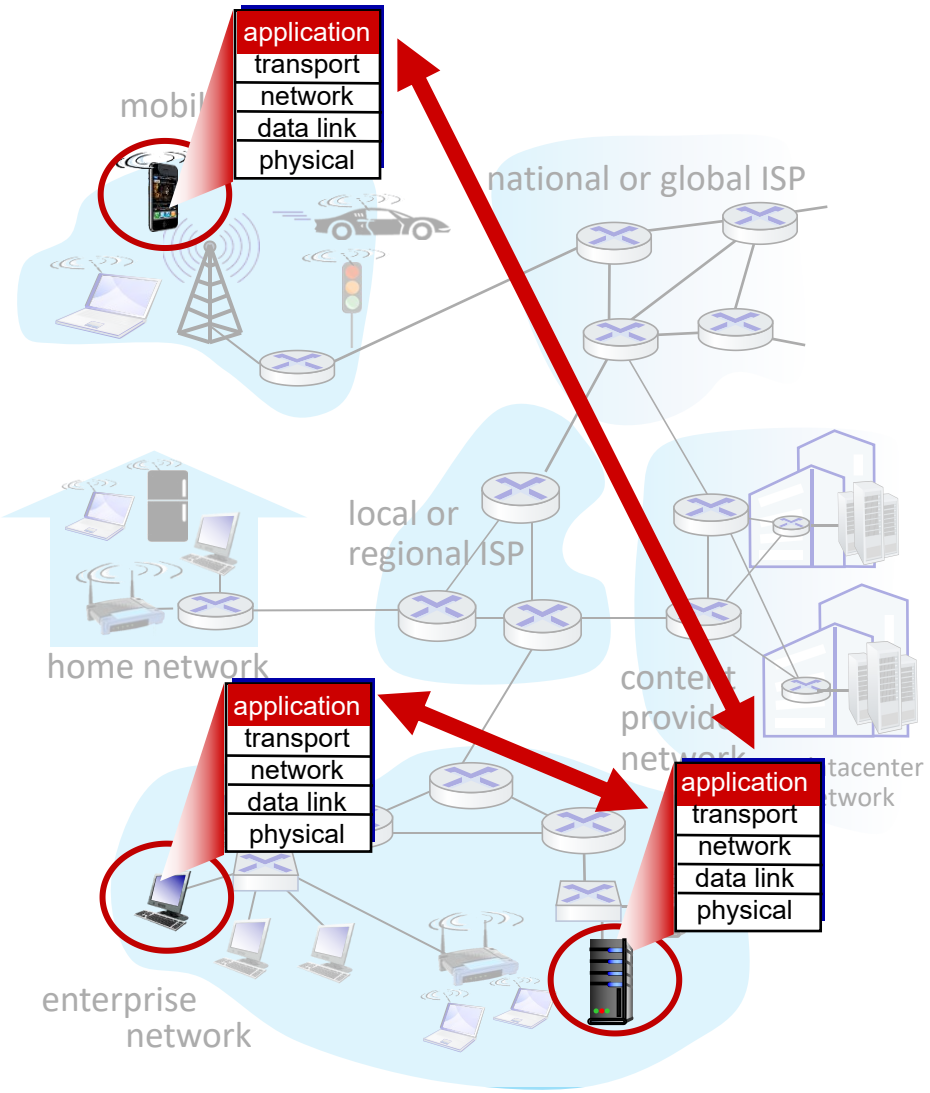
Creating a network app

write programs that:

- run on (different) end systems
- communicate over network
- e.g., web server software communicates with browser software

no need to write software for network-core devices

- network-core devices do not run user applications
- applications on end systems allows for rapid app development, propagation



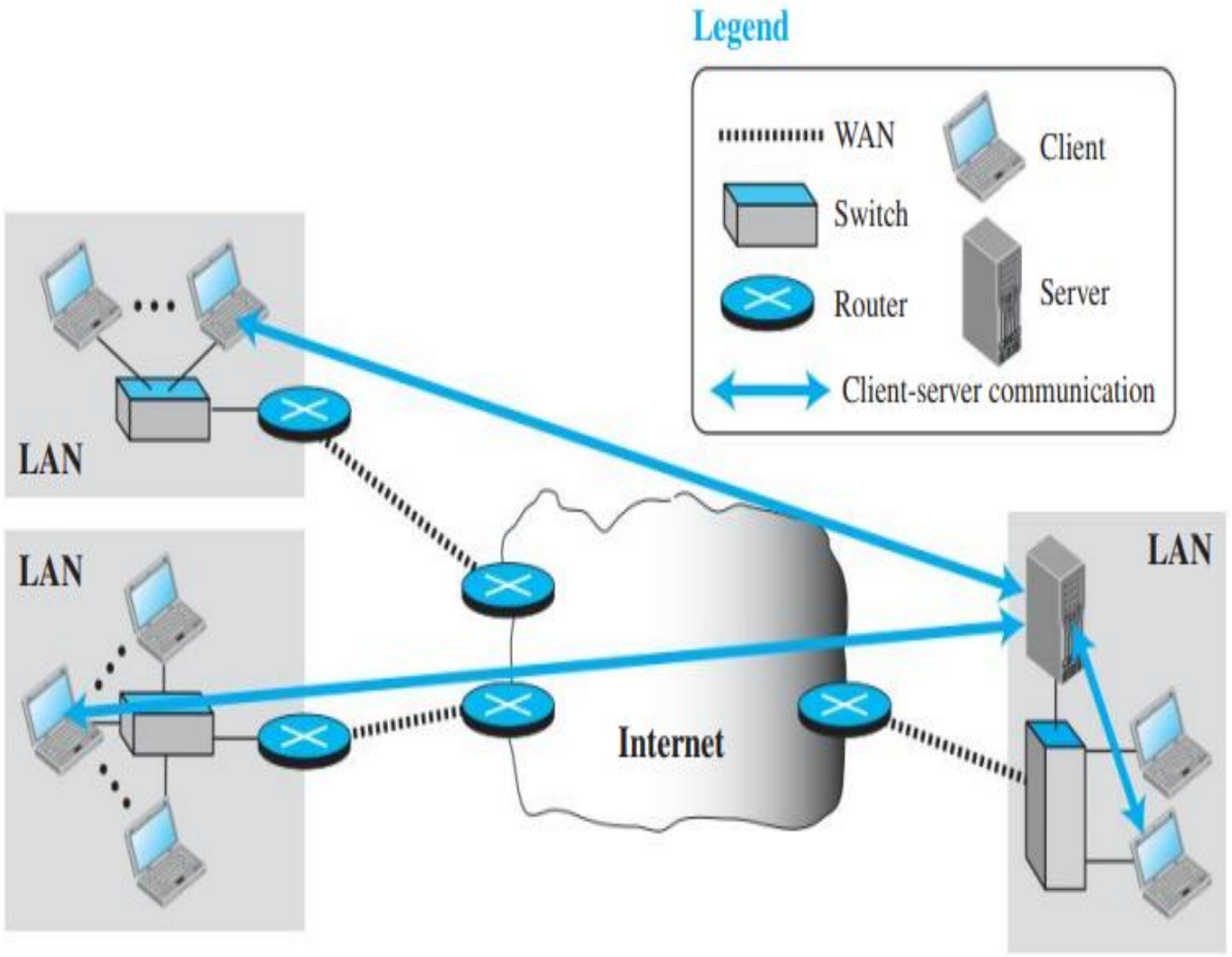
Providing Services

- The application layer is highest layer in protocol suite and its services only depends upon protocols in transport layer.
- Application layer is the only layer that provides services to the Internet user.
- The flexibility of the application layer allows new application protocols to be easily added to the Internet.
- New protocols can be added and removed from this layer easily as long as new added protocols works with protocols provided by transport layer.
- When the Internet was created, only a few application protocols were available to the users; today we cannot give a number for these protocols because new ones are being added constantly.

Application Layer Paradigms

- Defines the relationship between two application programs which are running on two different computer to interact with each other.
- Three types of paradigms used between two devices for communication in application layer
  - ✓Client - Server Paradigm
  - ✓Peer to Peer Paradigm
  - ✓Mixed Paradigm

Client - Server Paradigm



- It is a traditional paradigm that uses two running application programs, called the server process and the client process, to make a connection through the Internet and ask for service.
- A client initializes the communication by sending a request; a server 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 means that server must be running at all time but client only need to run when it requires services.

CLIENT - SERVER PARADIGM

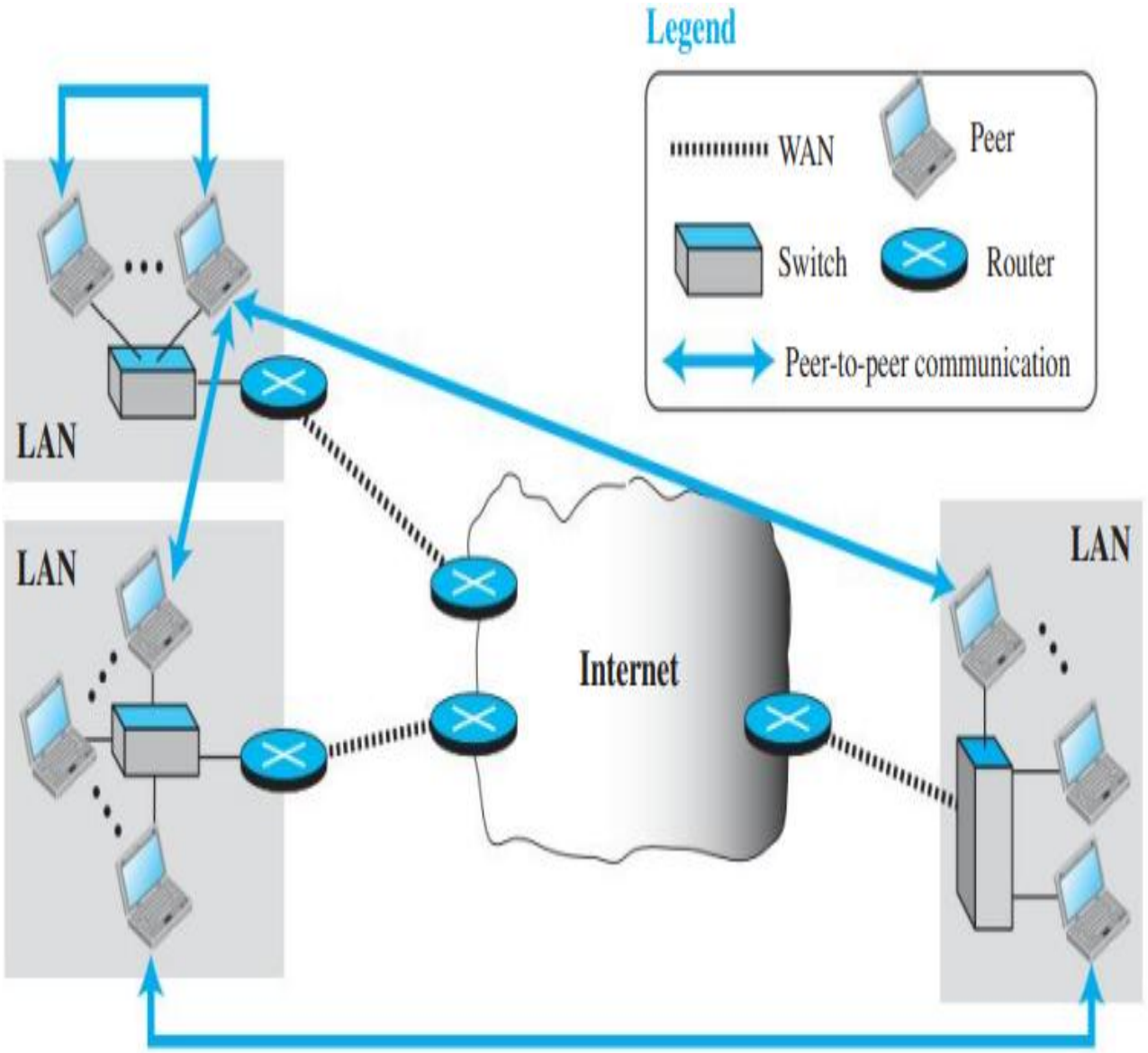
Advantages -

- 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

- If server is infected with virus then client might also get affected.
- Server are prone to Denial of Service (DoS) attacks.
- Servers have to handle most of the workload which increases as the number of clients increases.

