Chapter 5

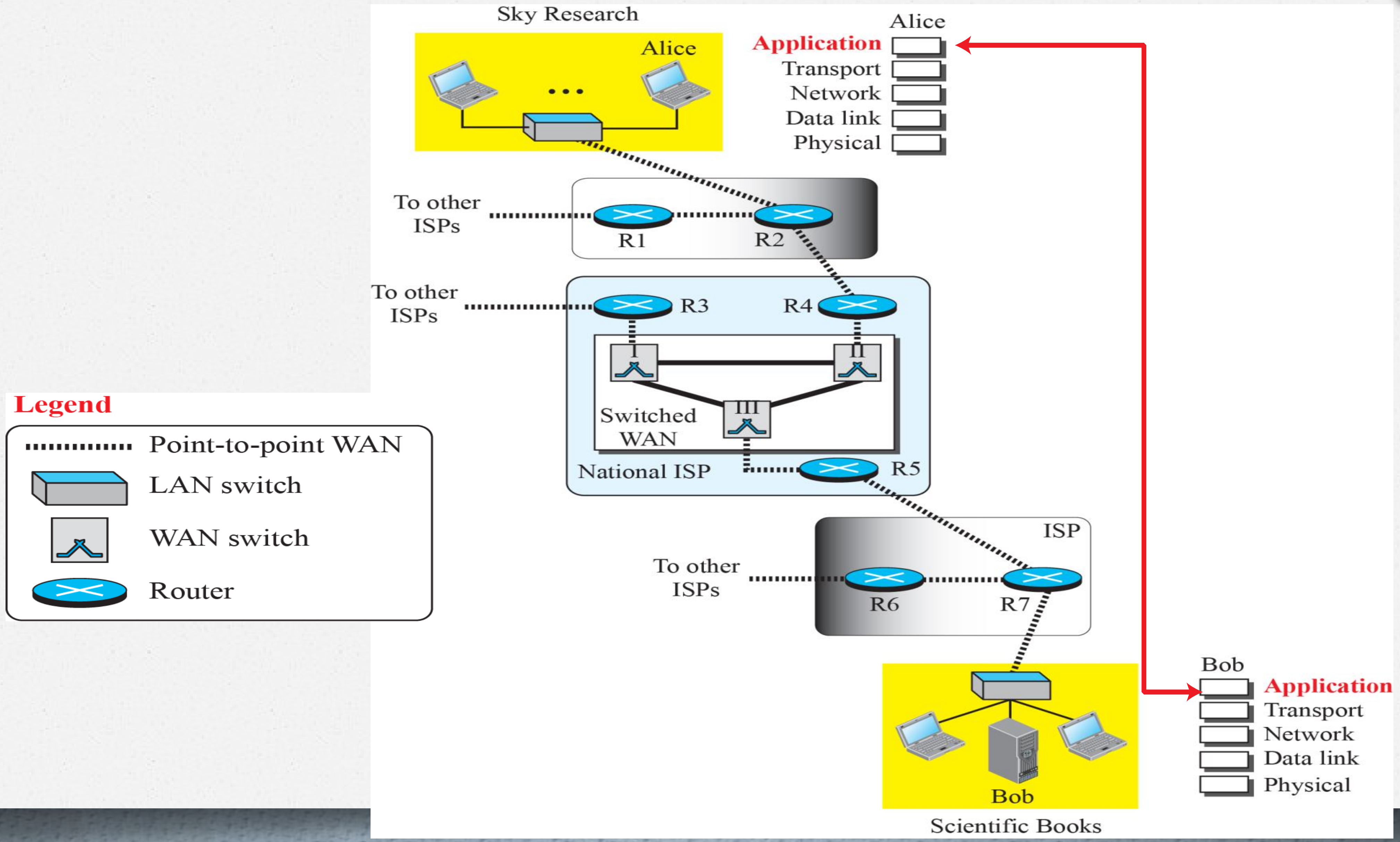
Application Layer

- INTRODUCTION
- CLIENT-SERVER PARADIGM
- STANDARD APPLICATIONS
    - HTTP
    - FTP
    - SMTP
    - Telnet
    - SSH
    - DNS

INTRODUCTION

- 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.

Logical connection at the application layer



Providing Services

The Internet was originally designed to provide service to users around the world.

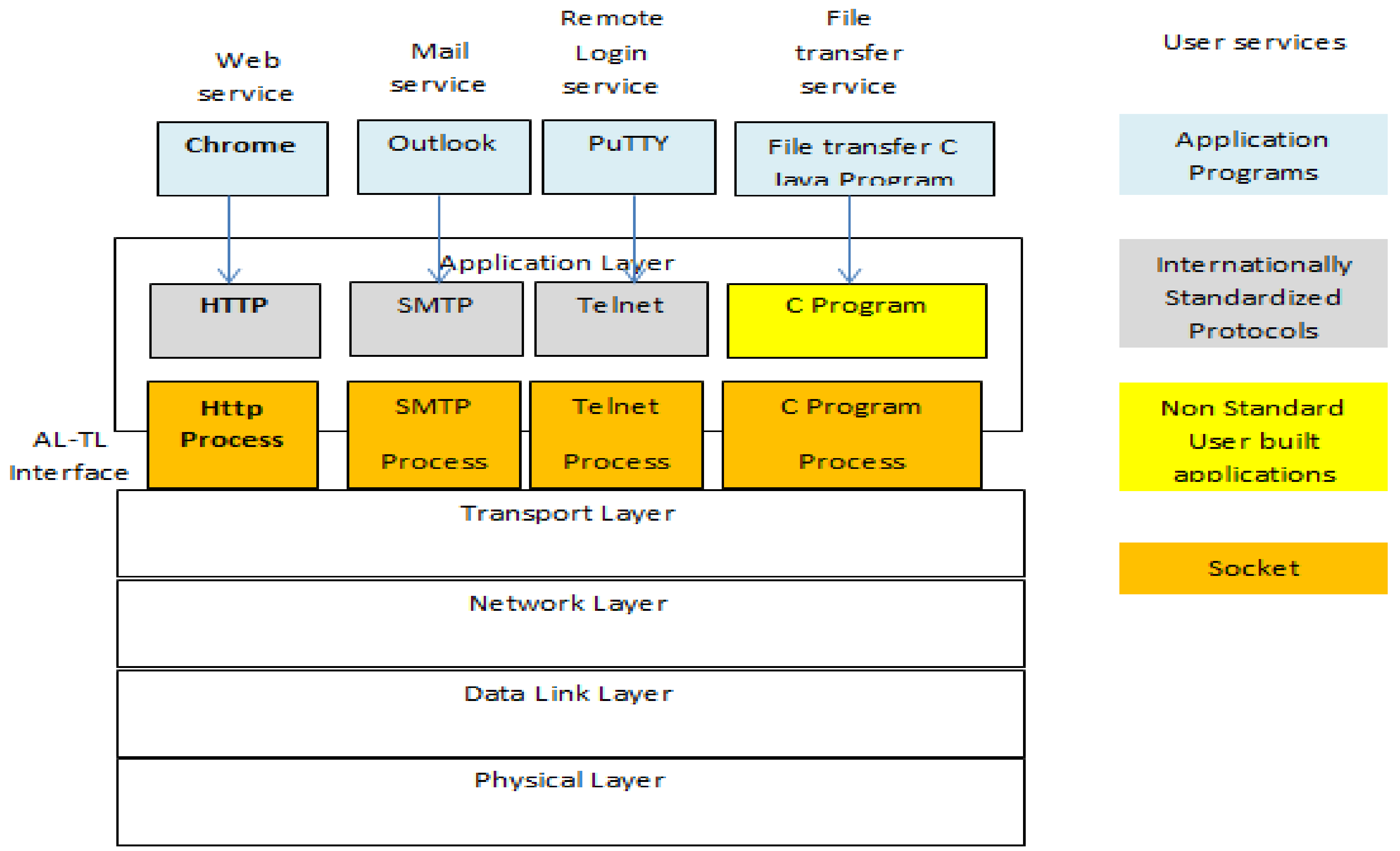
Since the application layer is the only layer that provides services to the Internet user, it allows new application protocols to be easily added to the Internet, which has been occurring during the lifetime of the Internet.

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.

- It should be clear that 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.

- 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.


Web service
Mail service
Remote Login service
File transfer service
User services
Chrome
Outlook
PuTTY
File transfer C Java Program
Application Programs
Application Layer
HTTP
SMTP
Telnet
C Program
Internationally Standardized Protocols
AL-TL Interface
Http Process
SMTP Process
Telnet Process
C Program Process
Non Standard User built applications
Socket
Transport Layer
Network Layer
Data Link Layer
Physical Layer

Standard Application-Layer Protocols

- 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.
- The study of these protocols enables a network manager to easily solve the problems that may occur when using these protocols

Nonstandard Application-Layer Protocols

- A programmer can create a nonstandard application-layer program if she can write two programs that provide service to the user by interacting with the transport layer.
- User can create a new customized application protocol to communicate using the service provided by the first four layers of the TCP/IP protocol suite without using any of the standard application programs
- User does not even need the approval of the Internet authorities if privately used, has made the Internet so popular in the world.

