Chapter 27

WWW and HTTP

27-1 ARCHITECTURE

The WWW today is a distributed client/server service, in which a client using a browser can access a service using a server. However, 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.

Figure 27.1 *Architecture of WWW*

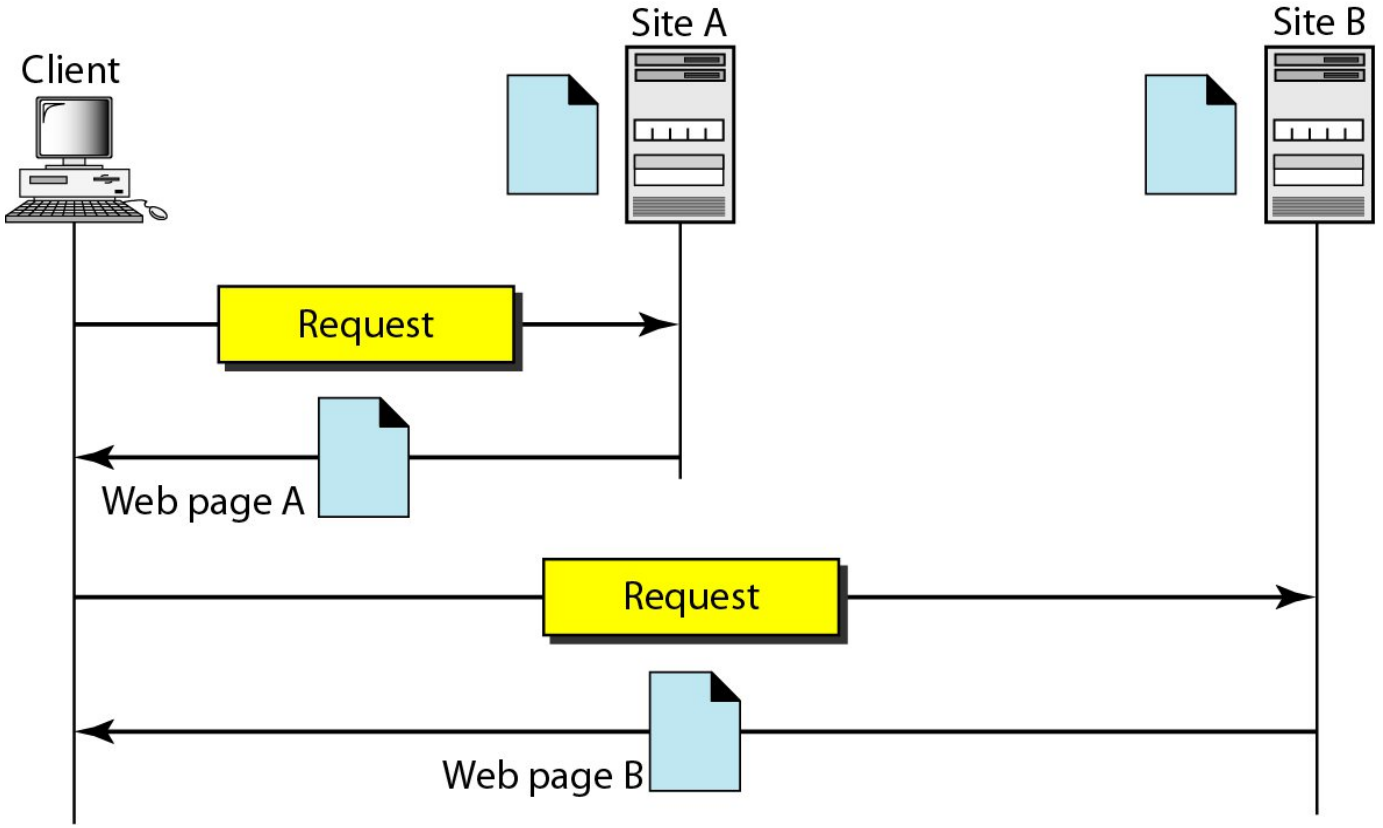


Figure 27.2 *Browser*

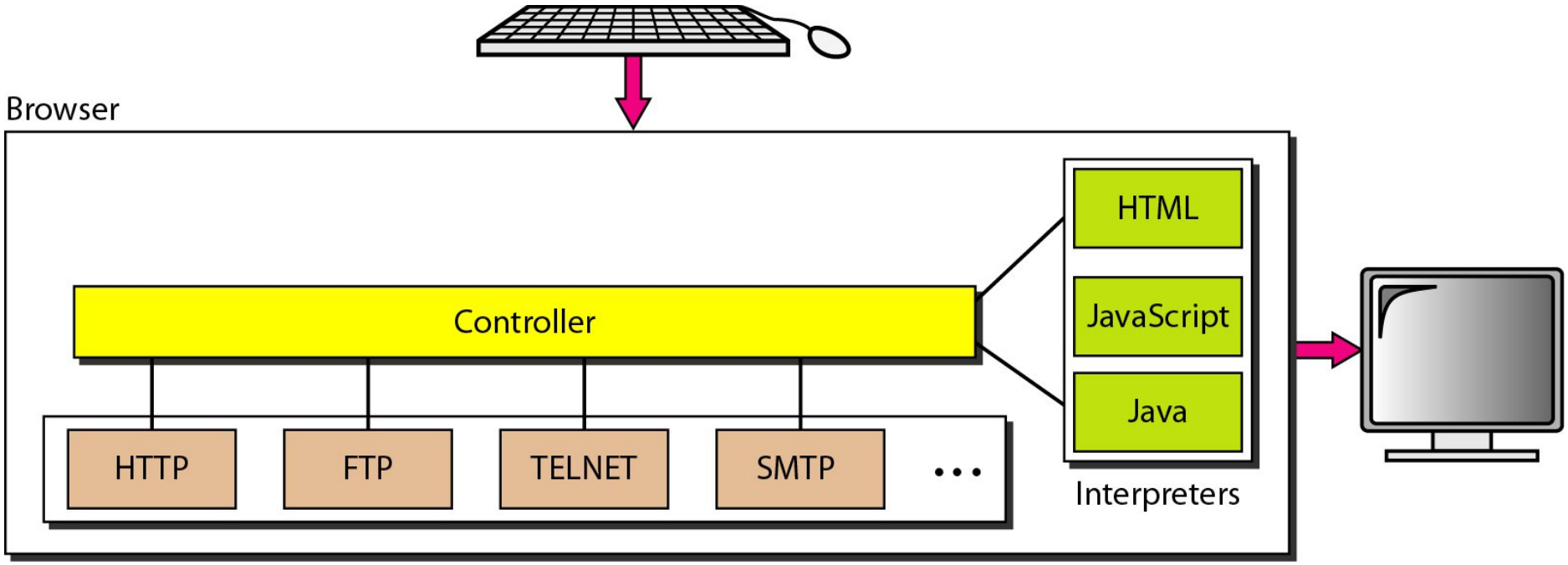
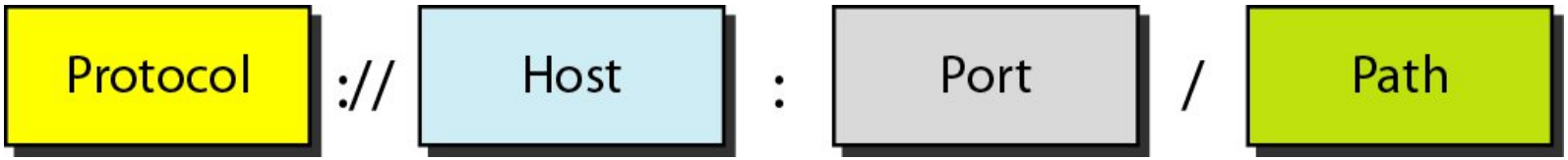


Figure 27.3 *URL*



- The ***protocol*** is the client-server application program used to retrieve the document.
- Many different protocols can retrieve a document; among them are Gopher, FTP, HTTP, News, and TELNET. The most common today is HTTP.
- The **host** is the domain name of the computer on which the information is located.
- The **URL** can optionally contain the port number of the server.
- **Path** is the pathname of the file where the information is located.