New Paradigm: Peer-to-Peer Paradigm

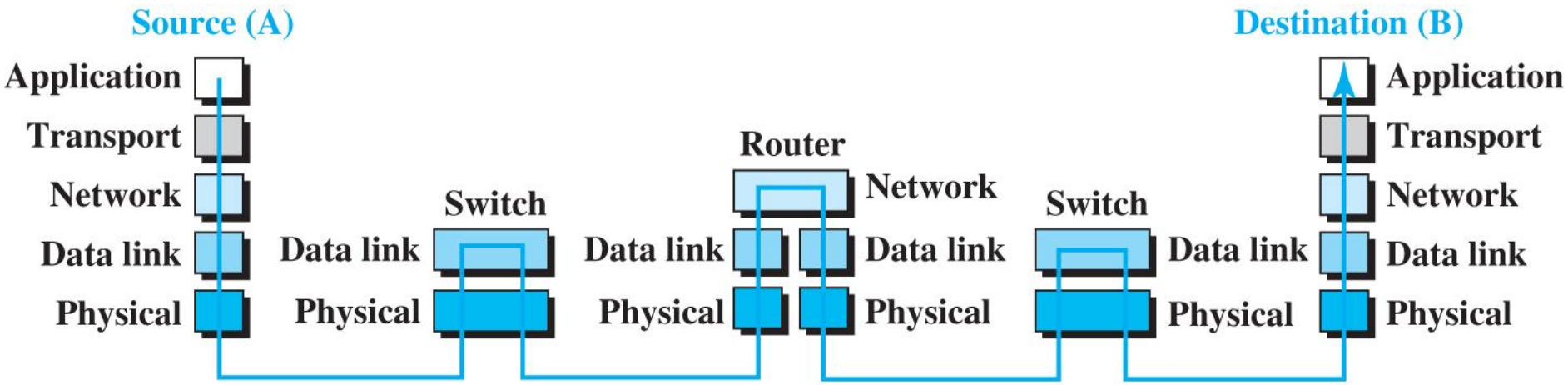


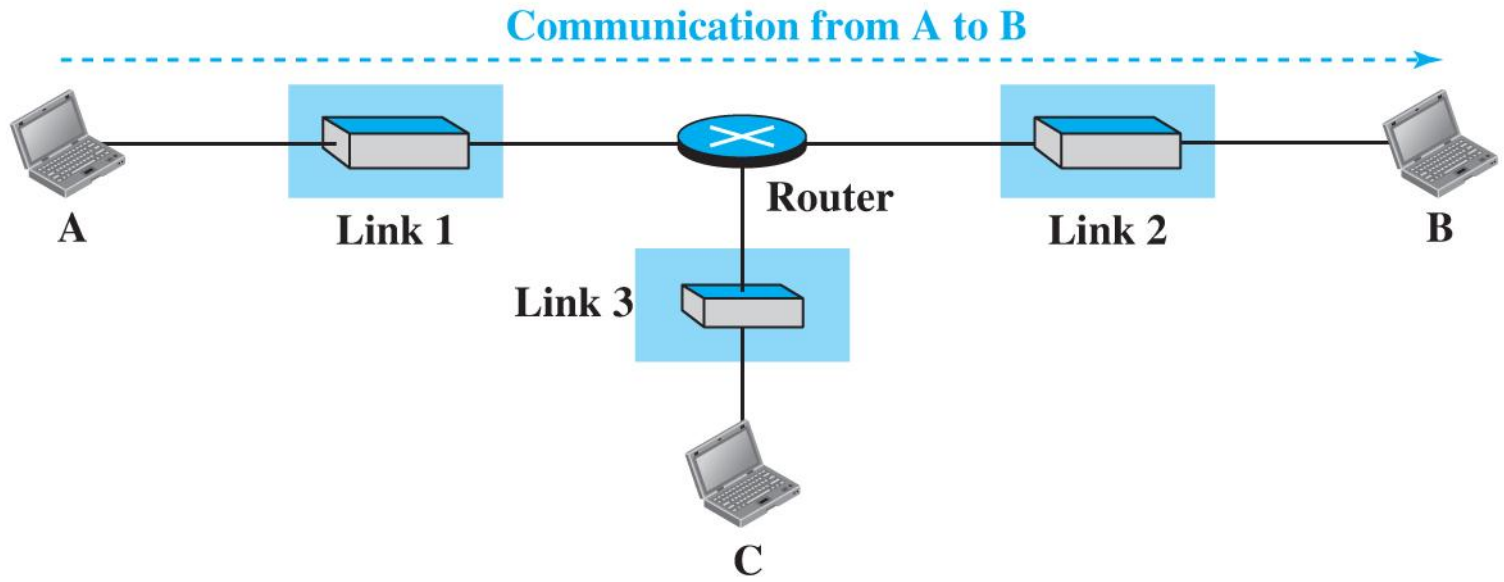
- A new paradigm, which has emerged to respond to the needs of some new applications. E.g BitTorrent, Skype
- In this paradigm, there is no need for a server process to be running all the time and waiting for the client processes to connect. The responsibility is shared between peers.
- A computer connected 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.

Mixed Paradigm

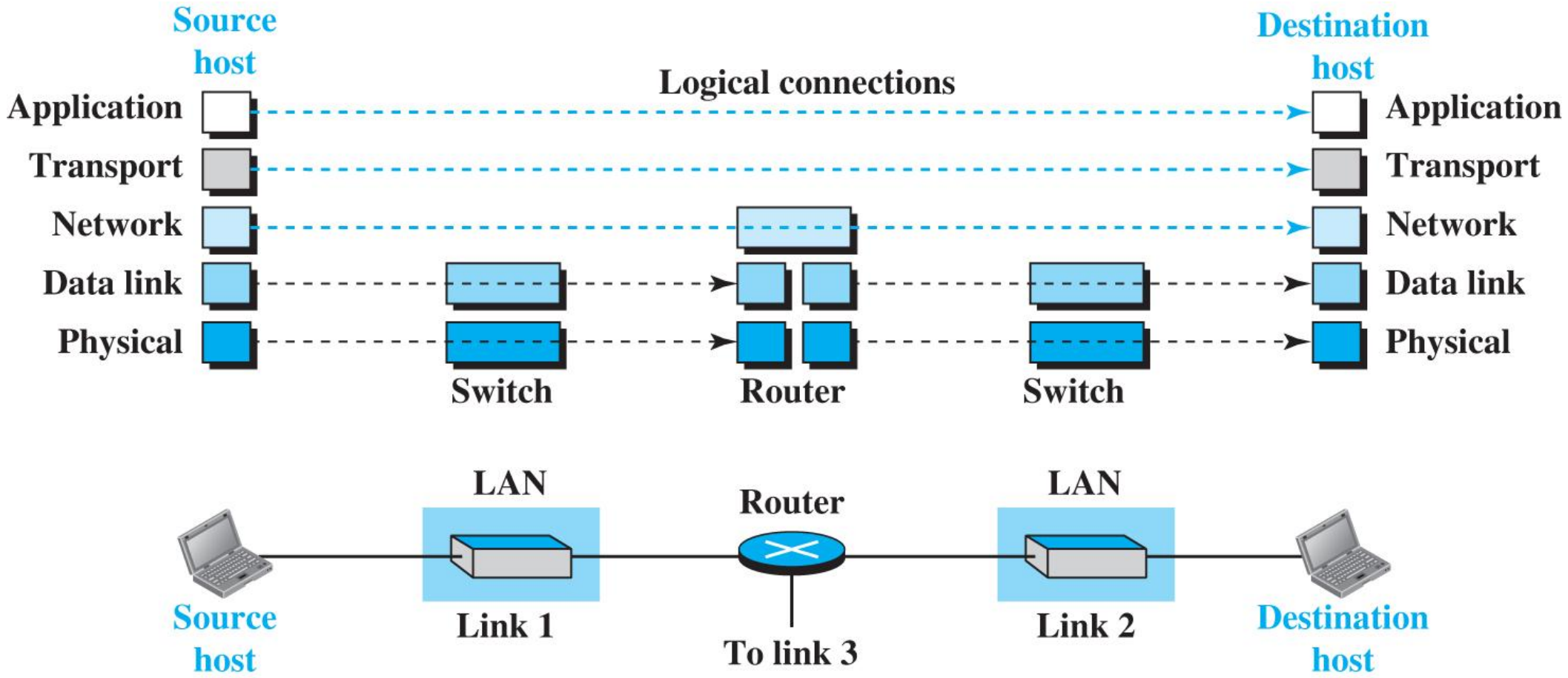
- An application may choose to use a mixture of the two paradigms by combining the advantages of both.
- For example, a light-load client-server communication can be used to find the address of the peer that can offer a service.
- When the address of the peer is found, the actual service can be received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

Packet Transmission in TCP/IP Stack





Logical Connections between layers in TCP/IP



CLIENT-SERVER PROGRAMMING

✓*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 initializes the communication by sending a request.*