Application-Layer Paradigm

Traditional Paradigm: Client-Server

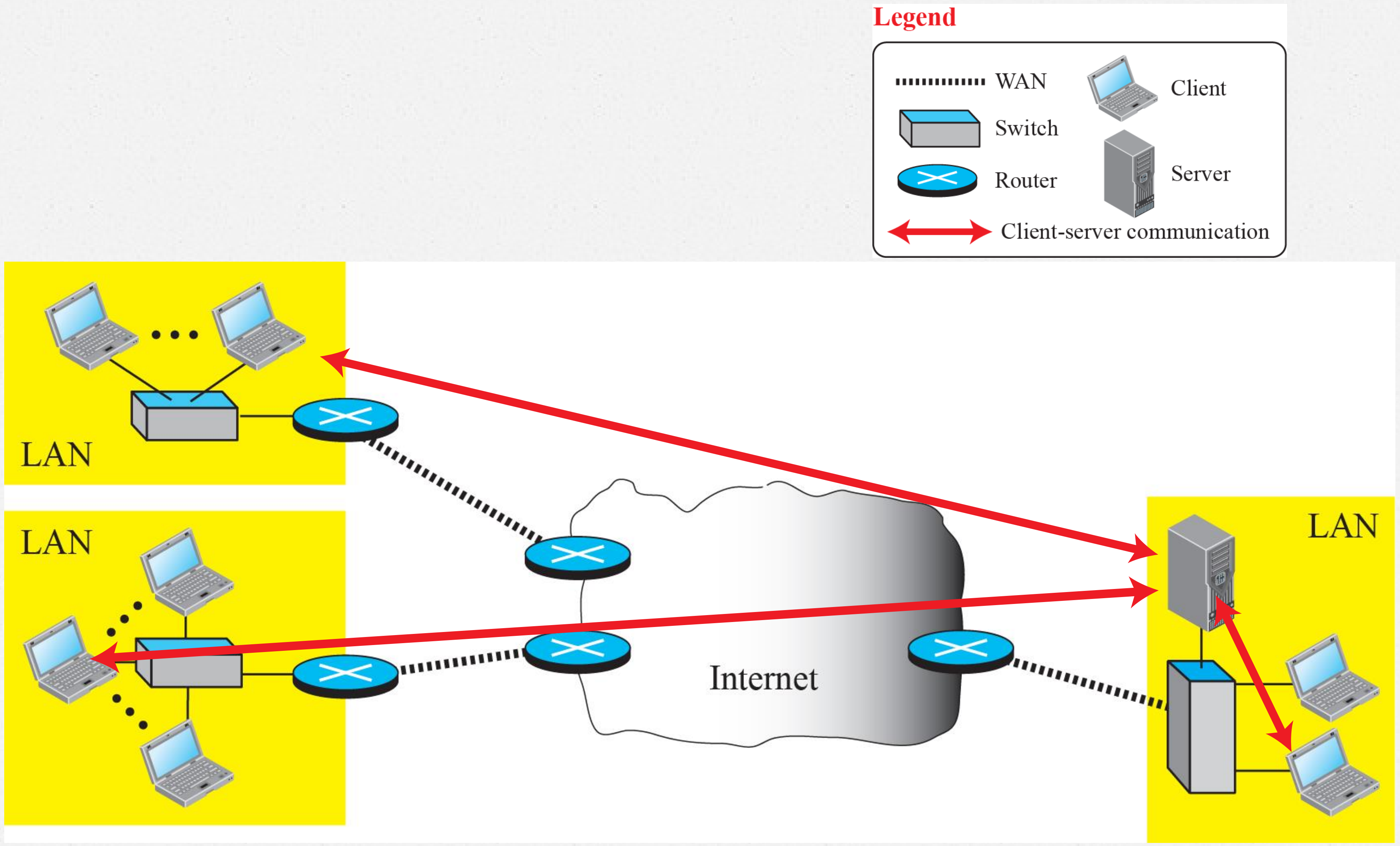
New Paradigm: Peer-to-Peer

Mixed Paradigm

CLIENT-SERVER PARADIGM

- In this 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 concentration of the communication load is on the shoulder of the server, which means the server should be a powerful computer.

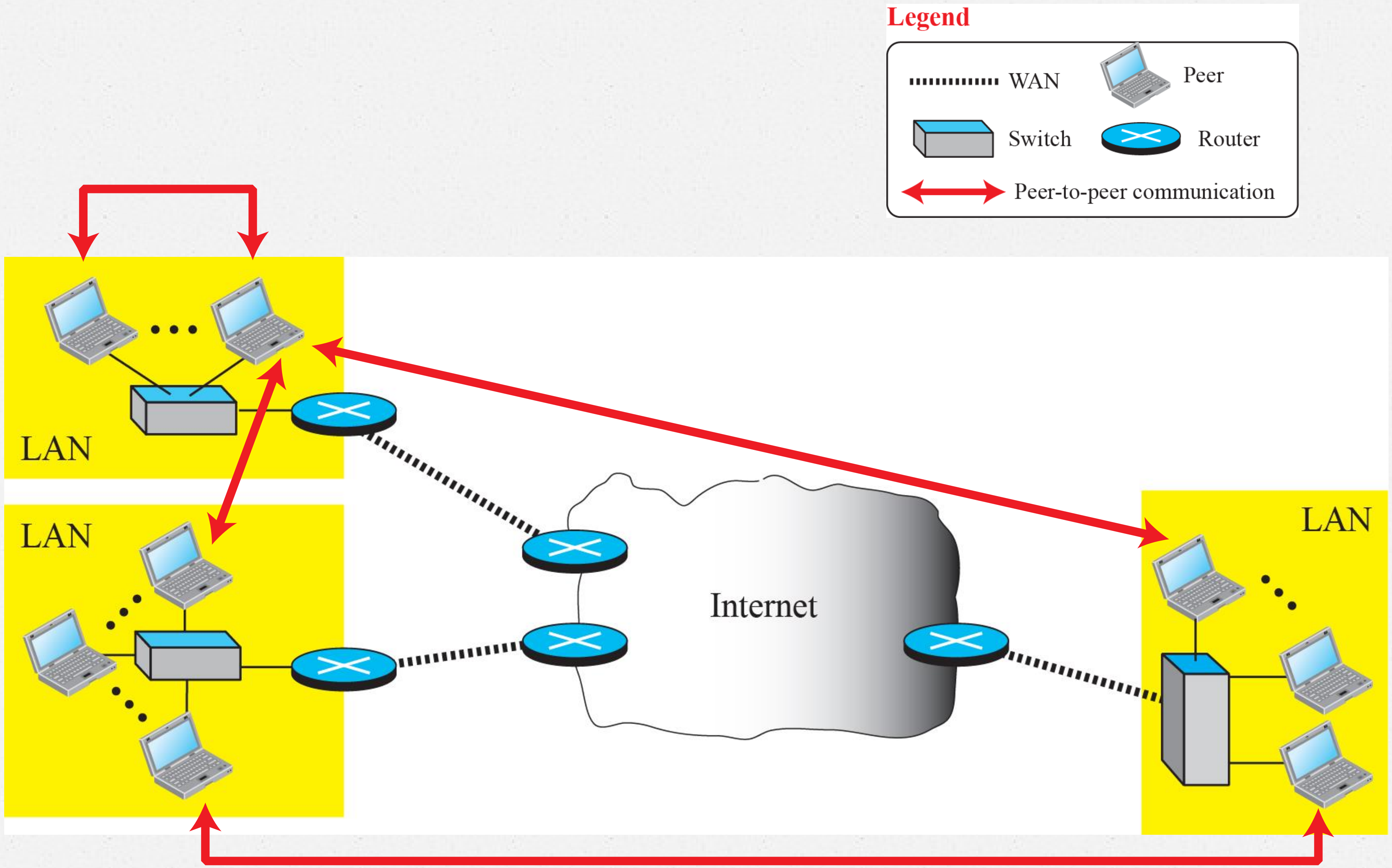
Example of a client-server paradigm



PEER to PEER PARADIGM

- 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.
- There is no need for a server process to be running all the time and waiting for the client processes to connect.
- easily scalable and cost-effective in eliminating the need for expensive servers to be running and maintained all the time
- It is more difficult to create secure communication between distributed services than between those controlled by some dedicated servers
- New applications, such as BitTorrent, Skype, IPTV, and Internet telephony, use this paradigm

Example of a peer-to-peer paradigm

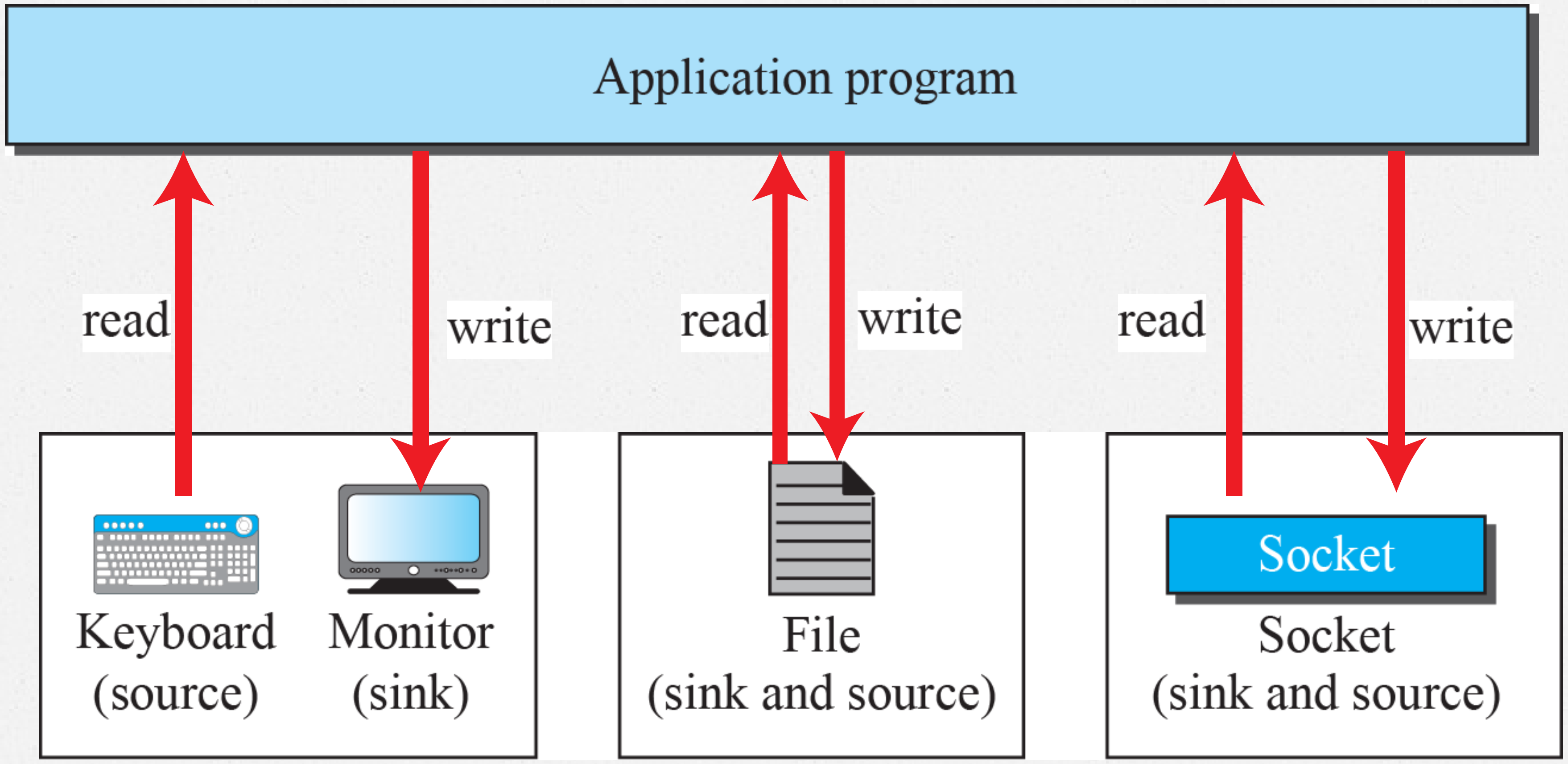


Application Programming Interface

A computer language has a set of instructions for mathematical operations, a set of instructions for string manipulation, a set of instructions for input/ output access, and so on.