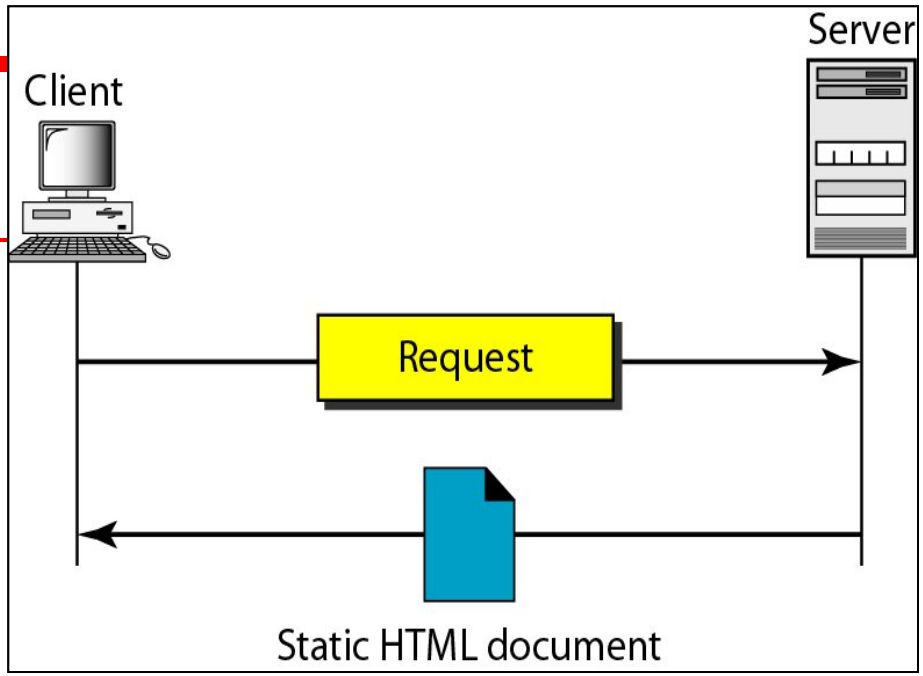
27-2 WEB DOCUMENTS

The documents in the WWW can be grouped into three broad categories: static, dynamic, and active. The category is based on the time at which the contents of the document are determined.

Topics discussed in this section:

Static Documents
Dynamic Documents
Active Documents

Figure 27.4 *Static document*



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 **Extended Hypertext Markup Language (XHTML)**.

Dynamic Documents

- A **dynamic document** is 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 server returns the output of the program or script as a response to the browser that requested the document

- A very simple example of a dynamic document is the retrieval of the time and date from a server.

- The **Common Gateway Interface (CGI)** is a technology that creates and handles dynamic documents.

- CGI is a set of standards that defines how a dynamic document is written, how data are input to the program, and how the output result is used.

- CGI defines is a set of rules and terms that the programmer must follow.

Dynamic Documents : CGI

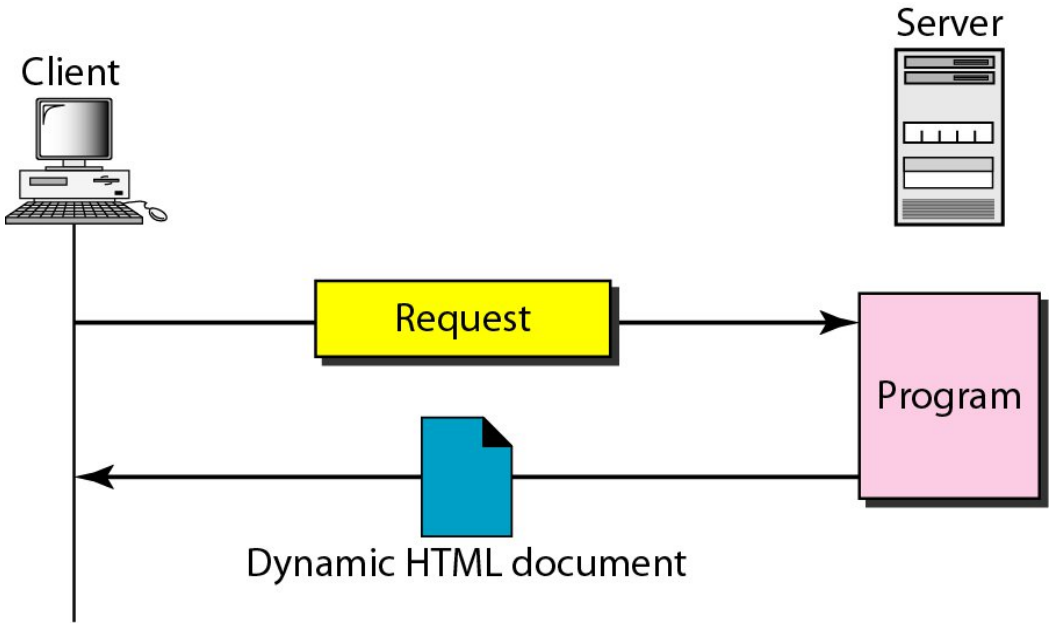
- The **Common Gateway Interface (CGI)** is a technology that creates and handles dynamic documents.