✓*A server waits for a request from a client.*

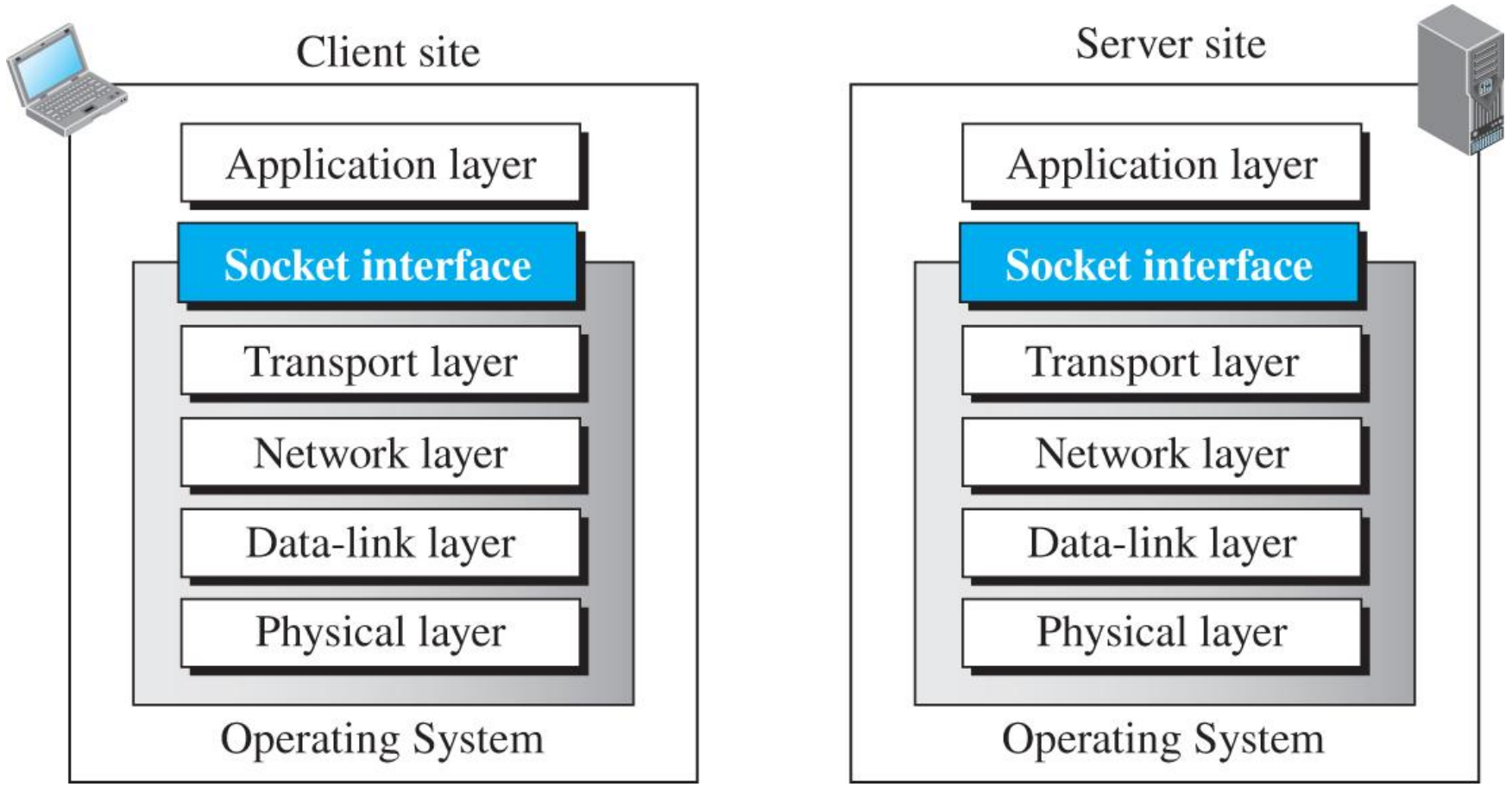
✓*The server handles the request received from a client, prepares a result, and sends the result back to the client.*

✓*A client process communicate with a server process using a set of predefined instructions referred to as an application programming interface (API).*

✓*The API for the network applications is known as socket API*

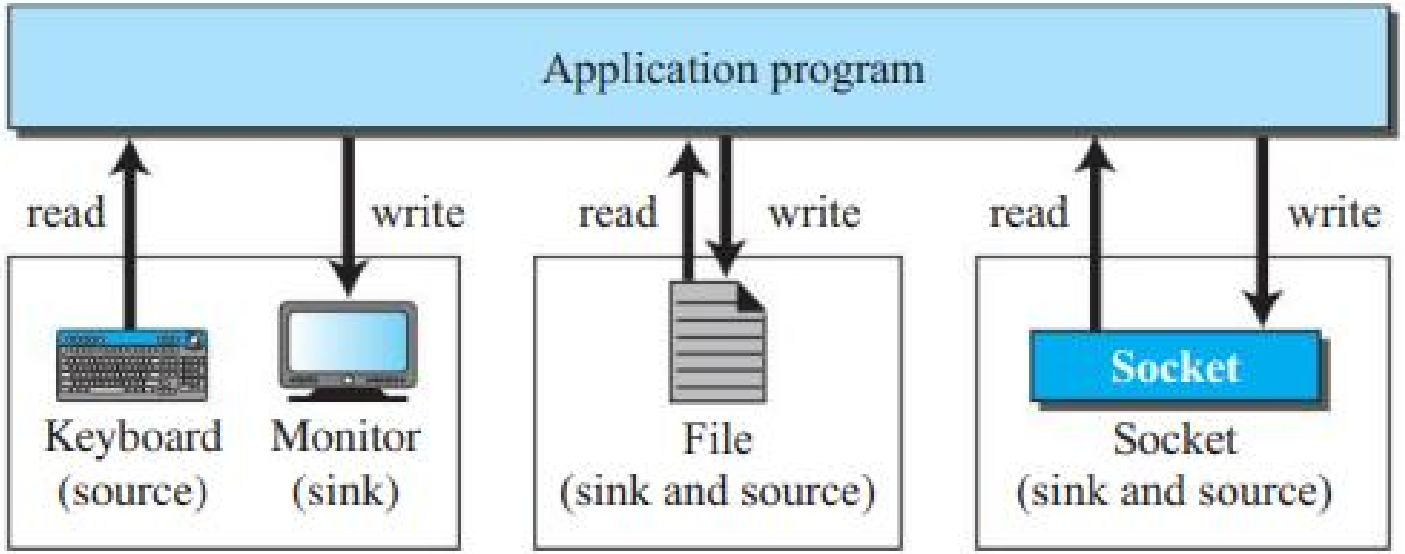
Sockets

✓A socket is the end point of communication between the client and server.
✓It acts as the interface between the application layer process and transport layer.
✓Socket API provides predefined instructions for the network applications.



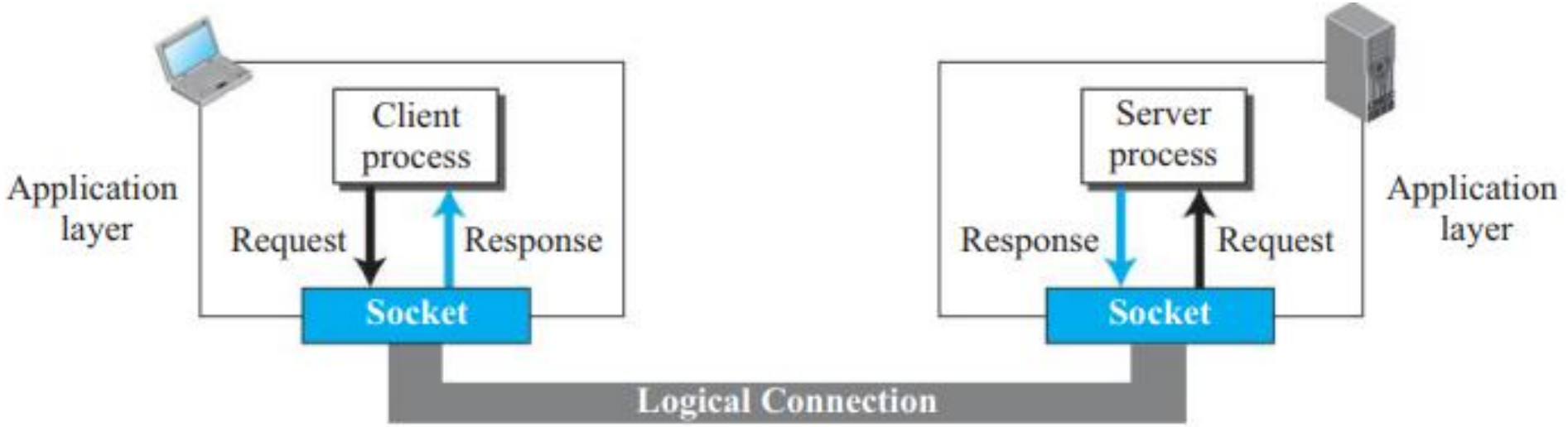
Sockets

- Sockets work just like any other input/output devices.
- A program can read from sockets and also write to sockets just like interacting with a text file or terminal.
- 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.



Role of Socket in Process to Process Communication

- Two processes think they are communicating to sockets and rest of the functionalities are handled by OS (which implements rest of layers of TCP/IP Protocol suite).



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.
- Socket address is defined using IP address and port number
- **:** is used to separate IP address and port number. For eg: 192.168.1.67:80.

10.2.2 Using Services of Transport Layer

✓*A pair of processes, need to use the services provided by the transport layer for communication because there is no physical communication at the application layer.*

✓ *There are three common transport-layer protocols in the TCP/IP suite: UDP, TCP, and SCTP.*

✓*Most standard applications have been designed to use the services of one of these protocols.*

UDP Protocol

- ✓ *UDP provides connectionless, unreliable, datagram service.*
- ✓ *Connectionless service means that there is no logical connection between the two ends exchanging messages.*
- ✓ *Each message is an independent entity encapsulated in a packet called a datagram.*
- ✓ *UDP does not see any relation (connection) between consequent datagrams coming from the same source and going to the same destination.*

TCP Protocol

✓*TCP provides connection-oriented, reliable, byte-stream service.*

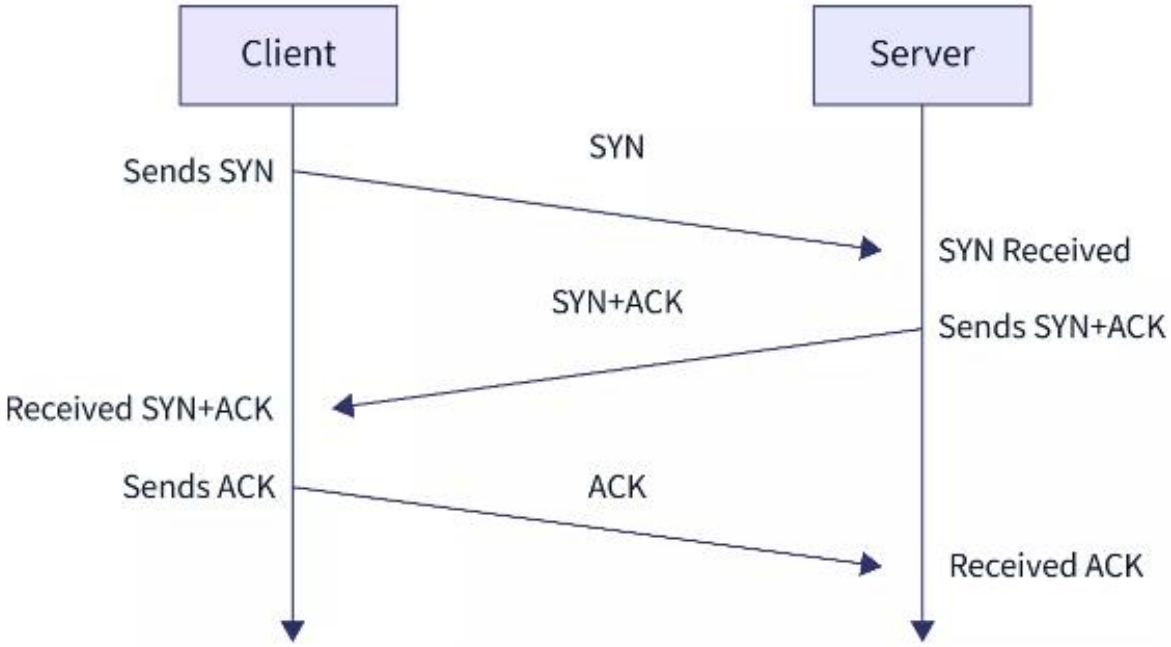
✓*TCP requires that two ends first create a logical connection between themselves by exchanging some connection-establishment packets which is called as handshaking.*

✓*Handshaking(3-Way Handshaking), establishes some parameters between the two ends including the size of the data packets to be exchanged, the size of buffers to be used for holding the chunks of data until the whole message arrives, and so on.*

TCP 3-Way Handshaking

➢*It involves three steps:*

- *SYN (Synchronize)*
- *SYN-ACK (Synchronize-Acknowledge)*
- *ACK (Acknowledge).*



Application Layer Protocols

- In particular, an application-layer protocol defines:
  - The types of messages exchanged, for example, request messages and response messages.
  - The syntax of the various message types, such as the fields in the message and how the fields are delineated.
  - The semantics of the fields, that is, the meaning of the information in the fields.
  - Rules for determining when and how a process sends messages and responds to messages.