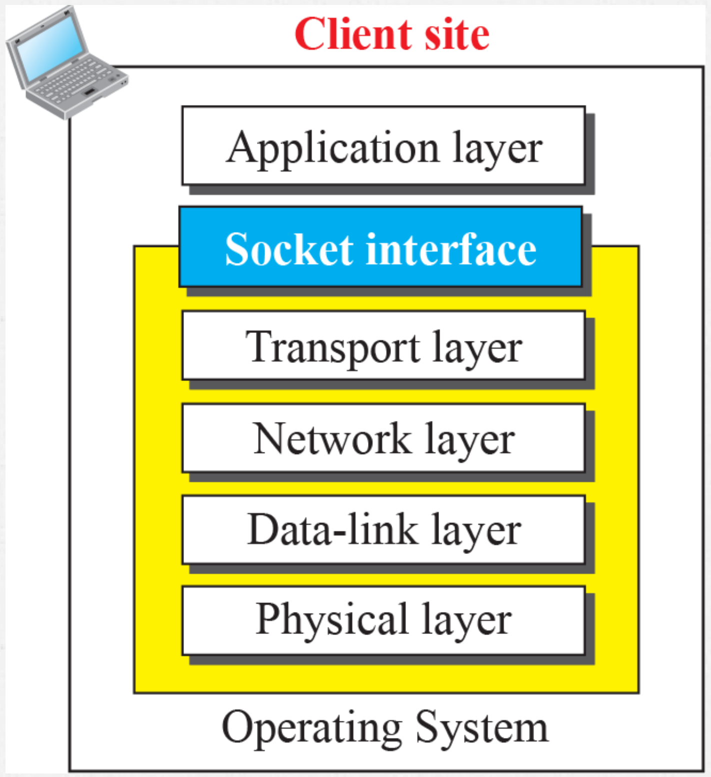
If we need a process(at AL) to be able to communicate with another process(i.e. OS), we need a new set of instructions to tell the lowest four layers of the TCP/IP suite to open the connection, send and receive data from the other end, and close the connection. A set of instructions of this kind is normally referred to as **Application Programming Interface (API).**

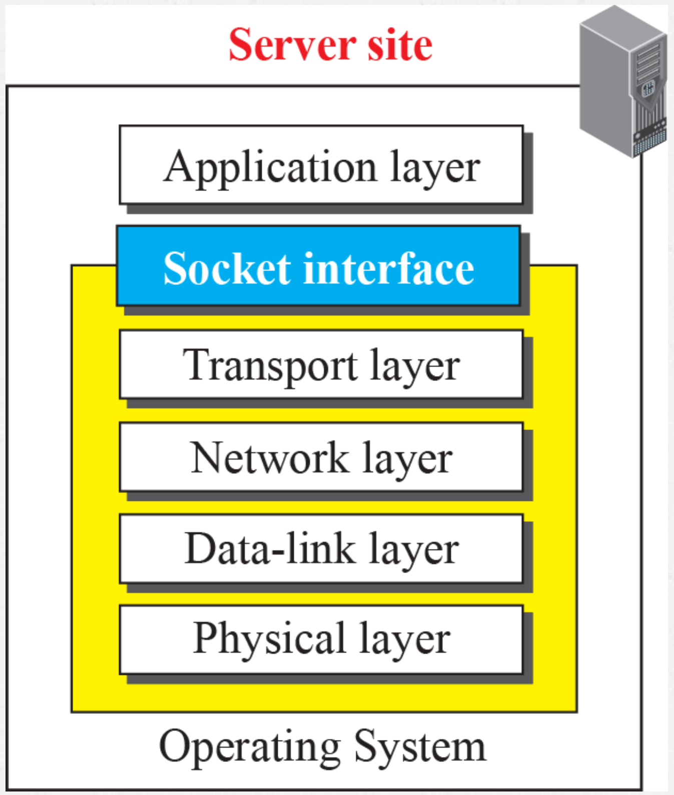
A Sockets used like other sources and sinks



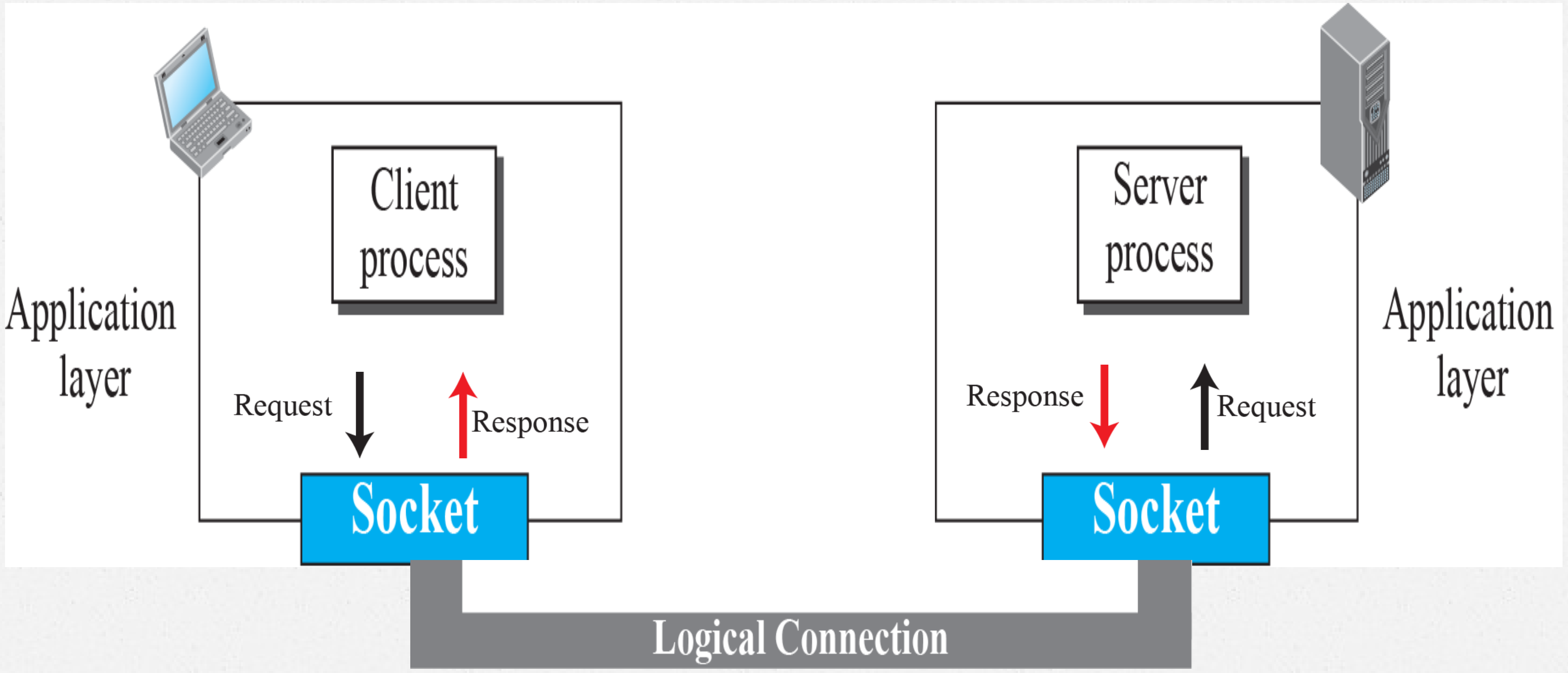
- ❑ Socket
- ❑ Socket Addresses
- ❑ Finding Socket Addresses
  - ❖ Server Site
  - ❖ Client Site

Position of the socket interface

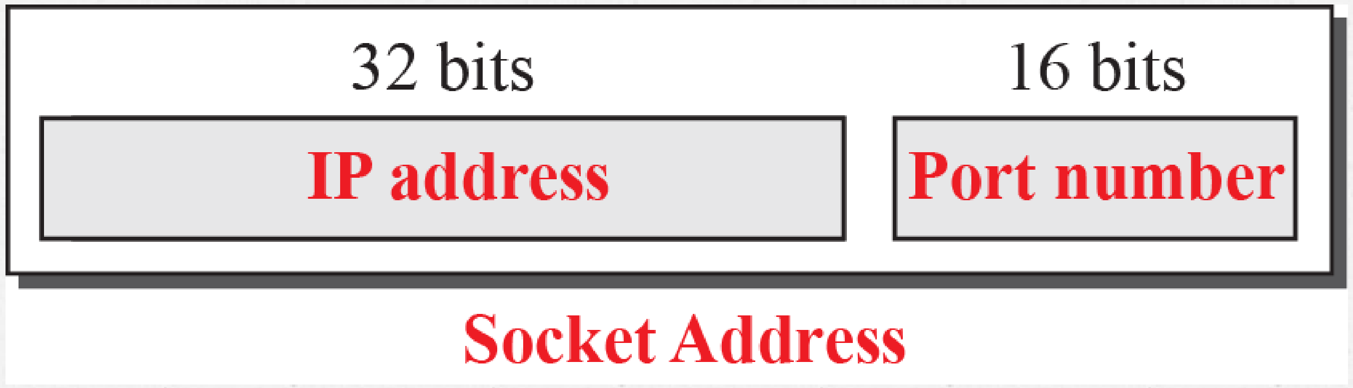




Use of sockets in process-to-process communication



A socket address



We can find a two-level address in telephone communication. A telephone number can define an organization, and an extension can define a specific connection in that organization. The telephone number in this case is similar to the IP address, which defines the whole organization; the extension is similar to the port number, which defines the particular connection.

Using Services of Transport Layer

A pair of processes provide services to the users of the Internet, human or programs. A pair of processes, however, 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.

- UDP Protocol
- TCP Protocol
- SCTP Protocol

STANDARD CLIENT-SERVER APPLICATIONS

During the lifetime of the Internet, several application programs have been developed. We do not have to redefine them, but we need to understand what they do. For each application, we also need to know the options available to us. The study of these applications can help us to create customized applications in the future.

World Wide Web and HTTP

In this section, we first introduce the World Wide Web (abbreviated WWW or Web). We then discuss the Hyper Text Transfer Protocol (HTTP), the most common client-server application program used in relation to the Web.

Founded 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

2.3.1 *(continued)*

- World Wide Web
  - Architecture –Distributed, linked
  - Web page
  - Hypertext / Hypermedia
  - Web Client(Browser)
  - Web Server
  - Uniform Resource Locator (URL)
  - Web Documents
    - Static
    - Dynamic
    - Active