- CGI is a set of standards that defines how a dynamic document is written, how data are input to the program, and how the output result is used.

- CGI defines is a set of rules and terms that the programmer must follow.

- The term gateway here means that a CGI program can be used to access other resources such as databases, graphic packages, and so on.
- The term interface here means that there is a set of predefined terms, variables, calls, and so on that can be used in any CGI program.

Figure 27.8 *Dynamic document using CGI*



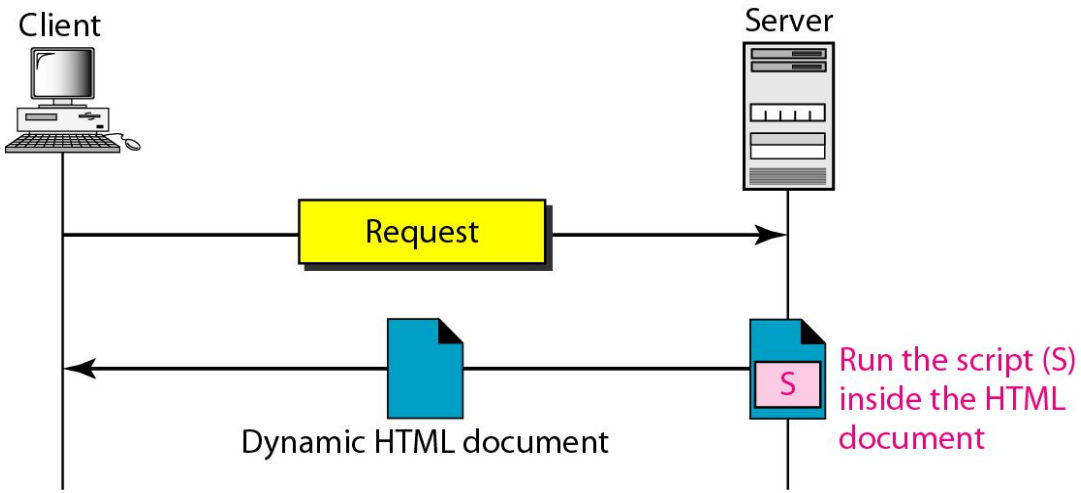
Scripting Technologies for Dynamic Documents

- problem with CGI technology is the inefficiency.

- Solution is to create a file containing the fixed part of the document using HTML and embed a script, a source code, that can be run by the server.

  Technologies involved in creating dynamic documents using scripts:
- Hypertext Preprocessor (PHP),
- Java Server Pages (JSP),
- Active Server Pages (ASP)
- ColdFusion,

Figure 27.9 *Dynamic document using server-site script*



Note

Dynamic documents are sometimes referred to as server-site dynamic documents.

Active Documents

- For many applications, we need a program or a script to be run at the client site. These are called **active documents.**

- 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.