APPLICATION LAYER PROTOCOLS

- **Standard Application-Layer Protocols**
  - Some application-layer protocols are specified in RFCs and are therefore in the public domain.
  - HTTP
  - SMTP
  - DNS
- **Non-standard Application-Layer Protocols**
  - Many application-layer protocols are proprietary and intentionally not available in the public domain.
  - Skype uses proprietary application-layer protocols.

The IETF (Internet Engineering Task Force), founded in 1986, is the premier standards development organization for the internet. The IETF publishes its technical documentation as RFCs (Requests for Comments). RFCs describe the Internet's technical foundations, such as addressing, routing, and transport technologies, etc.

The World Wide Web (WWW)

- WWW commonly known as the web, is a vast system of interlinked hypertext documents and multimedia content accessible via the internet.
- It is one of the primary services available on the internet and has transformed how people access and share information globally.
- 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.
- Linking allows one web page to refer to another web page stored in another server somewhere else in the world. The linking of web pages was achieved using a concept called *hypertext*.

Hypertext and Hyperlinks

- **Hypertext**: Text that contains links to other texts. It allows users to navigate between different documents or pages by clicking on hyperlinks.
- **Hyperlinks:** Connections within a hypertext that link one document to another.

Web Documents or Web Pages

- Web Page is a single document or file on the web, usually written in HTML (Hypertext Markup Language) and accessible via a web browser.
- A web page can contain text, images, videos, and other multimedia content.
- It can be accessed by entering the URL on the address bar of the web browser.

Website

- A collection of related web pages under a common domain name (e.g., www.example.com).
- Websites can range from a few pages to thousands of interconnected pages.

Web Documents types

The documents in the WWW can be grouped into three broad categories –

- ✓Static Documents
- ✓Dynamic Documents
- ✓Active Web Documents

Web Documents types

- **Static Documents -**
  - Static web documents are fixed content files that are delivered to the user exactly as stored on the web server. They do not change unless manually updated by a developer.
  - These are typically created using HTML and CSS and does not change based on user interaction.
- **Dynamic Documents -**
  - Dynamic web documents are generated in real-time by server-side scripts or programs in response to user requests. The content can change based on user interactions, inputs, or other parameters.
  - These are created using server-side languages like PHP, Python along with HTML, CSS, and JavaScript.

Web Documents types

Active Web Documents -

- Active web documents involve client-side scripting and interactivity, often using technologies like JavaScript, AJAX (Asynchronous JavaScript and XML), and WebSockets. They allow for real-time updates and highly interactive user experiences.
- These are created using client-side languages and framework like JavaScript, React, etc.

Exercise

A web client sends a request to a web server. The web server transmits a program to that client ans is executed at client. It creates a web document. What are such web documents called?

- [ ] Active
- [ ] Static
- [ ] Dynamic
- [ ] Passive

Answer - Active Document

Uniform Resource Locator (URL)

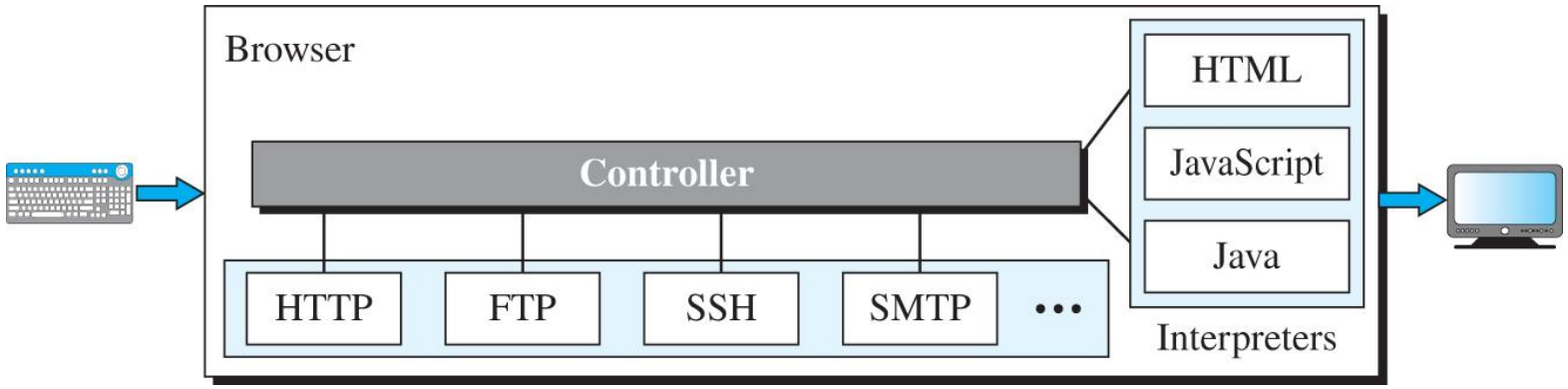
- The URL, or Uniform Resource Locator, is the web address used to access a resource on the internet.
- The URL defines four things: *protocol, host computer, port* and *path*.
- The protocol is the client/server program used to retrieve the document.
- The host is the computer on which the information is located. Web pages are usually stored in computers, and computers are given alias names that usually begin with the characters "www".
- The URL can optionally contain the port number of the server. If the port is included, it is inserted between the host and the path, and it is separated from the host by a colon.
- Path is the pathname of the file where the information is located.

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

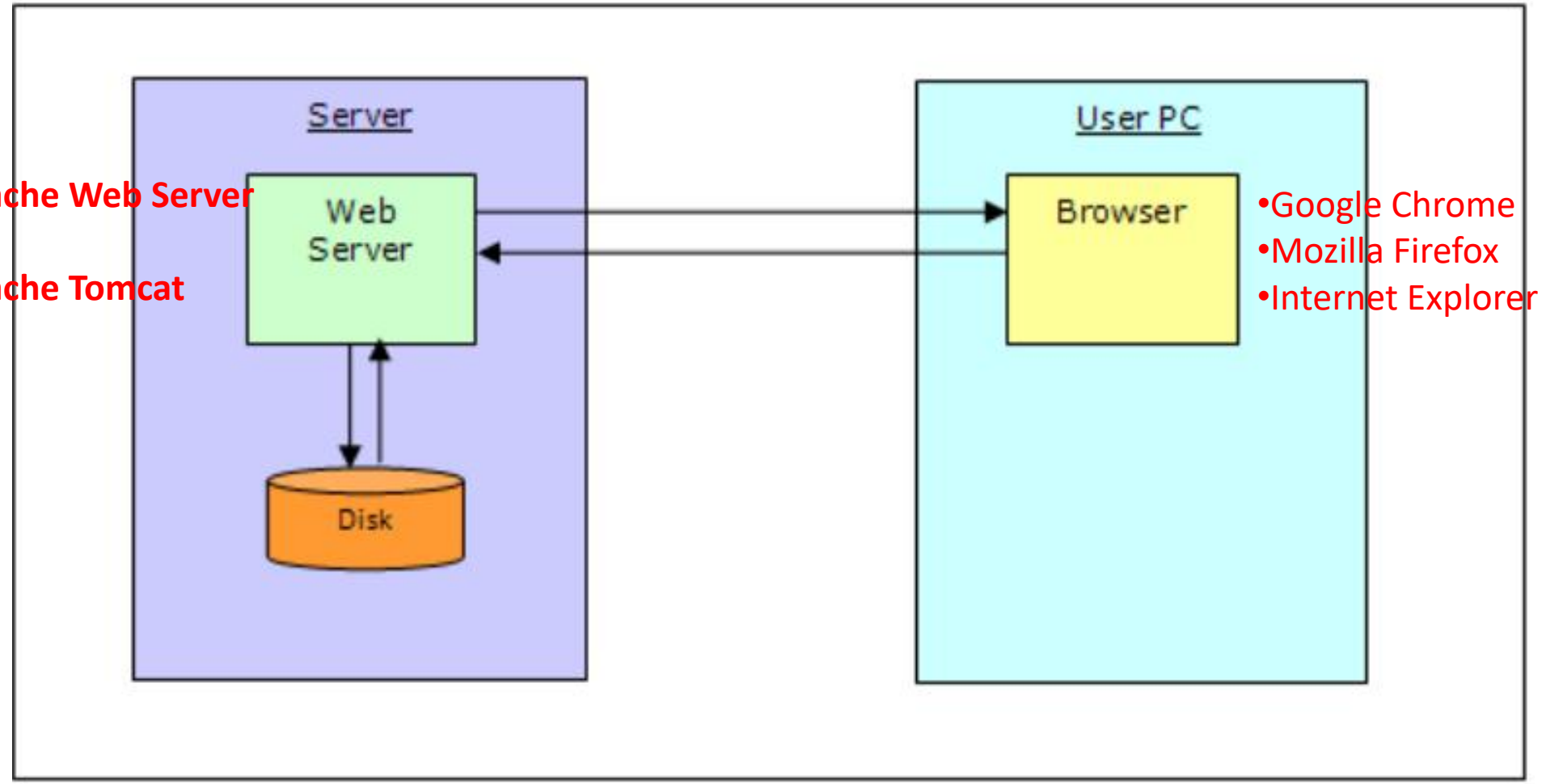
https://github.com/practical-tutorials/project-based-learning?tab=readme-ov-file#cc

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 -
  - **Controller** - receives input from the keyboard or mouse.
  - **Client Protocols** - Controller uses client protocols like HTTP to fetch the data from server.
  - **Interpreters** - After the document has been accessed, the controller uses one of the interpreters to interpret and display the document on the screen.



Interaction between web browser and web server



- **Apache Web Server**
- **IIS**
- **Apache Tomcat**

- Google Chrome
- Mozilla Firefox
- Internet Explorer

Web Server

- A web server is a type of software or hardware that serves web pages to users over the internet. Its main job is to handle incoming requests from web browsers or other clients and deliver the requested web content.