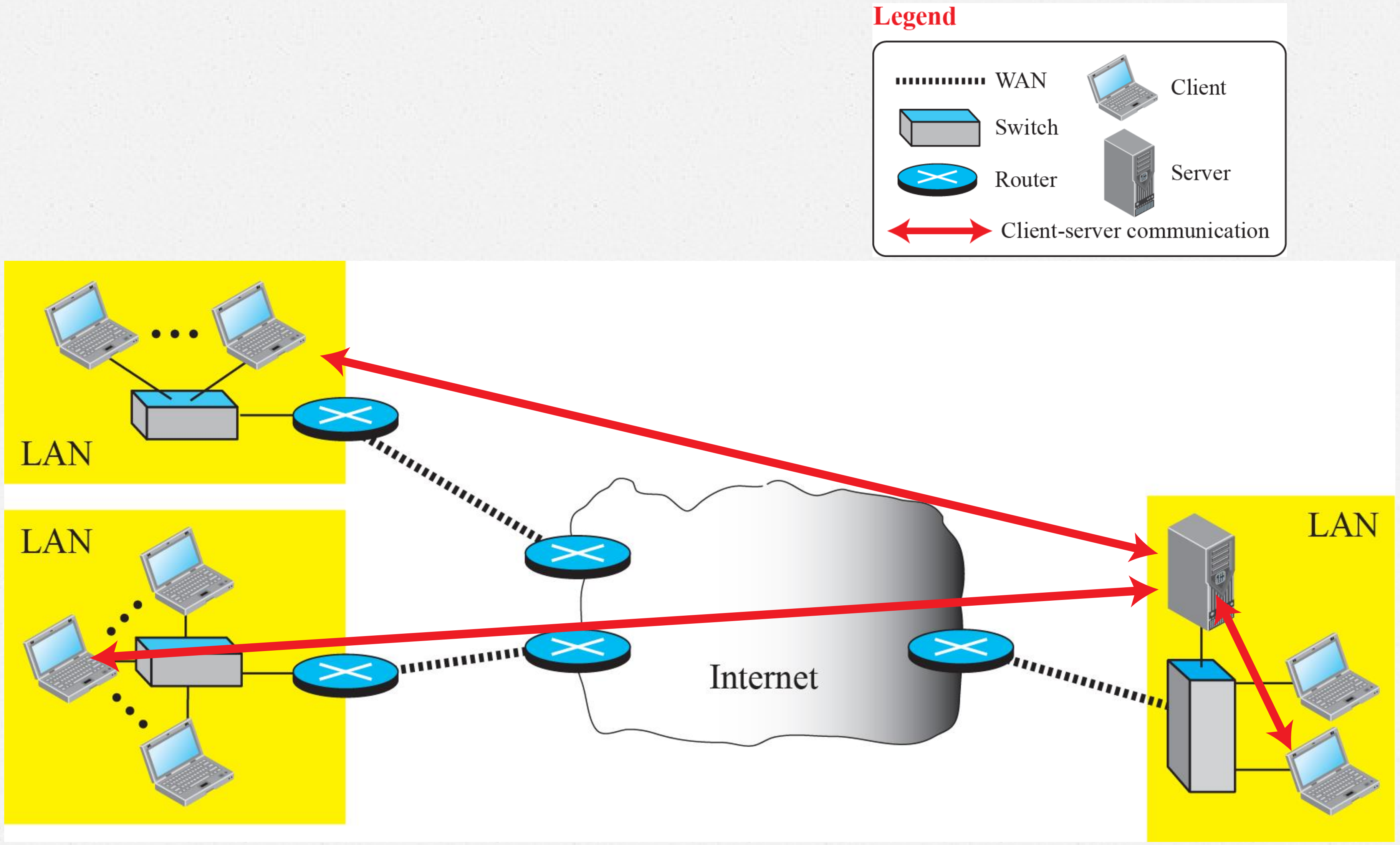
2.3.1 *(continued)*

- HyperText Transfer Protocol (HTTP)
  - Nonpersistent versus Persistent Connections
  - Message Formats
  - Conditional Request
  - Cookies

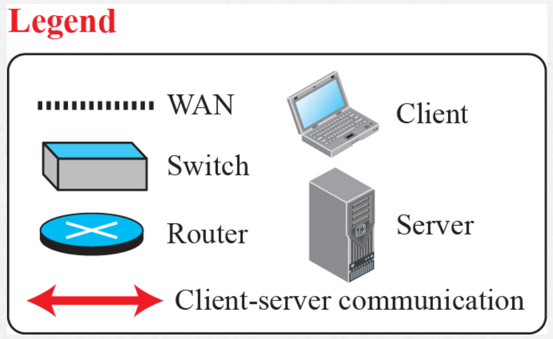
2.3.1 (continued)

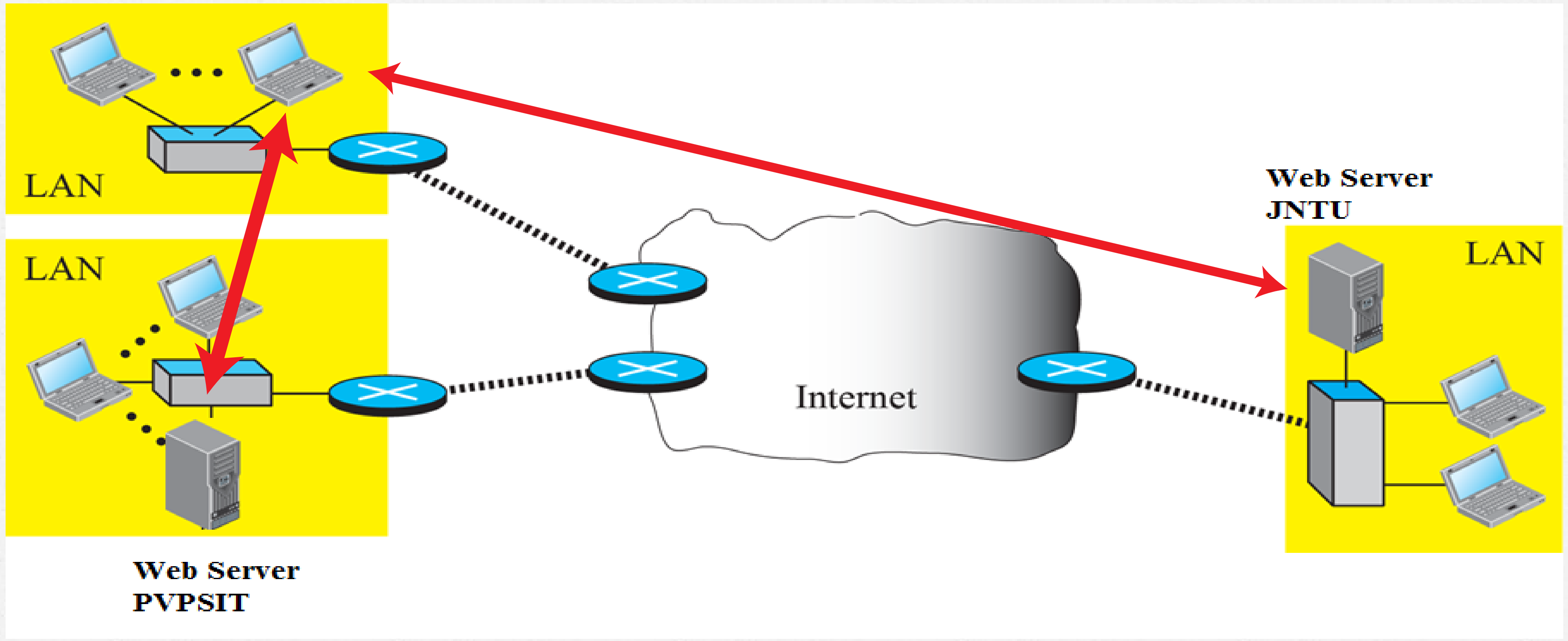
- ❑ Web Caching: Proxy Server
  - ❖ Proxy Server Location
  - ❖ Cache Update
- ❑ HTTP Security

Example of a client-server paradigm Single server



Example of a client-server paradigm multiple web servers

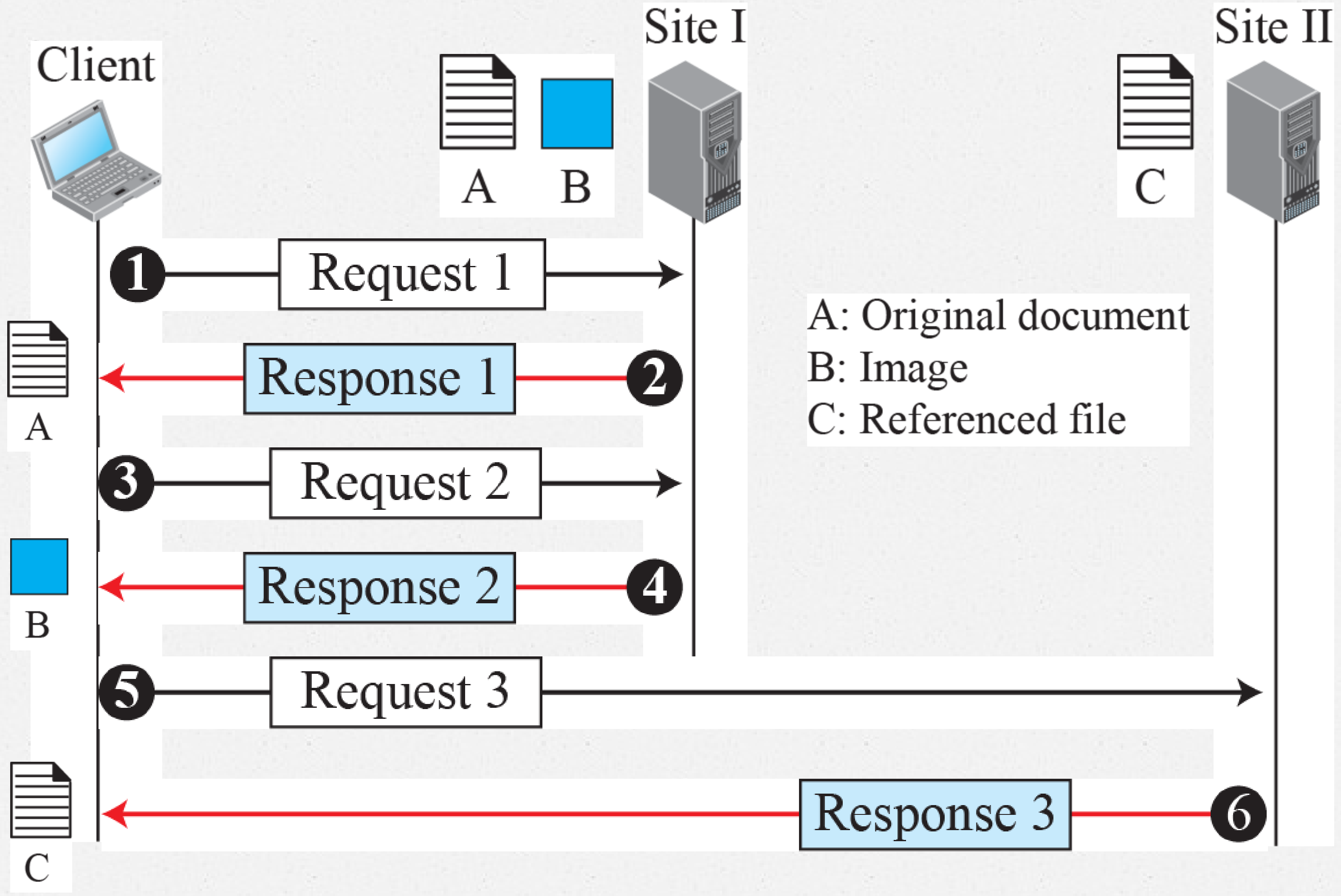




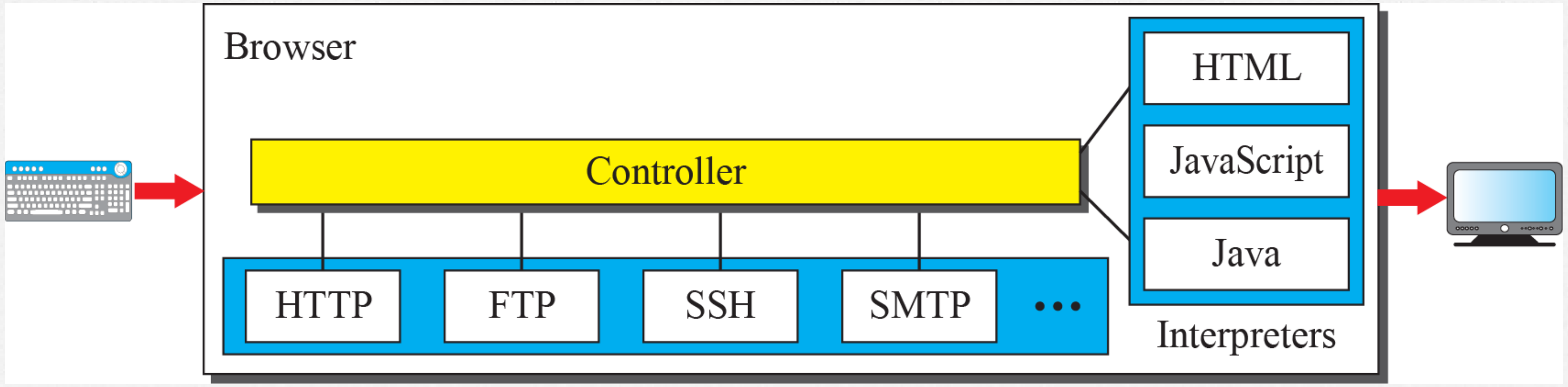
Assume we need to retrieve a scientific document that contains one reference to another text file and one reference to a large image. Figure 2.8 shows the situation.