Figure 27.10 *Active document using Java applet*

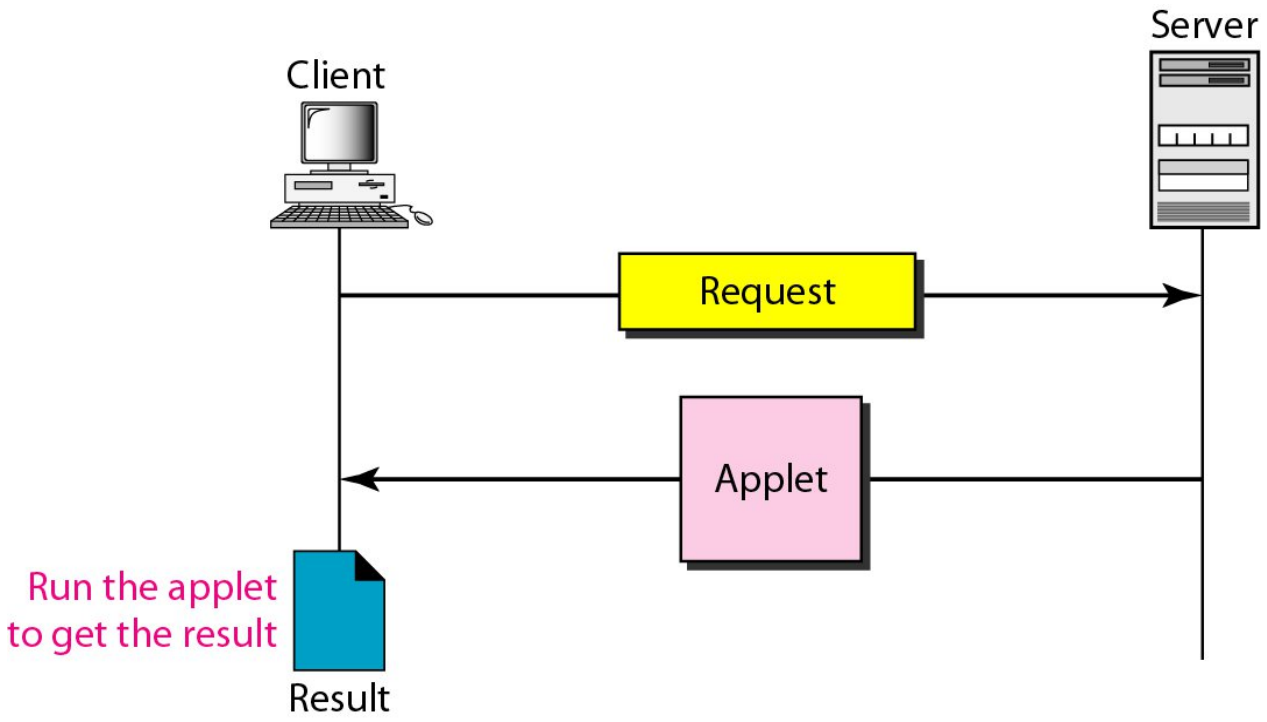
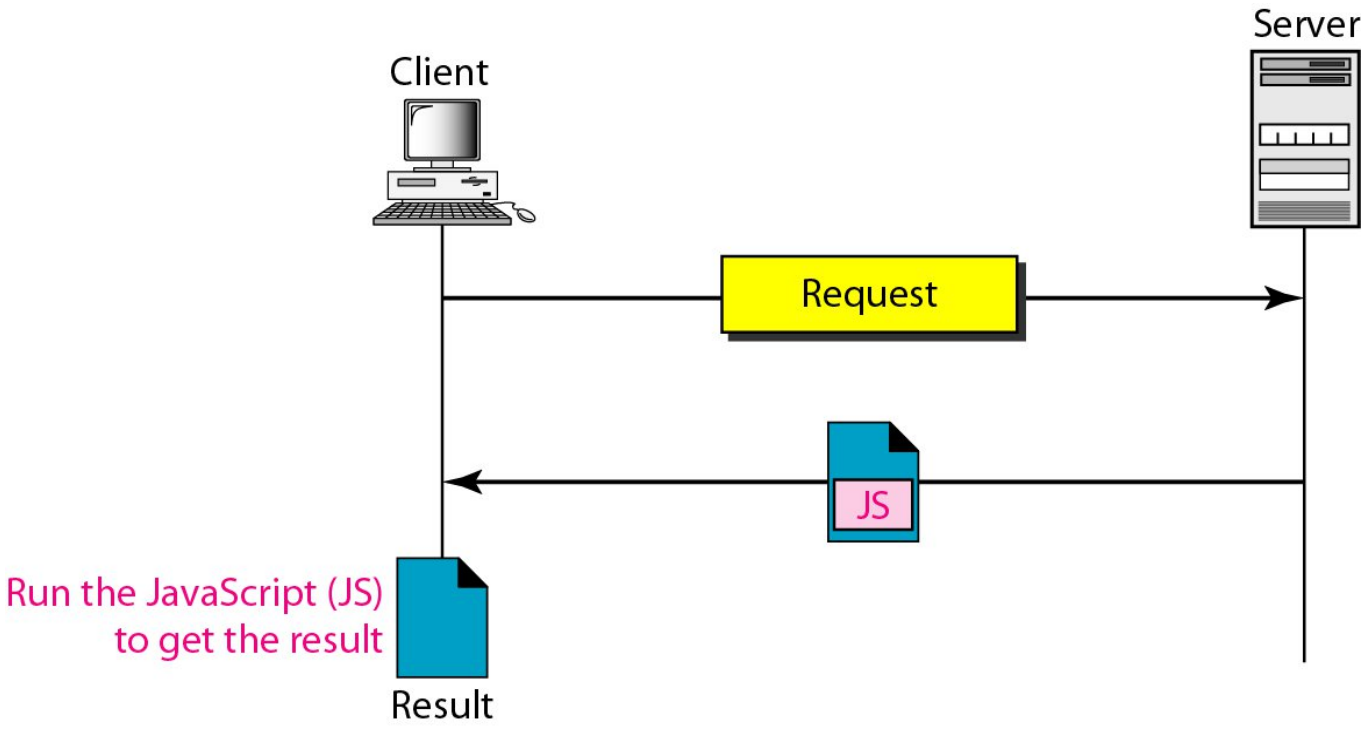
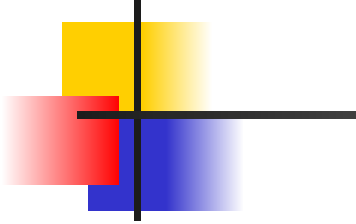