Working of Web Server

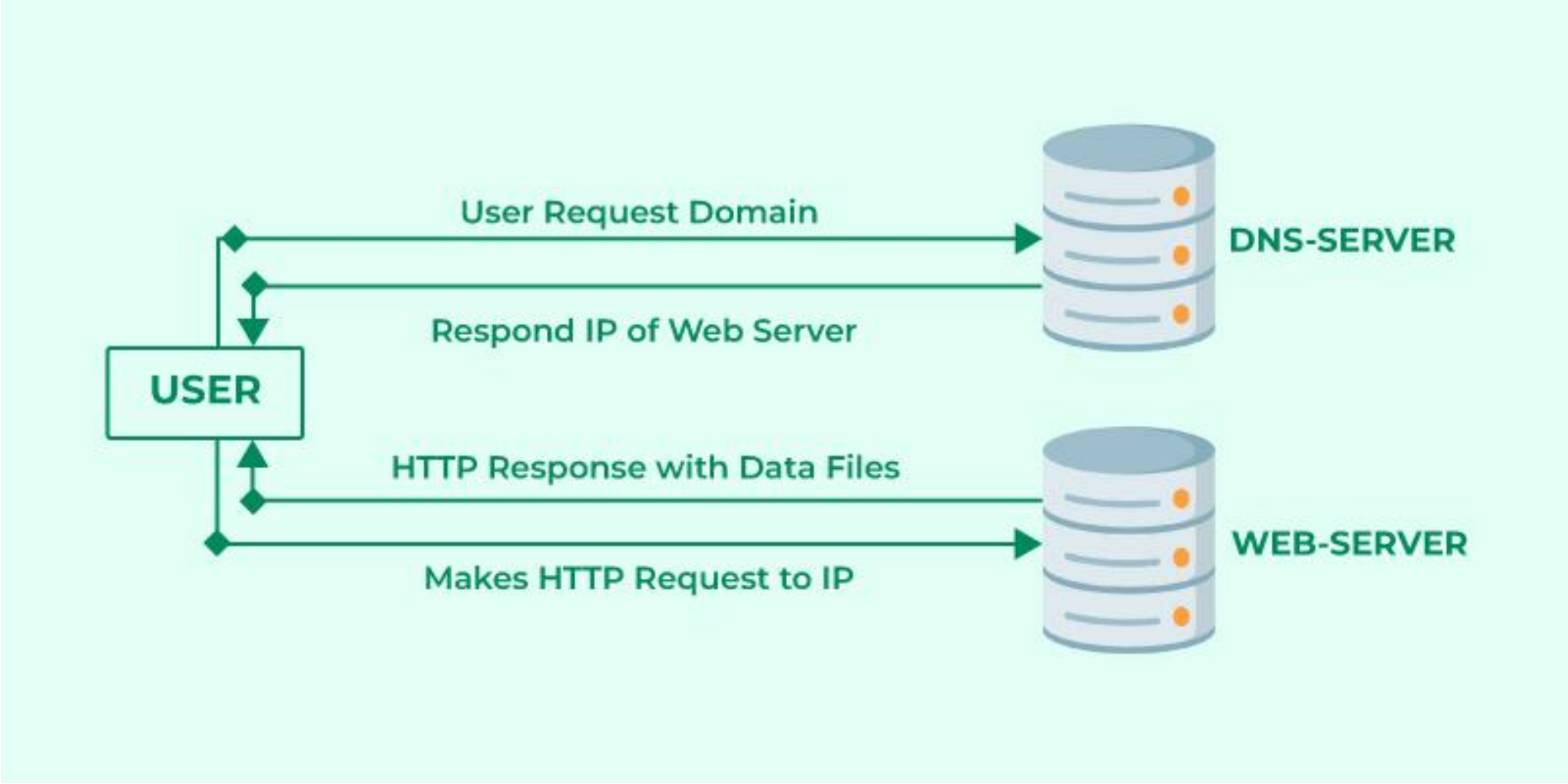
1. **Receiving Requests**: When you type a URL into your browser or click on a link, your browser sends a request to a web server to access a specific web page.
2. **Processing Requests**: The web server receives this request and determines what content is needed. It could be a static file (like an HTML page or an image) or it might need to run some server-side code (like PHP, Python, or JavaScript) to generate the content dynamically.
3. **Sending Responses**: Once the web server has the content ready, it sends it back to your browser as an HTTP response. Your browser then renders this content so you can view the web page.

How the browser interacts with the servers?

- There are few steps to follow to when client wants to interact with a server -

1. User enters the URL(Uniform Resource Locator) of the website or file. The Browser then requests the DNS(DOMAIN NAME SYSTEM) Server.
2. DNS Server lookup for the address of the WEB Server.
3. DNS Server responds with the IP address of the WEB Server.
4. Browser sends over an HTTP/HTTPS request to WEB Server's IP (provided by DNS server).
5. Server sends over the necessary files of the website.
6. 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.

INTERACTION BETWEEN WEB BROWSER AND WEB SERVER



Example 10.2

Assume we need to retrieve a scientific document that contains one reference to another text file and one reference to a large image. Figure 10.8 shows the situation.

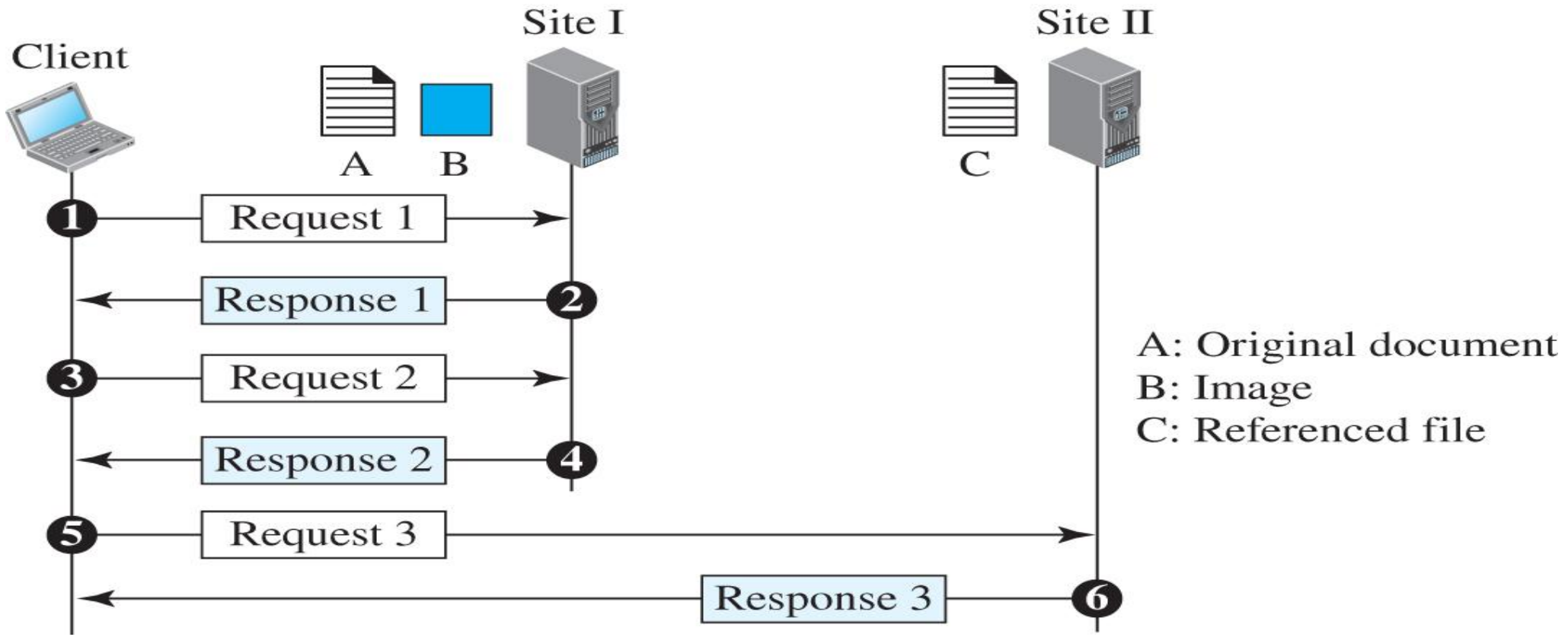


Figure 10.8

HypterText Transfer Protocol (HTTP)

- It is a protocol used for transferring hypertext requests and information on the World Wide Web. Here's how it works in the application layer:
- The HTTP uses the services of TCP, which is a connection-oriented and reliable protocol. This means that before any transaction between the client and the server can take place, a connection needs to be established between them. After the transaction is completed, connection is terminated.
- 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.
- The client and server do not need to worry about errors in messages exchanged or loss of any message because the TCP protocol will handle all these issues.

Working of http in the application layer

1. Client-Server Model: *http uses client server model*

- **Client:** The client sends a request to a server using a web browser or any HTTP client (like curl or Postman).
- This request could be for web pages, images, or other resources.
- **Server:** The server processes the client's request and sends back the appropriate response, typically in the form of HTML, JSON, XML, images, or other media.

Working of http in the application layer

2. Request/Response Structure:

- **HTTP Request**: The HTTP request includes:
  - **Request Line**: Specifies the method (e.g., GET, POST), the resource URL, and the HTTP version.
  - **Headers:** Provide additional information about the request (e.g., User-Agent, Content-Type(Accept: image/jpeg)).
  - **Body:** (Optional) Contains data to be sent to the server, mainly in POST or PUT requests.
- **HTTP Response:** The server responds with:
  - **Status Line**: Indicates the HTTP version and the status code (e.g., 200 OK, 404 Not Found).
  - **Headers**: Provide information about the response (e.g., Content-Type, Content-Length).
  - **Body:** Contains the requested resource or error message.