The main document and the image are stored in two separate files in the same site (file A and file B); the referenced text file is stored in another site (file C). Since we are dealing with three different files, we need three transactions if we want to see the whole document.

Retrieving two files and one image

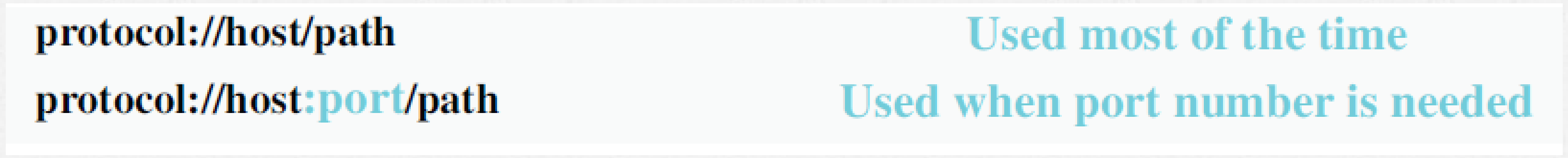


Browser



- controller
- client protocols
- interpreters

Uniform Resource Locator (URL)



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

- Protocol
- Host
- Port
- Path

The URL **http://www.mhhe.com/compsci/forouzan/** defines the web page related to one of the 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).

Web Documents

Static documents

- fixed-content documents that are created and stored in a server. The client can get a copy of the document only.
- Hypertext Markup Language (HTML), Extensible Markup Language (XML), Extensible Style Language (XSL), and Extensible Hypertext Markup Language (XHTML)

Web Documents

Dynamic Documents

- created by a web server whenever a browser requests the document
- When a request arrives, the web server runs an application program or a script that creates the dynamic document
- the contents of a dynamic document may vary from one request to another
- 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

Web Documents

Active Documents

- a program or a script to be run at the client site.
- Java applets, a program written in Java on the server. It is compiled and ready to be run. The document is in bytecode (binary) format.
- Another way is to use JavaScripts but download and run the script at the client site.

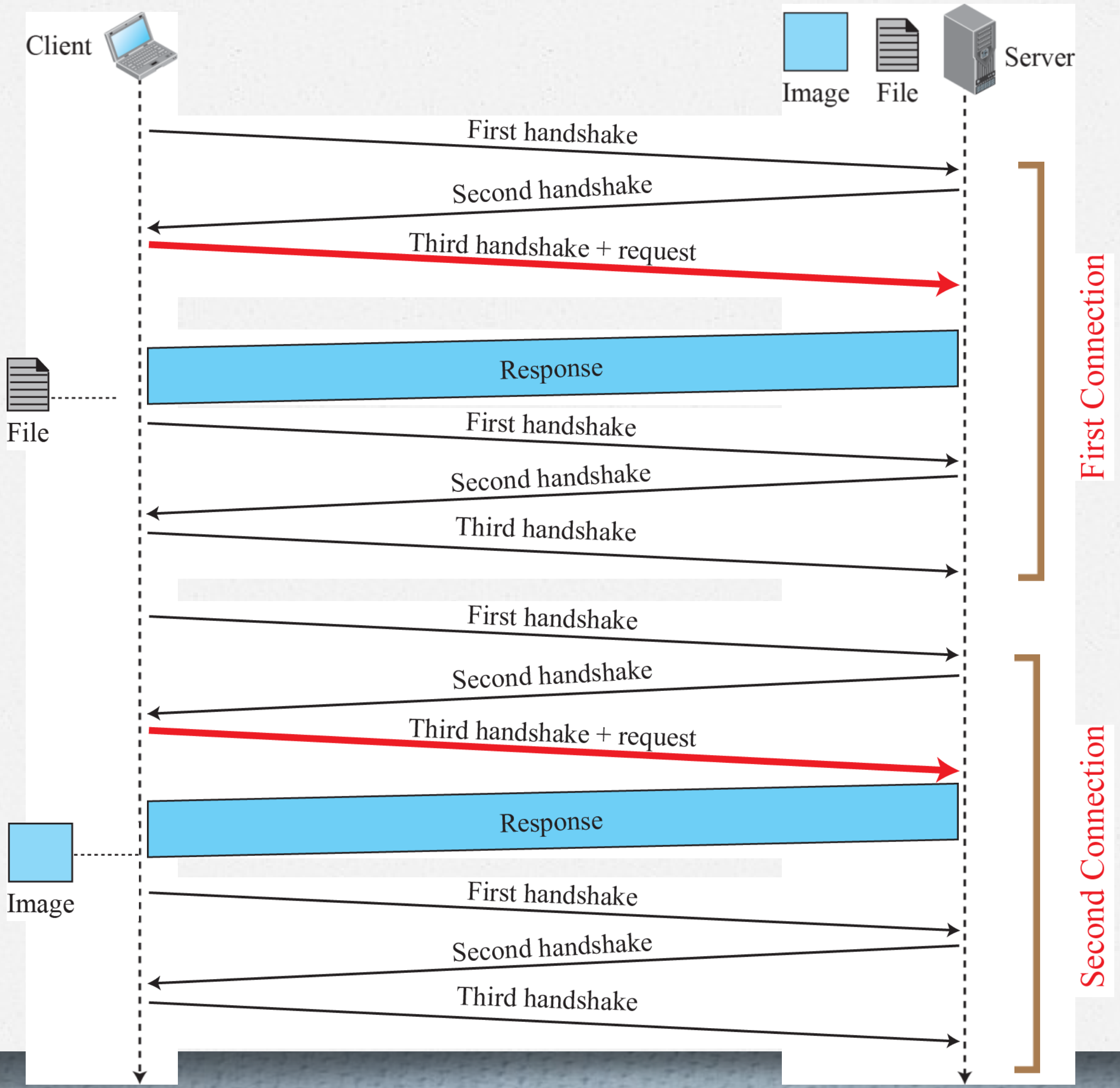
Hyper Text Transfer Protocol (HTTP)

- It is a protocol that is used to define how the client-server programs can be written to retrieve web pages from the Web.
- HTTP Client sends request and HTTP Server returns a response
- Server Port Number:80
- Client Port Number: temporary port Number
- HTTP is connection oriented i.e. uses TCP protocol which is connection oriented and reliable
- The type of connections to retrieve a web pages are:
  - Persistent Connection
  - Non Persistent Connection

Non-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. 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.