Figure 27.11 *Active document using client-site script*





Note

Active documents are sometimes referred to as client-site dynamic documents.

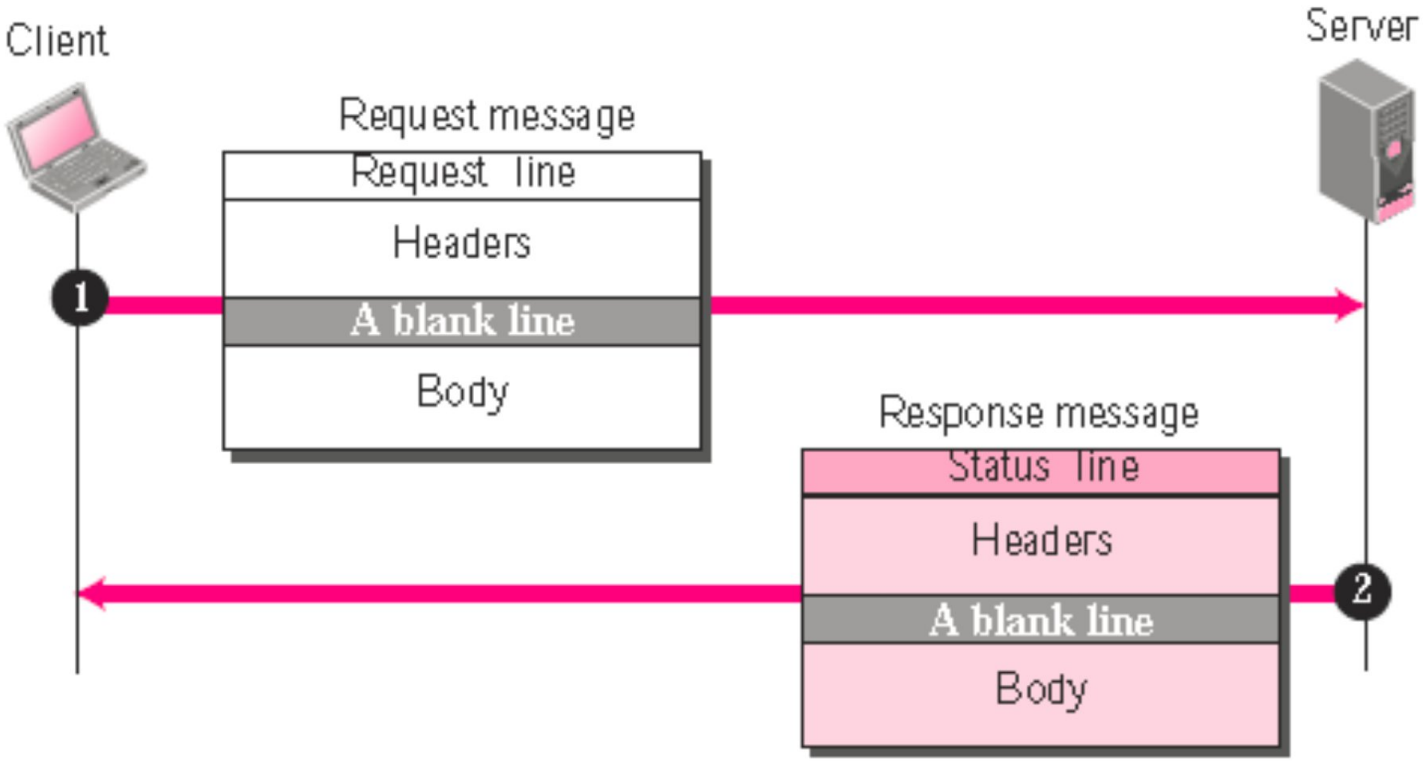
27-3 HTTP

- *The Hypertext Transfer Protocol (HTTP) is a protocol used mainly to access data on the World Wide Web. HTTP functions as a combination of FTP and SMTP.*
- *It is similar to FTP because it transfers files and uses the services of TCP.*
- *HTTP is like SMTP because the data transferred between the client and the server look like SMTP messages.*