Working of http in the application layer

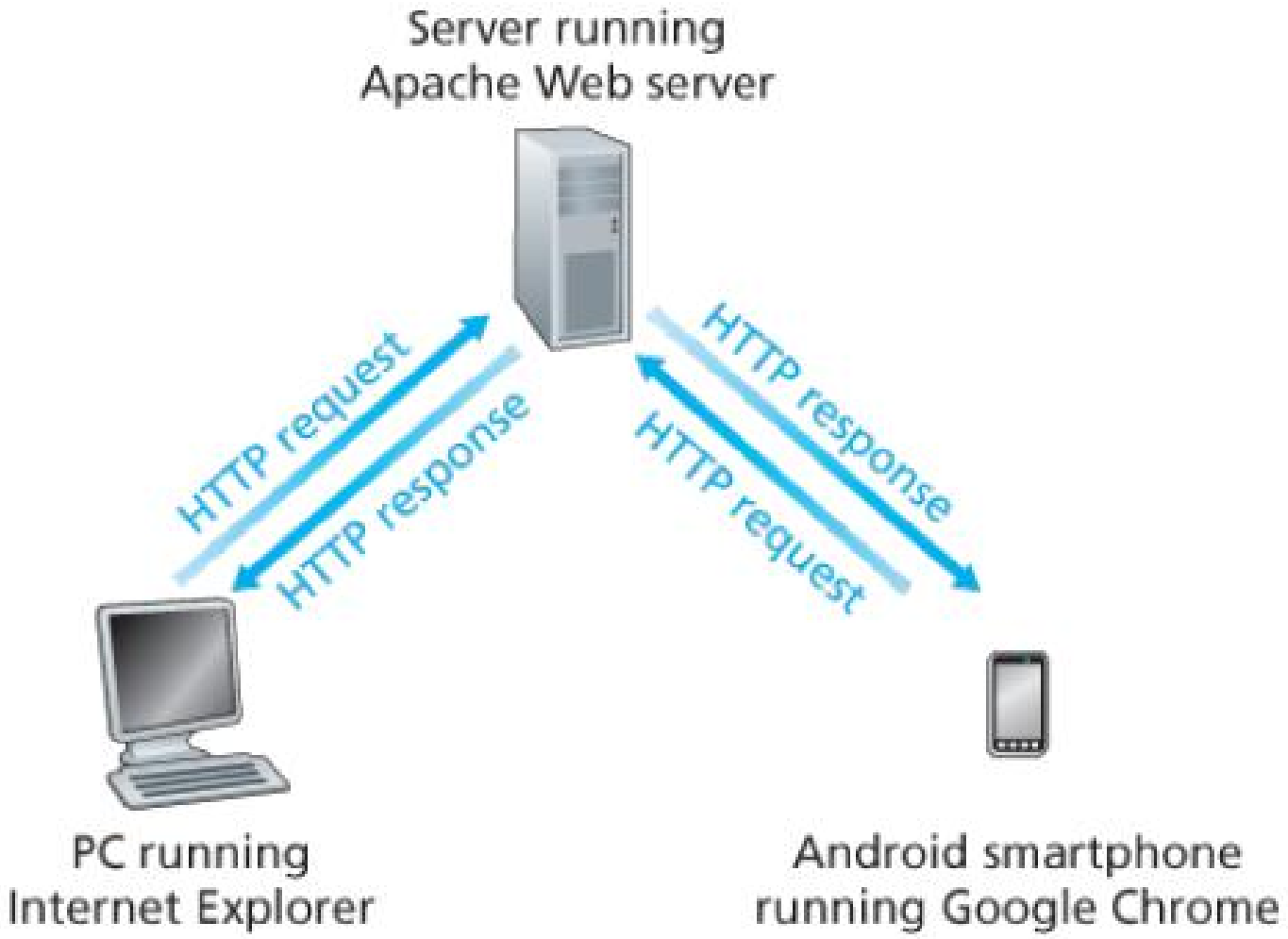
3. Methods used by http:

- ✓**GET:** Requests data from a specified resource.
- ✓**POST:** Sends data to the server to create or update a resource
- ✓**PUT:** Updates a current resource with new data.
- ✓ **DELETE:** Deletes a specified resource.
- ✓**HEAD, OPTIONS, PATCH**: Other methods used for specific purposes.

HTTP a Stateless Protocol:

- HTTP is stateless, meaning each request from the client to the server is independent.
- The server does not store any state about the client between requests.
- However, sessions and cookies can be used to maintain stateful interactions.

HTTP Server-Client Communication



Connection Types: Non persistent vs Persistent Connections

- As we have discussed, HTTP establishes a connection between server and client before they can communicate with each other.
- So if a client wants to retrieve a webpage, it need to establish a connection with server first.
- WHAT IF client want to retrieve multiple webpages?
- WHAT IF all webpages are on different servers?
- WHAT IF all the webpages are on the same server?
- Would it need a new connection for each webpage? or It can be done using a single connection only?

Nonpersistent Connections

- In a non persistent connection, separate TCP connection is made for each request made by client.
- Steps:
  - The client opens a TCP connection and sends a request.
  - The server sends the response and closes the connection.
  - The client reads the data until it encounters an end-of-file marker; 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 non persistent strategy imposes high overhead on the server because the server needs *N+1* different buffers each time a connection is opened.

Example: Nonpersistent Connection

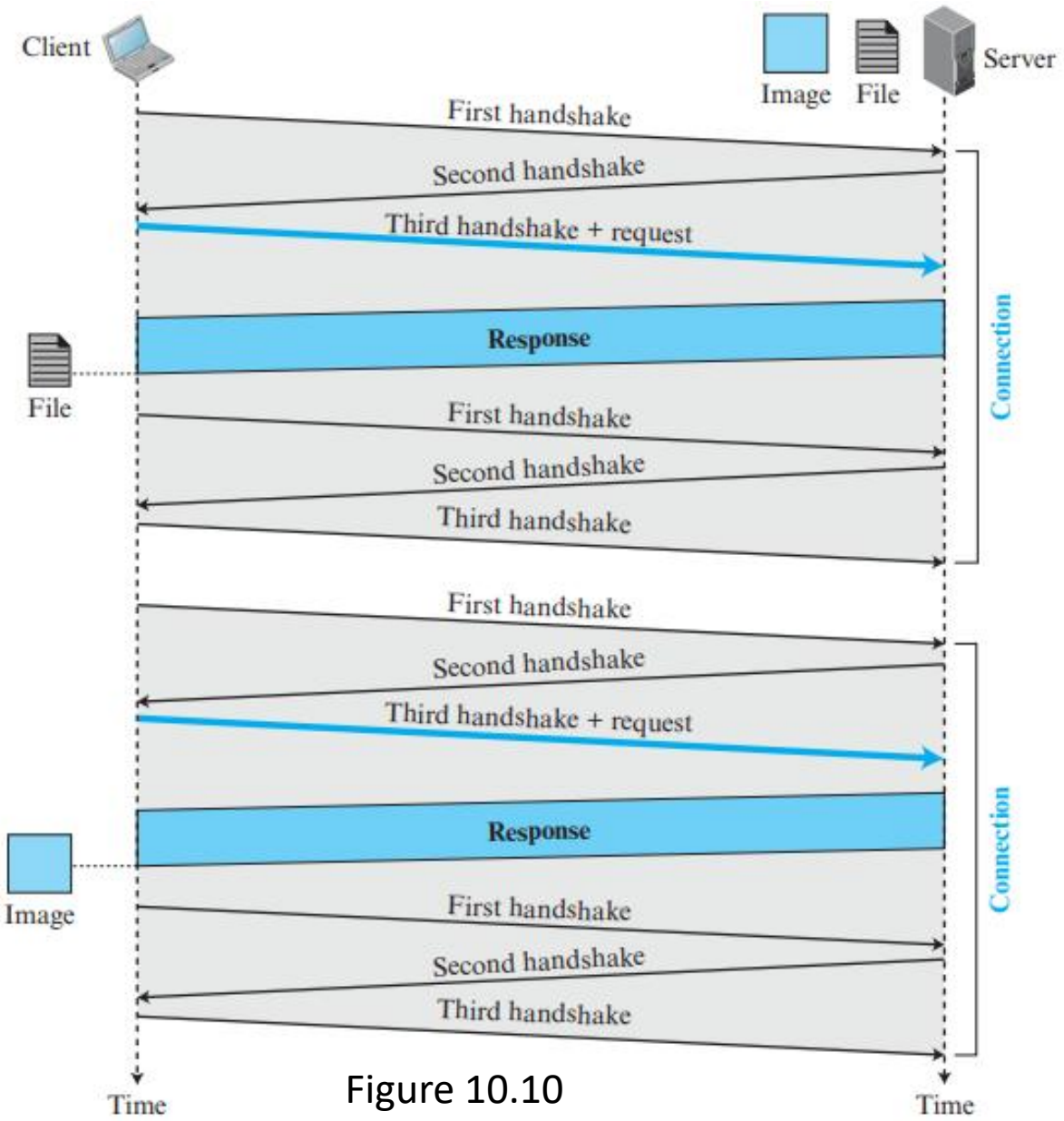


Figure 10.10

- Figure 10.10 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, but the request can be sent with the third one.
- After the connection is established, the object can be transferred.
- After receiving an object, another three handshake messages are needed to terminate the connection.

Persistent Connections

- In a persistent connection, the server leaves the connection open for more requests after sending a response.
- The server can close the connection at the request of a client or if a time-out has been reached.
- From HTTP 1.1 persistent connections are default option. Although It can be changed based on user requirements.
- 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.
- The sender usually sends the length of the data with each response. However, 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.

Example: Persistent Connection

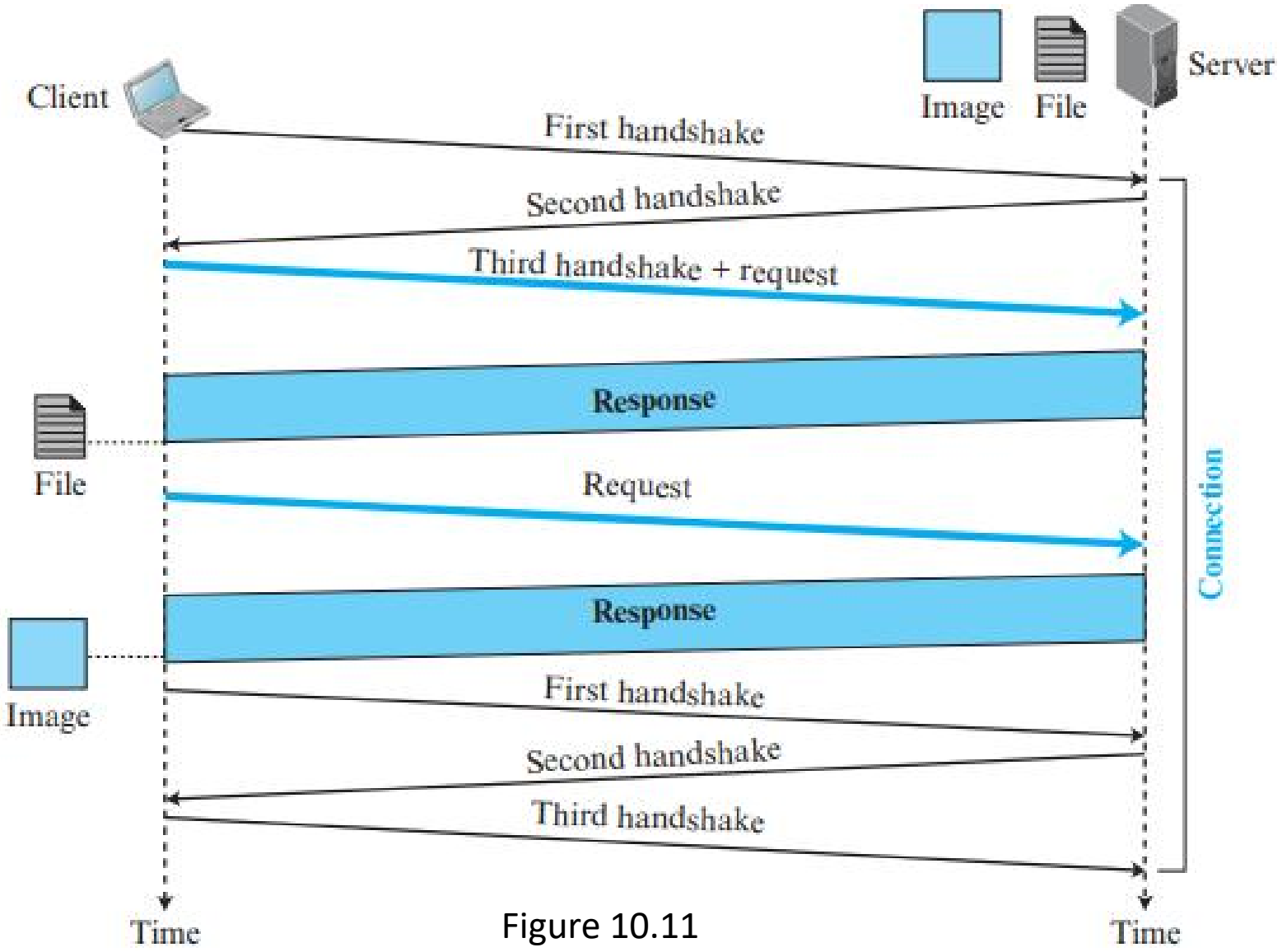


Figure 10.11

- Figure 10.11 shows an example of a persistent 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.
- Only one connection establishment and connection termination is used, but the request for the image is sent separately.