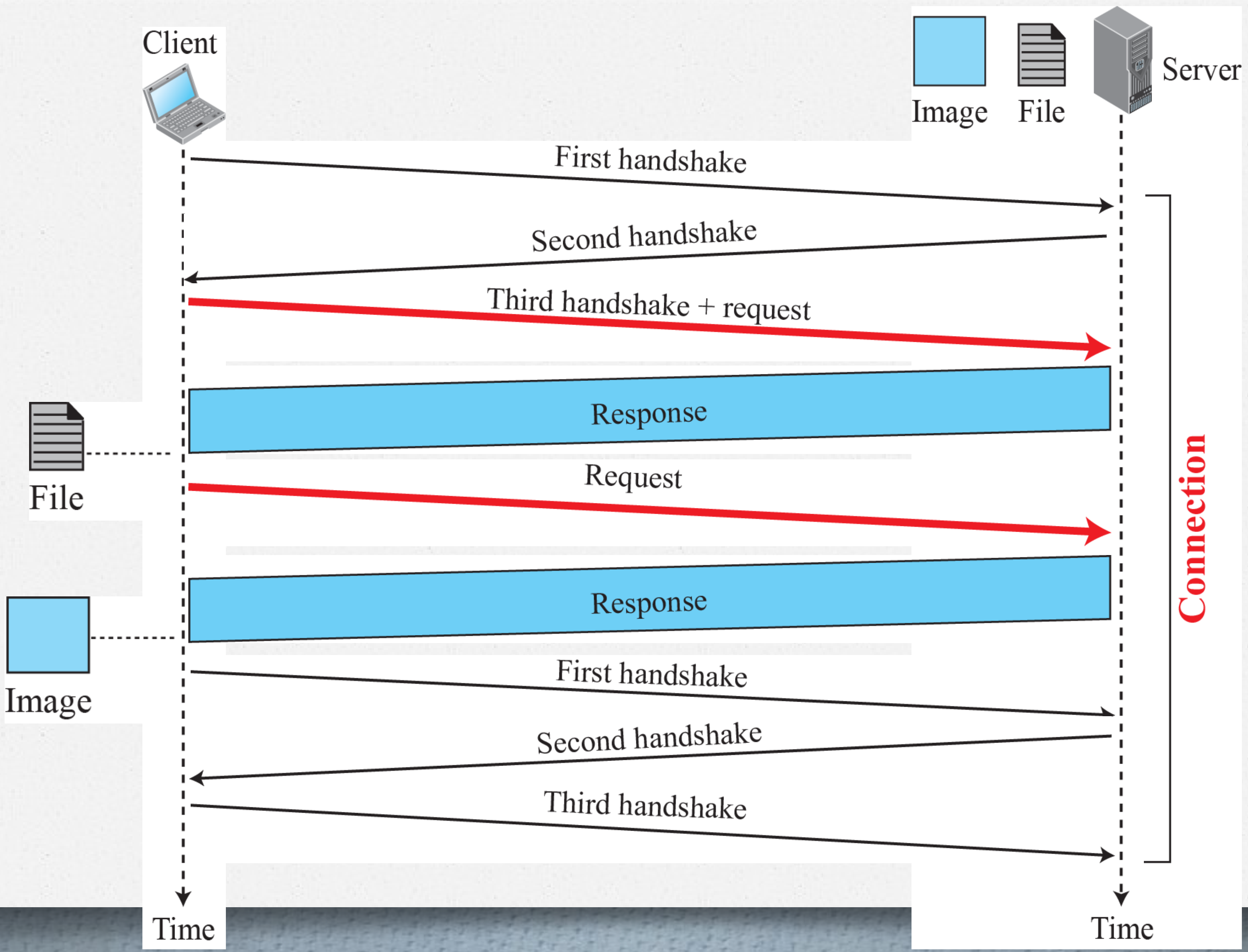
Example


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

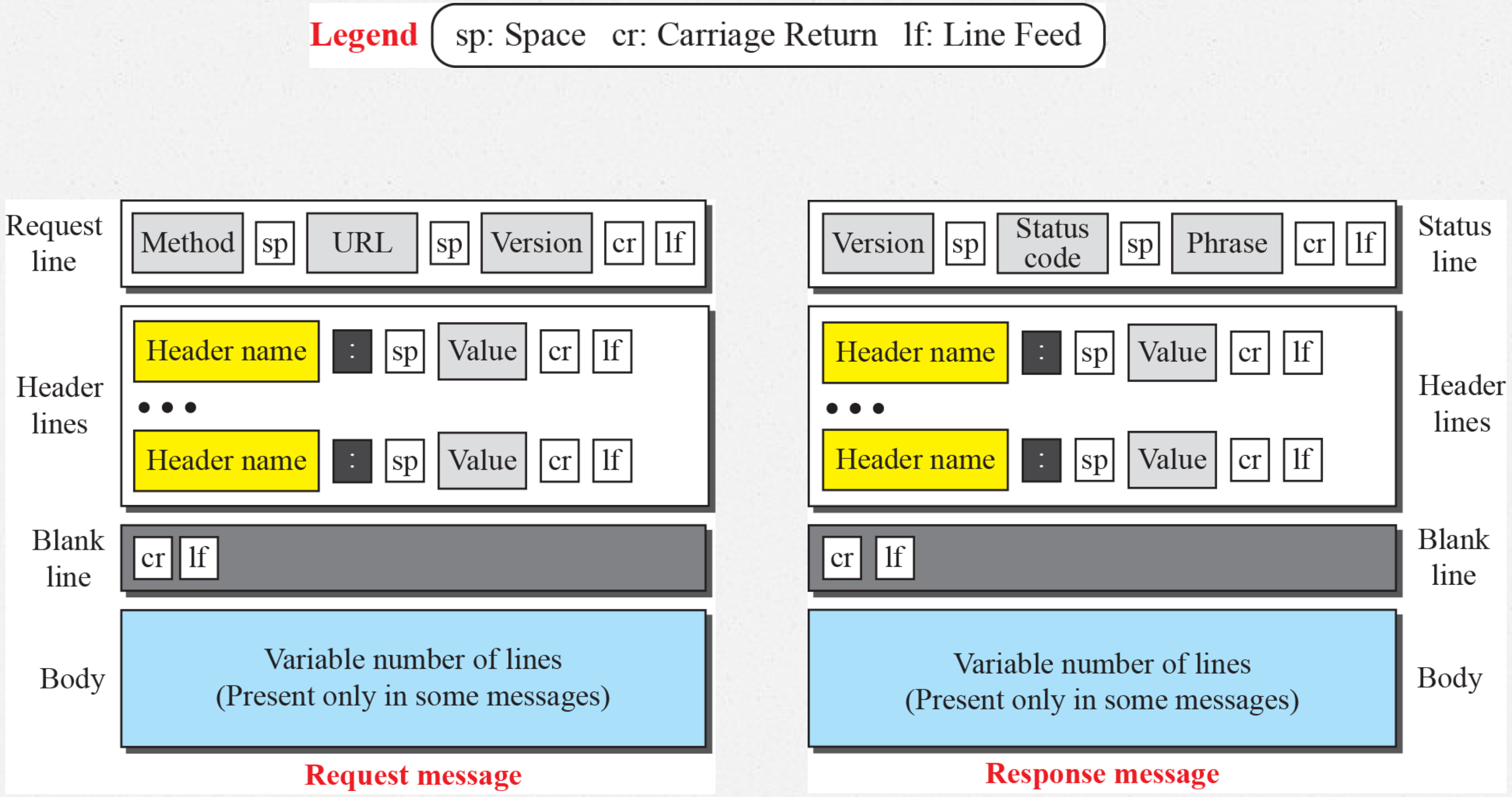
Persistent connection.

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

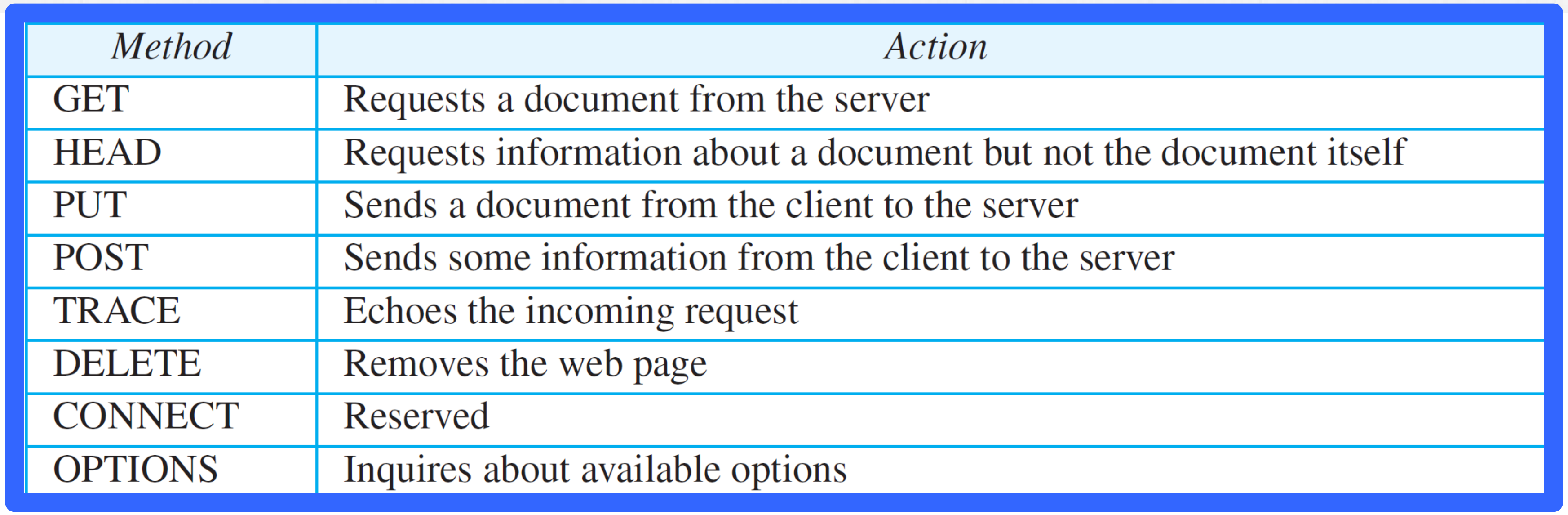
Example


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 of the request and response messages

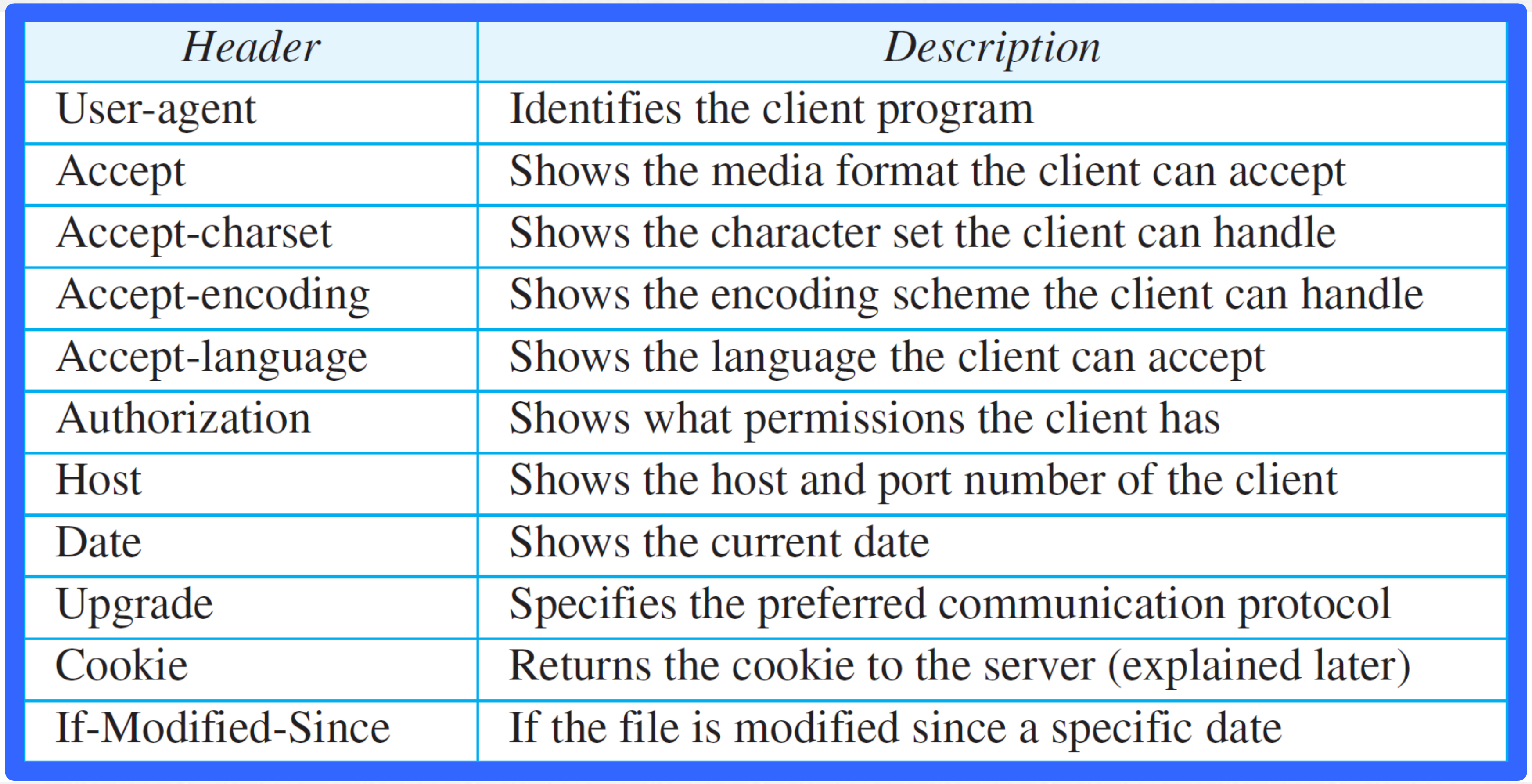

Legend
sp: Space cr: Carriage Return lf: Line Feed
Request line
Method sp URL sp Version cr lf
Header lines
Header name : sp Value cr lf
• • •
Header name : sp Value cr lf
Blank line
cr lf
Body
Variable number of lines
(Present only in some messages)
Request message
Status line
Version sp Status code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