Note

HTTP uses the services of TCP on well-known port 80.

Figure 27.12 *HTTP transaction*



HTTP Transaction

- HTTP uses the services of TCP, HTTP itself is a stateless protocol, which means that the server does not keep information about the client.

- The client initializes the transaction by sending a request. The server replies by sending a response.

Request Message

•A request message consists of a request line, a header, and sometimes a body.

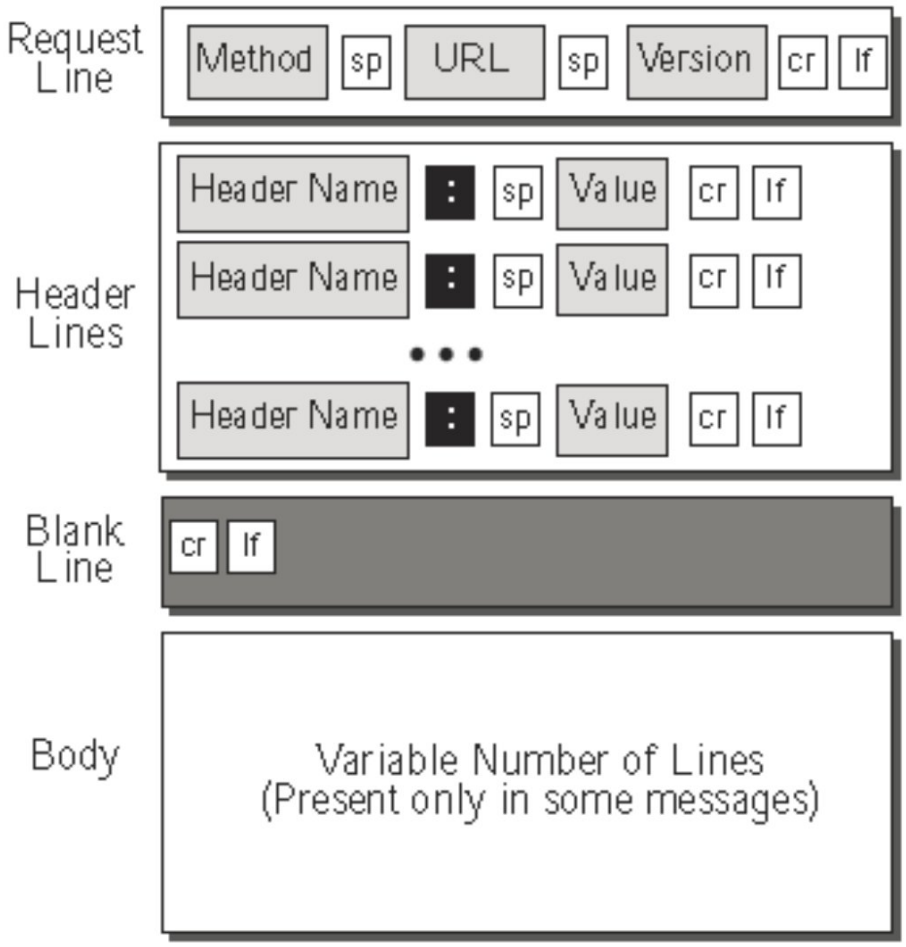
•**Request Line:** The first line in a request message is called a **request line.** There are three fields in this line separated by some character delimiter. The fields are called methods, URL, and Version.

•These three should be separated by a space character.

•At the end two characters, a carriage return followed by a line feed, terminate the line.

•The method field defines the **request type**.

Format of request message:



Legend

sp: Space
cr: Carriage Return
lf: Line Feed

Request Message (cont.)

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

Header Lines In Request Message

•After the request line, we can have zero or more **request header** lines. Each header line sends additional information from the client to the server.