Exercise

Identify the correct order in which the following actions take place in an interaction between a web browser and a web server.

1. The web browser requests a webpage using HTTP.
2. The web browser establishes a TCP connection with the web server.
3. The web server sends the requested webpage using HTTP.
4. The web browser resolves the domain name using DNS.

Correct sequence is: 4 -> 2 -> 1 -> 3

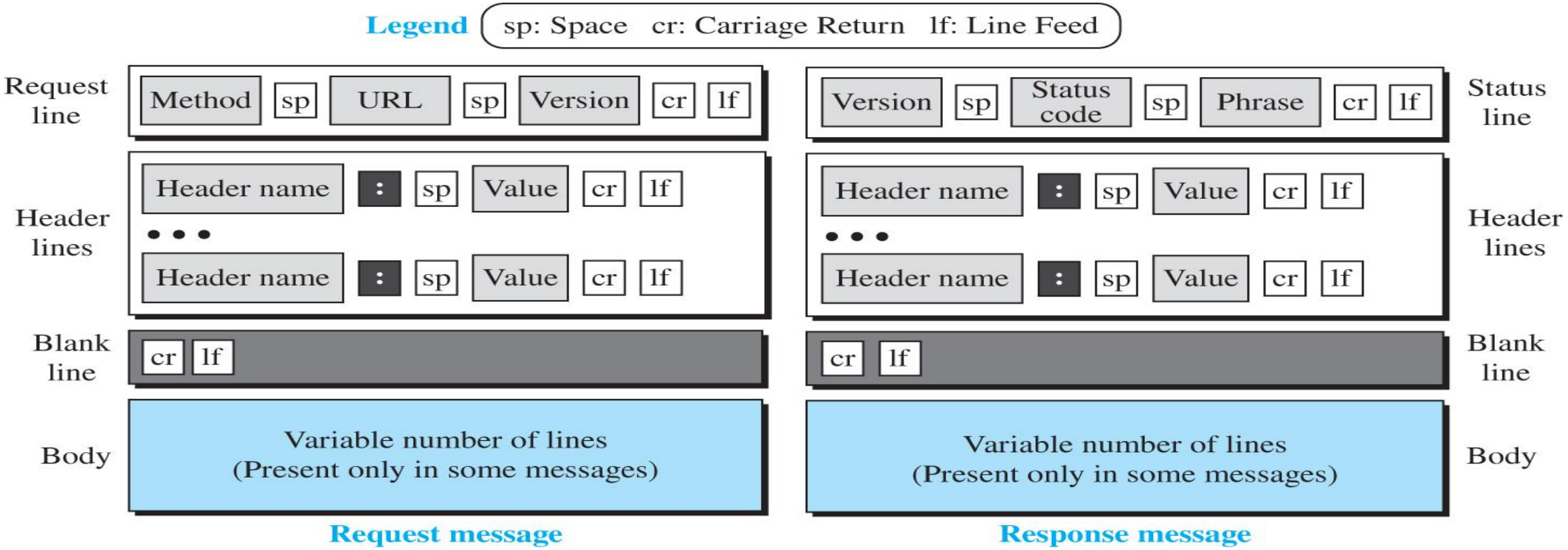
Exercise

A graphical HTML browser resident at a network client machine Q accesses a static HTML webpage from a HTTP server S. The static HTML page has exactly one static embedded image which is also at S. Assuming no caching, which one of the following is correct about the HTML webpage loading (including the embedded image)?

- [ ] Q needs to send at least 2 HTTP requests to S, each necessarily in a separate TCP connection to server S
- [ ] Q needs to send at least 2 HTTP requests to S, but a single TCP connection to server S is sufficient
- [ ] A single HTTP request from Q to S is sufficient, and a single TCP connection between Q and S is necessary for this
- [ ] A single HTTP request from Q to S is sufficient, and this is possible without any TCP connection between Q and S

Answer - Q needs to send at least 2 HTTP requests to S, but a single TCP connection to server S is sufficient.

Figure 10.12 Formats of the request and response messages



Access the text alternative for slide images.

Table 10.1 Methods

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

Table 10.2 Request header names

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

Table 10.3 Response header names

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

Example 10.6

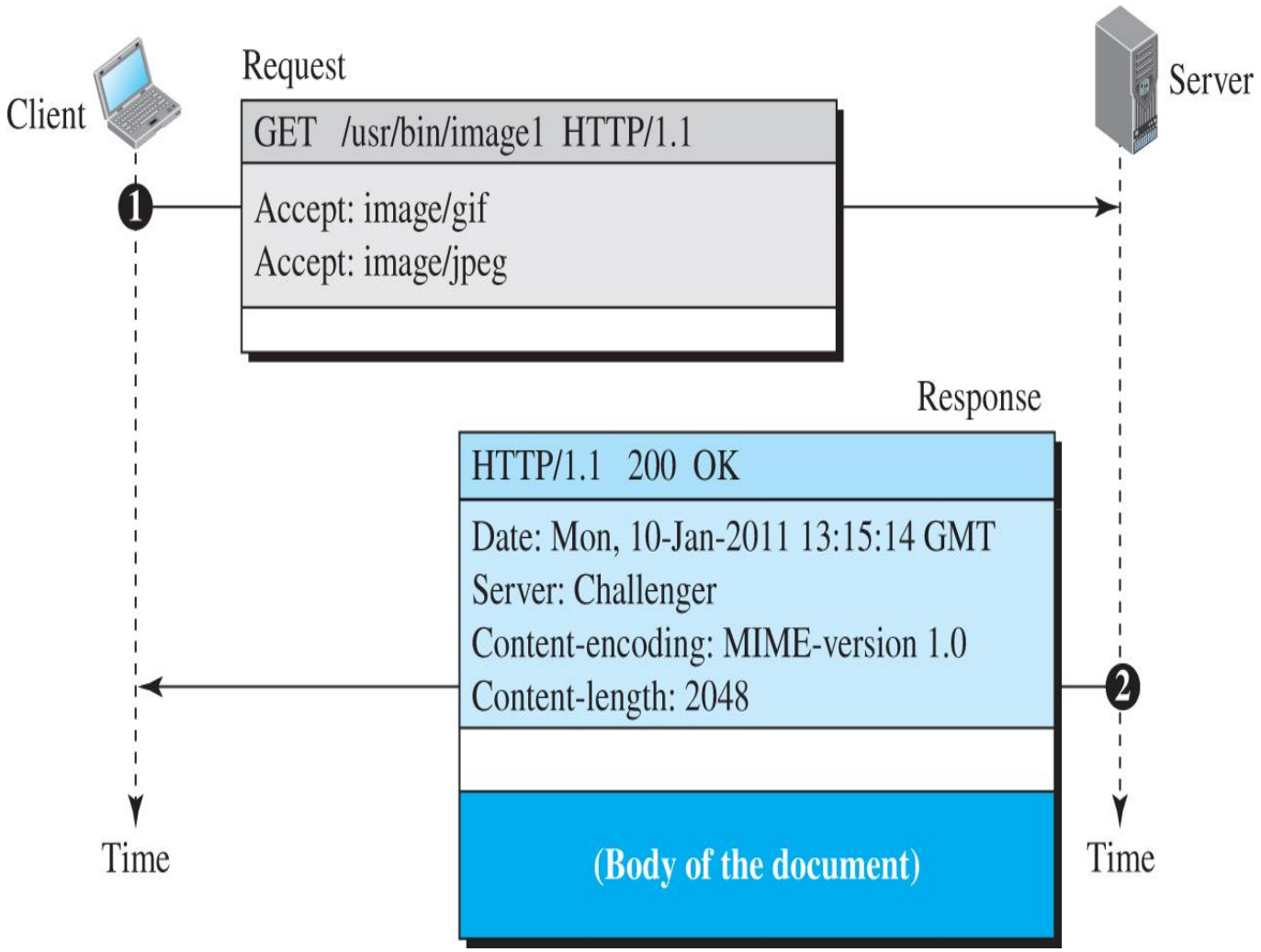


Figure 10.13.

- ✓ In this example client wants to retrieve a document (see Figure 10.13). 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 (MIME version, which will be described in electronic mail), and length of the document. The body of the document follows the header.

Access the text alternative for slide images.