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

Request Header Names



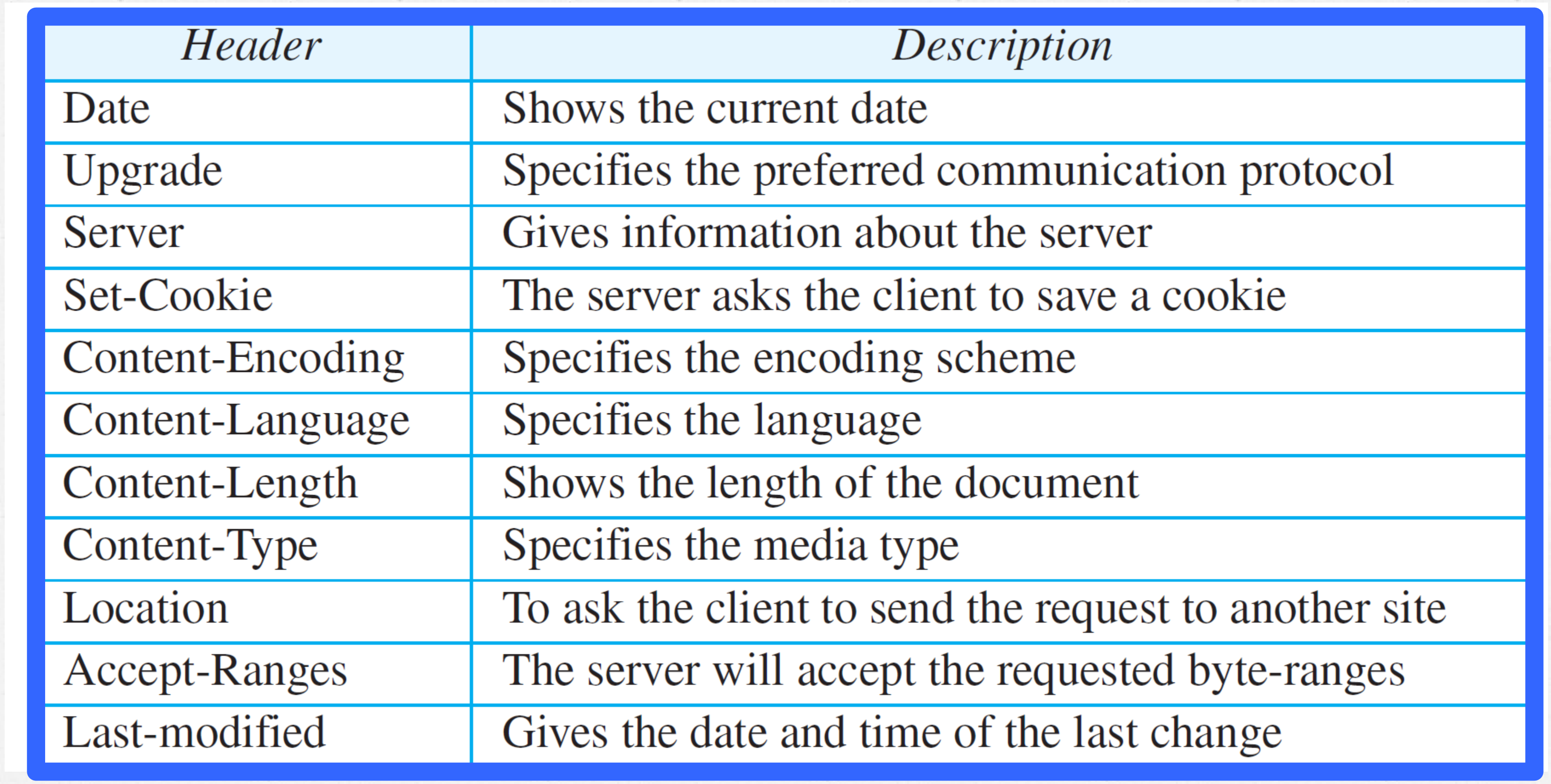
Header	*Description*
User-agent	Identifies the client program
Accept	Shows the media format the client can accept
Accept-charset	Shows the character set the client can handle
Accept-encoding	Shows the encoding scheme the client can handle
Accept-language	Shows the language the client can accept
Authorization	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

Status codes

100 range are only informational,
200 range indicate a successful request.
300 range redirect the client to another URL,
400 range indicate an error at the client site.
500 range indicate an error at the server site.

The status phrase explains the status code in text form.

Response Header Names

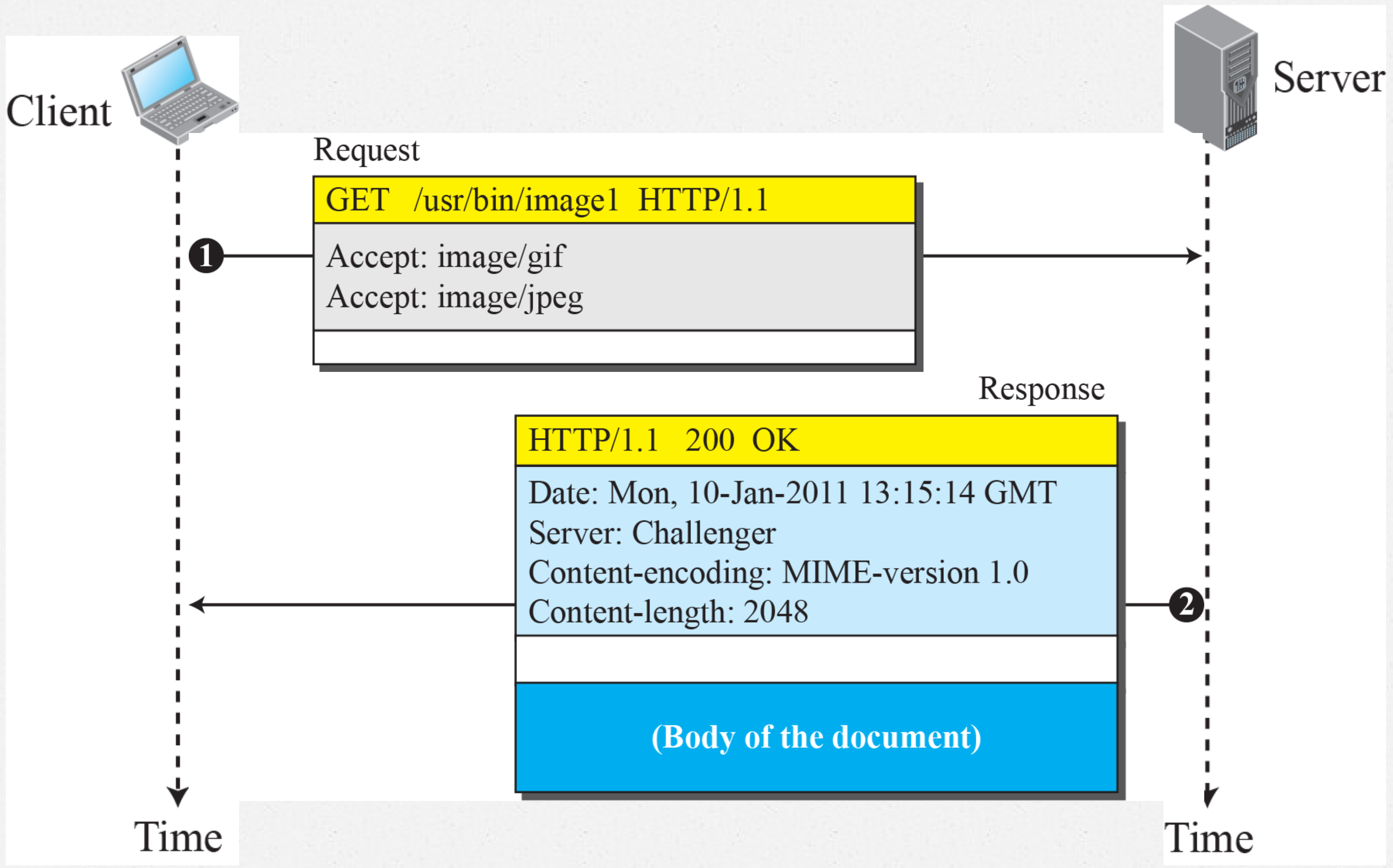


Header	*Description*
Date	Shows the current date
Upgrade	Specifies the preferred communication protocol
Server	Gives information about the server
Set-Cookie	The server asks the client to save a cookie
Content-Encoding	Specifies the encoding scheme
Content-Language	Specifies the language
Content-Length	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

This example retrieves a document. 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.

Example

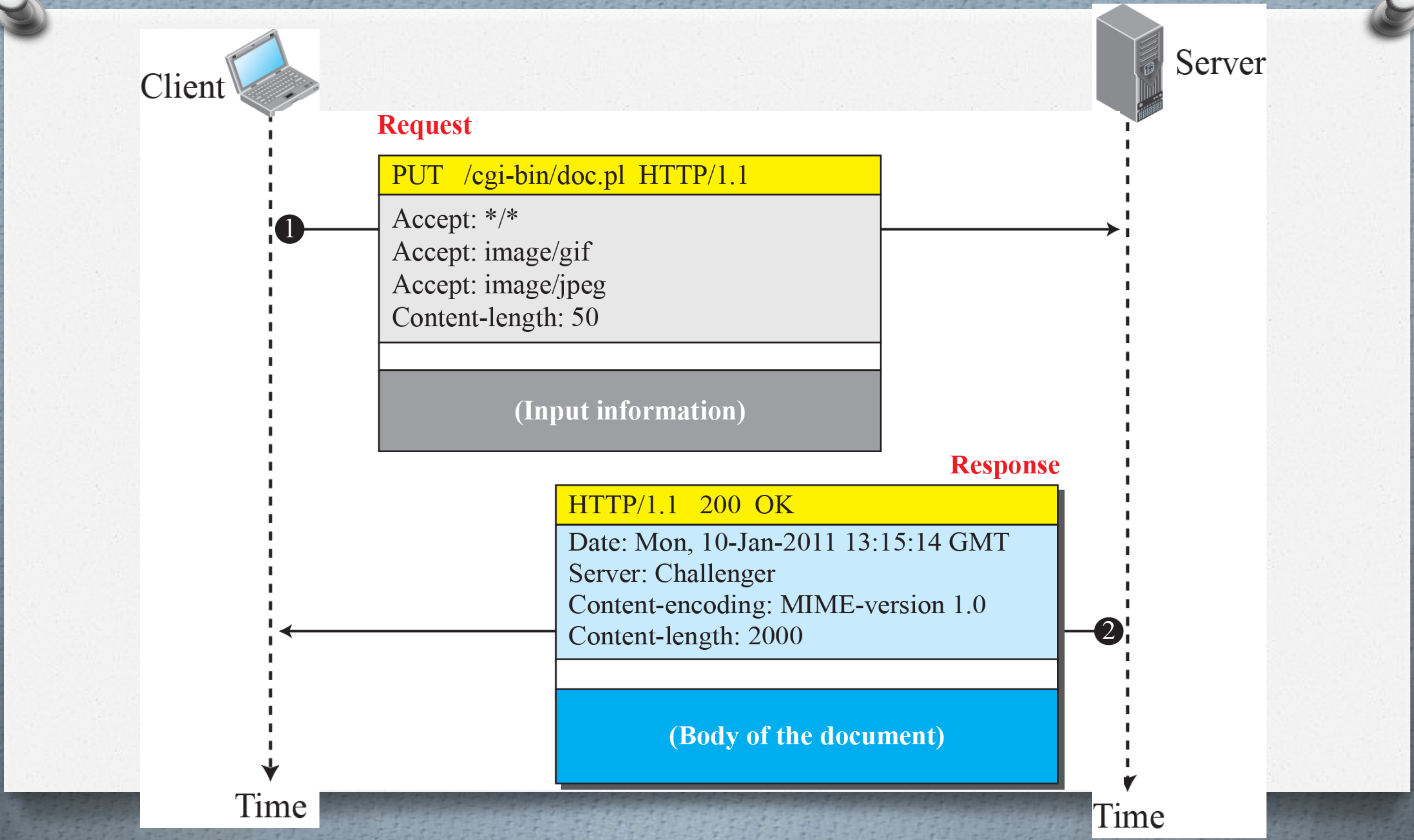


```
GET   /usr/bin/image1  HTTP/1.1
Accept: image/gif
Accept: image/jpeg
```

```
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

(Body of the document)
```

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.