Example 10.7

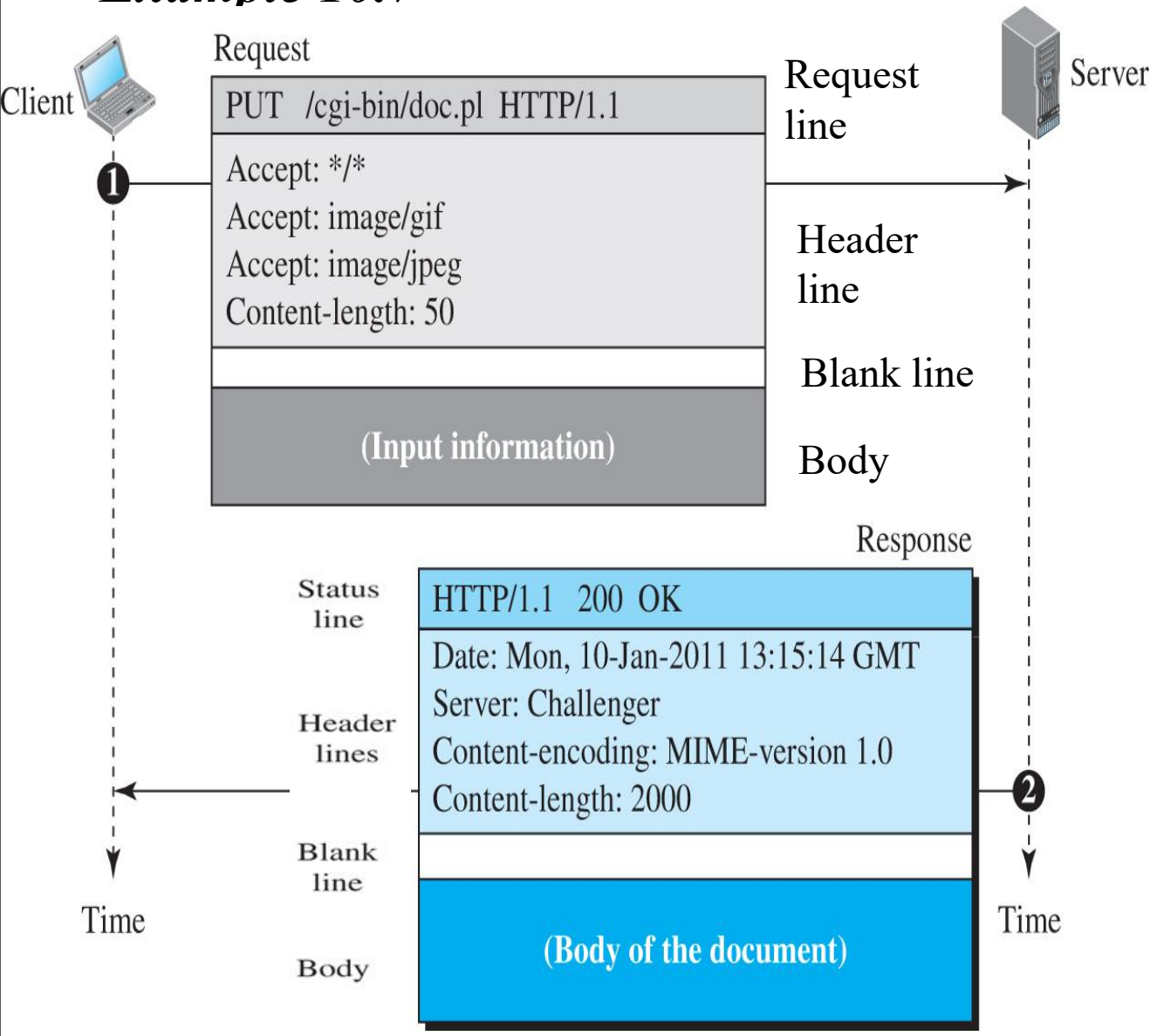


Figure 10.14).

- ✓In this example, the client wants to send a web page to be posted on the server using 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 10.14).

Access the text alternative for slide images.

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.

Example 10.8

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**

Stateless Server

- A stateless server does not retain any information about client interactions between different requests.
- Each request from a client to the server is independent.
- Server does not store any session information about the client's state between requests.
- HTTP is an example of a stateless protocol.

Cookies

✓Internet Cookies (aka web cookies) are small text files which store user-specific information and user preferences.

✓Cookies can help to track your activity and behavior online in order to personalize your experience.

✓However, they can also be used to violate your online privacy.

Creating and Storing Cookies

- When a server receives a request from a client, it stores information about the client and preferences in a file known as **cookie**.
- The server includes the cookie in the response that it sends to the client.
- When the client receives the response, the browser stores the cookie in the cookie directory as the server domain name.
- Cookies are sent back to the server on each request to the same domain.

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.

Example: E-commerce Cookie

- An electronic store (e-commerce) can use a cookie for its client shoppers.
- When a client selects an item and inserts it in a cart, a cookie that contains information about the item, such as its number and unit price, is sent to the browser.
- If the client selects a second item, the cookie is updated with the new selection information, and so on.
- When the client finishes shopping and wants to check out, the last cookie is retrieved and the total charge is calculated.

Examples of Cookies

Registered Clients Cookie

- The site that restricts access to registered clients only sends a cookie to the client when the client registers for the first time.
- For any repeated access, only those clients that send the appropriate cookie are allowed.

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 behaviour for targeted advertising.
- Now these agencies can sell this information to other interested parties.

Example: Cookie

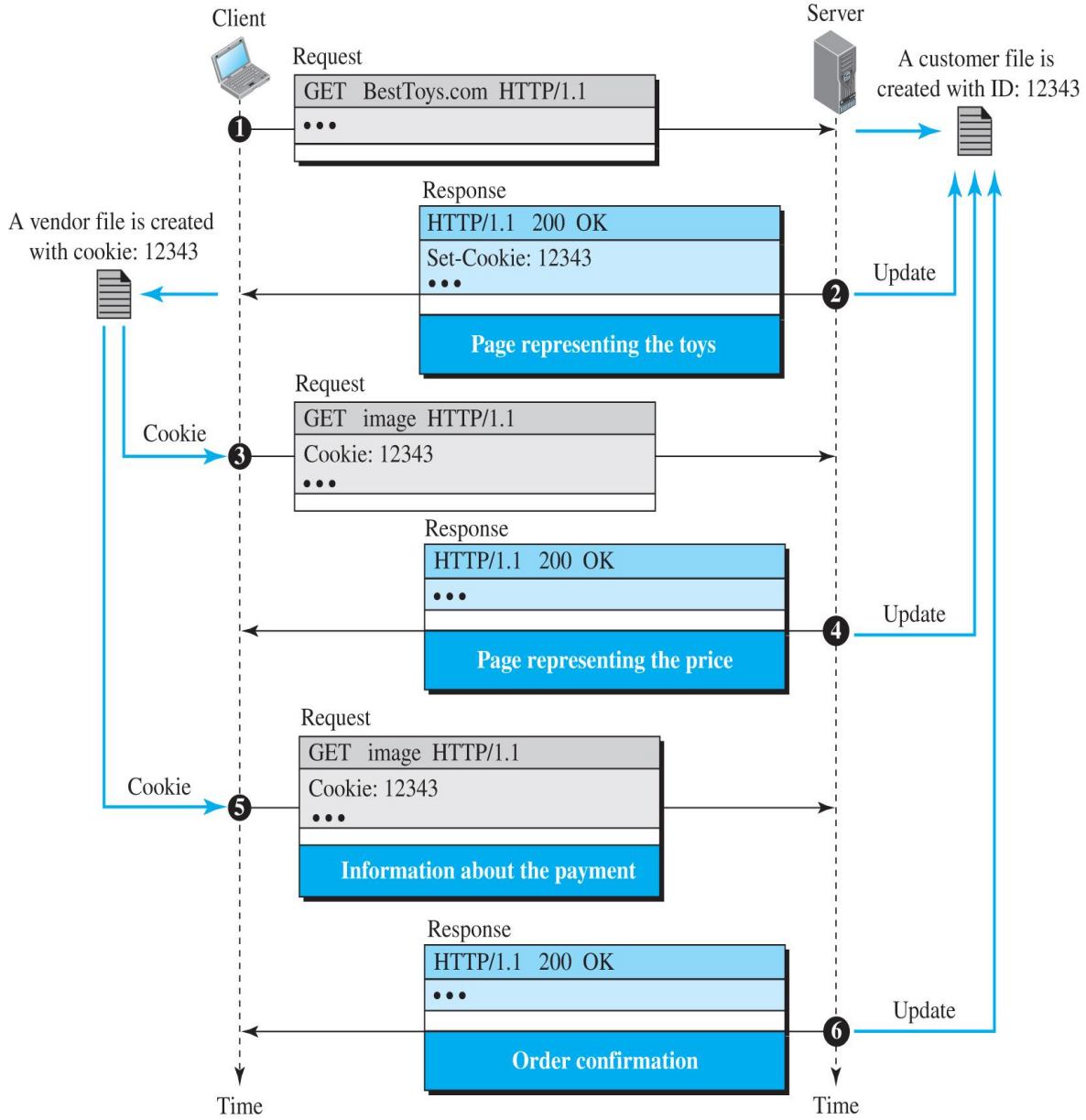


Figure 10.15

- ✓ Figure 10.15 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.

Web Caching: Proxy Server

- A proxy server is a computer that keeps copies of responses to recent requests.
- When the HTTP client sends a request, it is first goes to the proxy server.
- The proxy server checks the response in its cache and sends the response to the client if found.
- 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.
- Proxy server acts as both server and client.

Proxy Server Location

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

Example of a proxy server

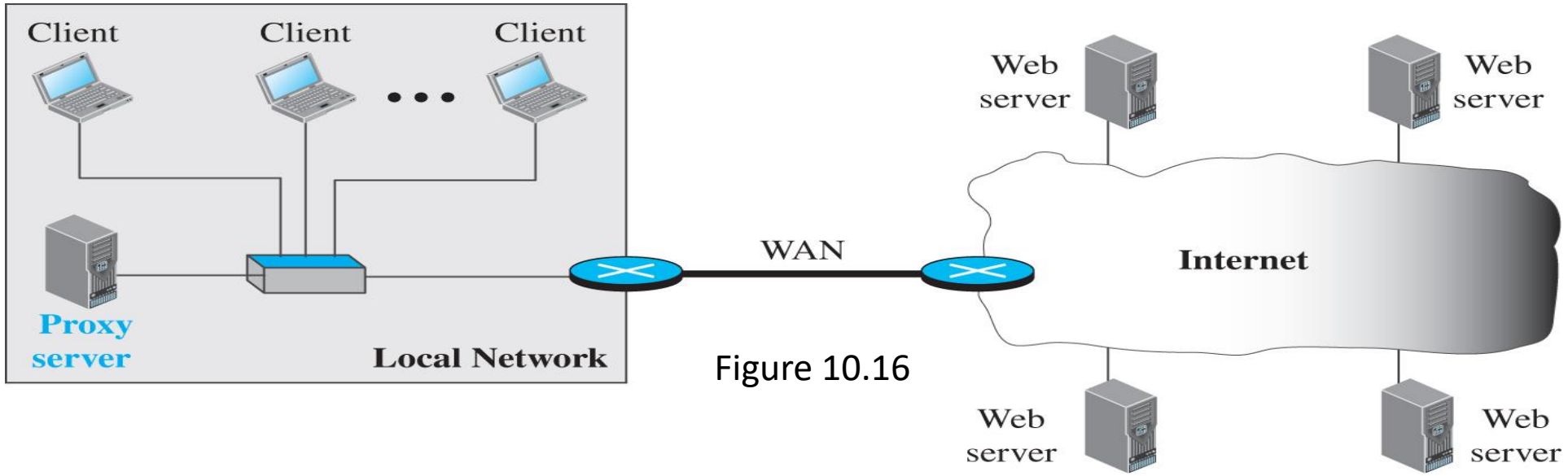


Figure 10.16

- Figure 10.16 shows an example of a use of a proxy server in a local area network, such as the network on a campus or in a company.
- 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.

Proxy Server: Cache Update Strategies

- **Time-Based Expiry (TTL):** After TTL expires, the proxy server will fetch a fresh copy from the origin server.
- **Cache Invalidation**: You can manually clear or invalidate specific cache entries.
- **Cache Purging:** Cache is automatically updated regularly based on some policies without manual intervention.
- **Cache Control Headers:** header from the original server tells the proxy server how long to cache the response.

HTTP Security

- HTTP does not have inbuilt mechanism for providing security.
- But HTTP can be run over the Secure Socket Layer (SSL) which is referred as HTTPS.
- HTTPS provides confidentiality, client and server authentication, and data integrity.

Exercise

The protocol data unit (PDU) for the application layer in the Internet stack is

- [ ] Segment
- [ ] Datagram
- [ ] Message
- [ ] Frame

Answer - Message