Example 2

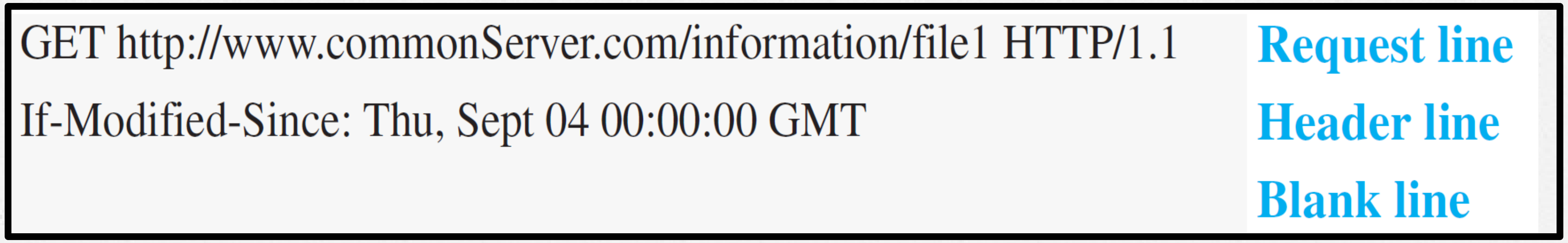


```
PUT   /cgi-bin/doc.pl  HTTP/1.1
Accept: */*
Accept: image/gif
Accept: image/jpeg
Content-length: 50

(Input information)
```

```
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

(Body of the document)
```

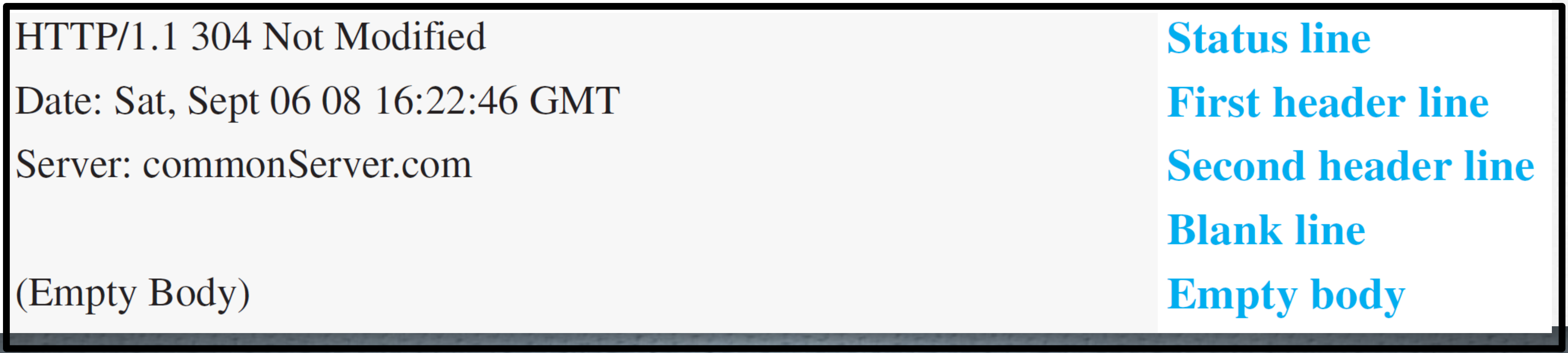
Conditional Request

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
- **Cookies** are messages that **web** servers pass to your **web** browser when you visit **Internet** sites.
- **cookie** is a small piece of data stored on the user's computer by the **web** browser
- Cookies are most commonly used to track website activity.
- When you visit some sites, the server gives you a cookie that acts as your identification card. Upon each return visit to that site, your browser passes that cookie back to the server.

Creating & 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.

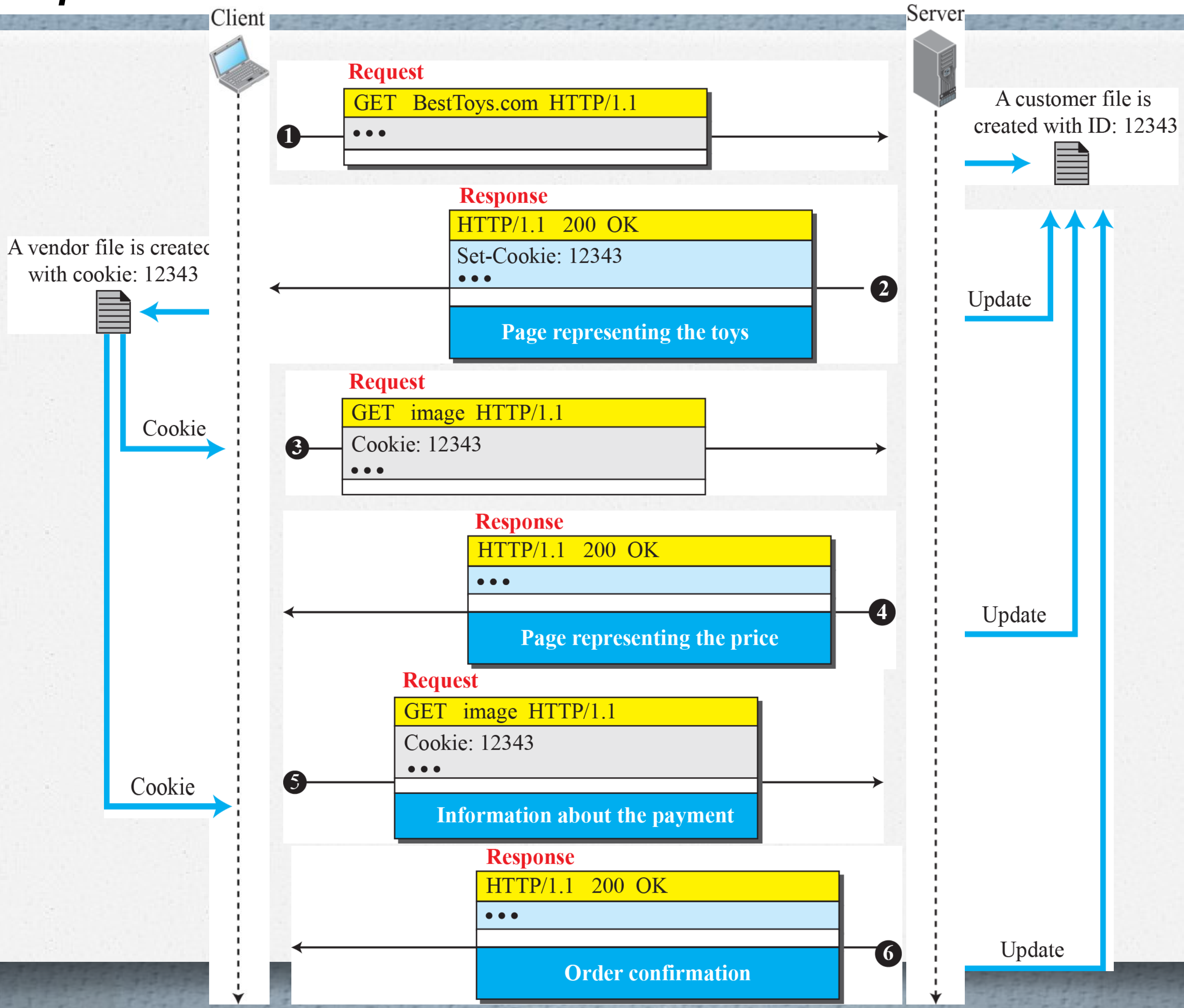
Using Cookies

- 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

Cookies

Figure 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.

Example

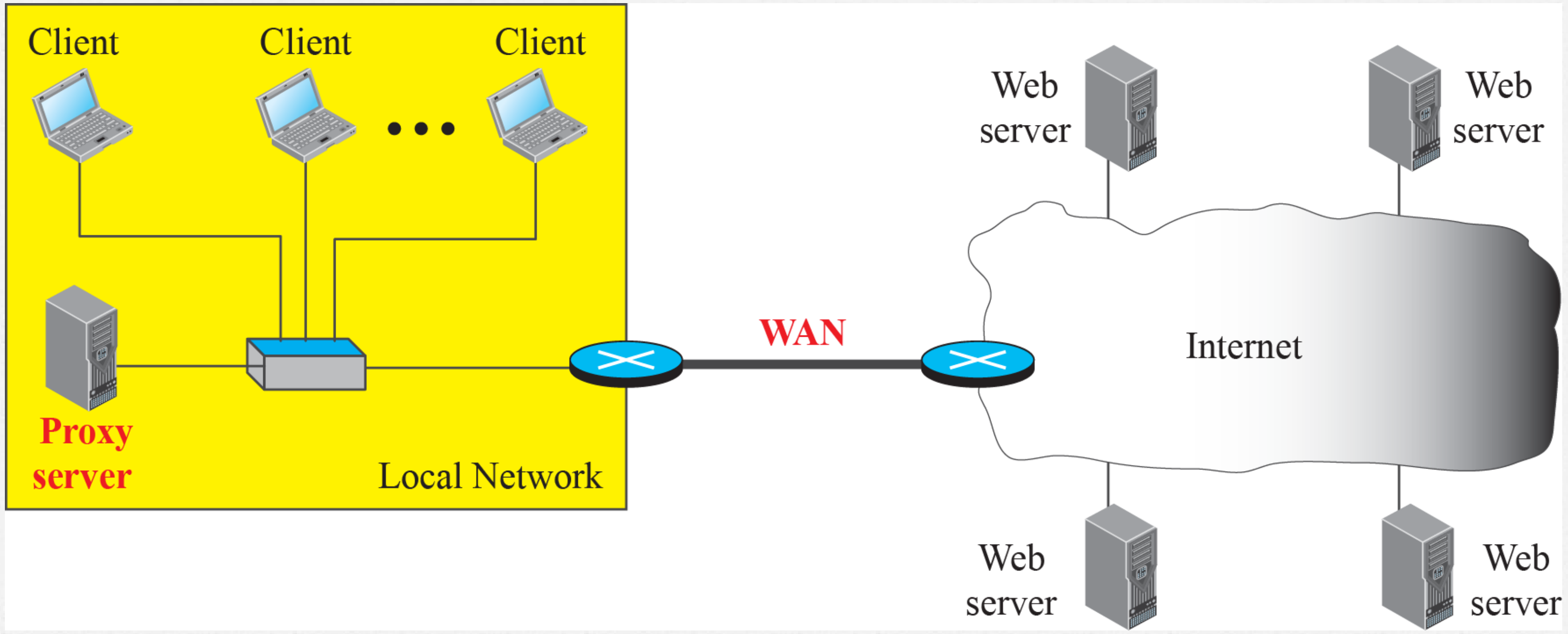

Client
Server
Request
GET BestToys.com HTTP/1.1
• • •
1
A customer file is created with ID: 12343
Response
HTTP/1.1 200 OK
Set-Cookie: 12343
• • •
Page representing the toys
2
A vendor file is createc with cookie: 12343
Update
Request
GET image HTTP/1.1
Cookie: 12343
• • •
3
Cookie
Response
HTTP/1.1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

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**.
- The proxy server acts as both server and client.
- The proxy servers are normally located at the client site such as client system, LAN proxy server or at ISP.

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.

Example of a proxy server



Cache Update

- How long a response should remain in the proxy server before being deleted and replaced
  - store the list of sites whose information remains the same for a while e.g. news page could be loaded every morning
  - to add some headers to show the last modification time of the information.

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.
- HTTPS provides confidentiality, client and server authentication, and data integrity.