•Each header line has a header name, a colon, a space, and a header value

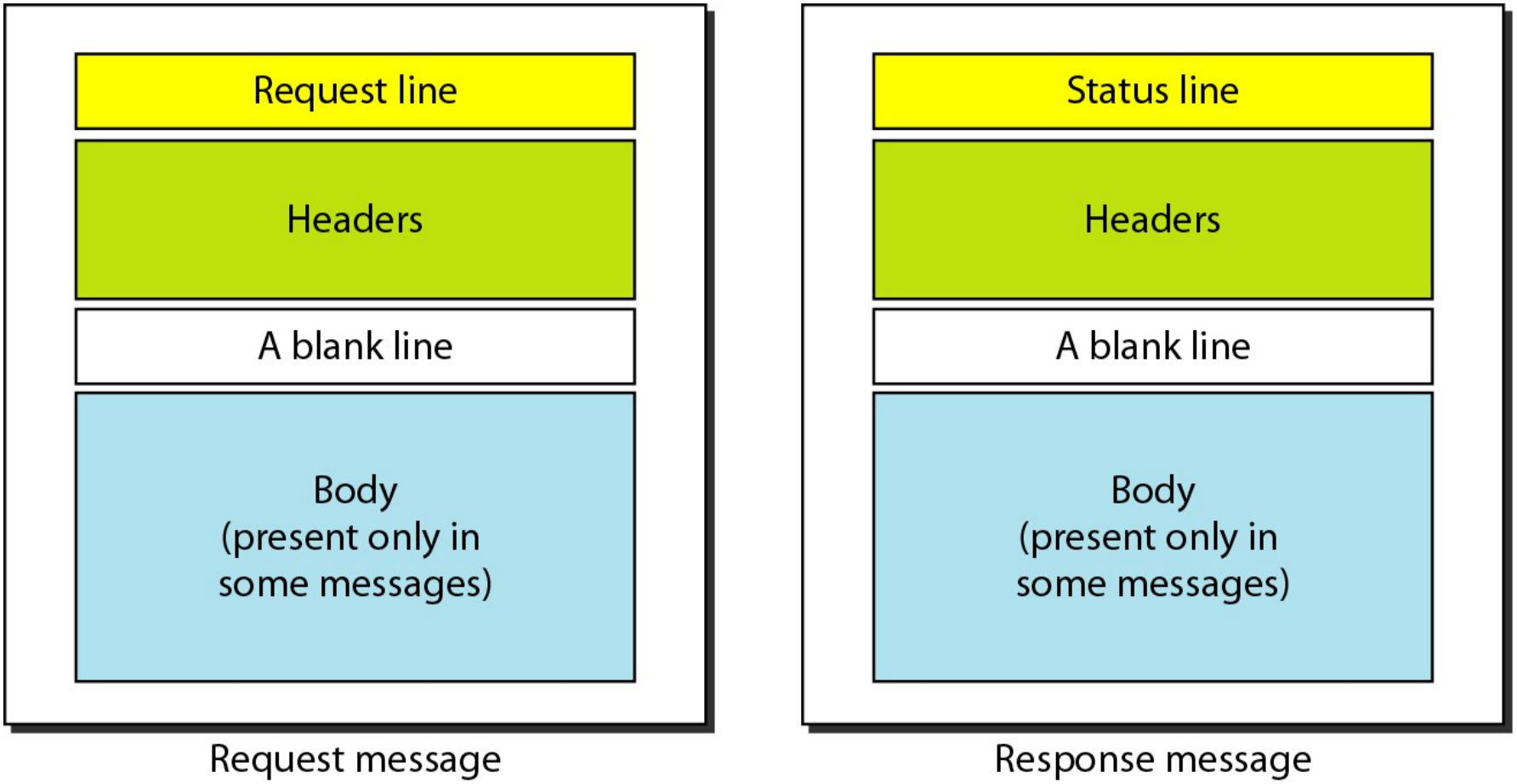
•The value field defines the values associated with each header name.

Body In Request Message The body can be present in a request message. Usually, it contains the comment to be sent.

Request Header Names

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

Figure 27.13 *Request and response messages*



Response Message

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

Status Line

- The first line in a response message is called the **status line**.
- There are three fields in this line separated by spaces and terminated by a carriage return and line feed.
- The first field defines the version of HTTP protocol, currently 1.1.
- The status code field defines the status of the request. It consists of three digits.

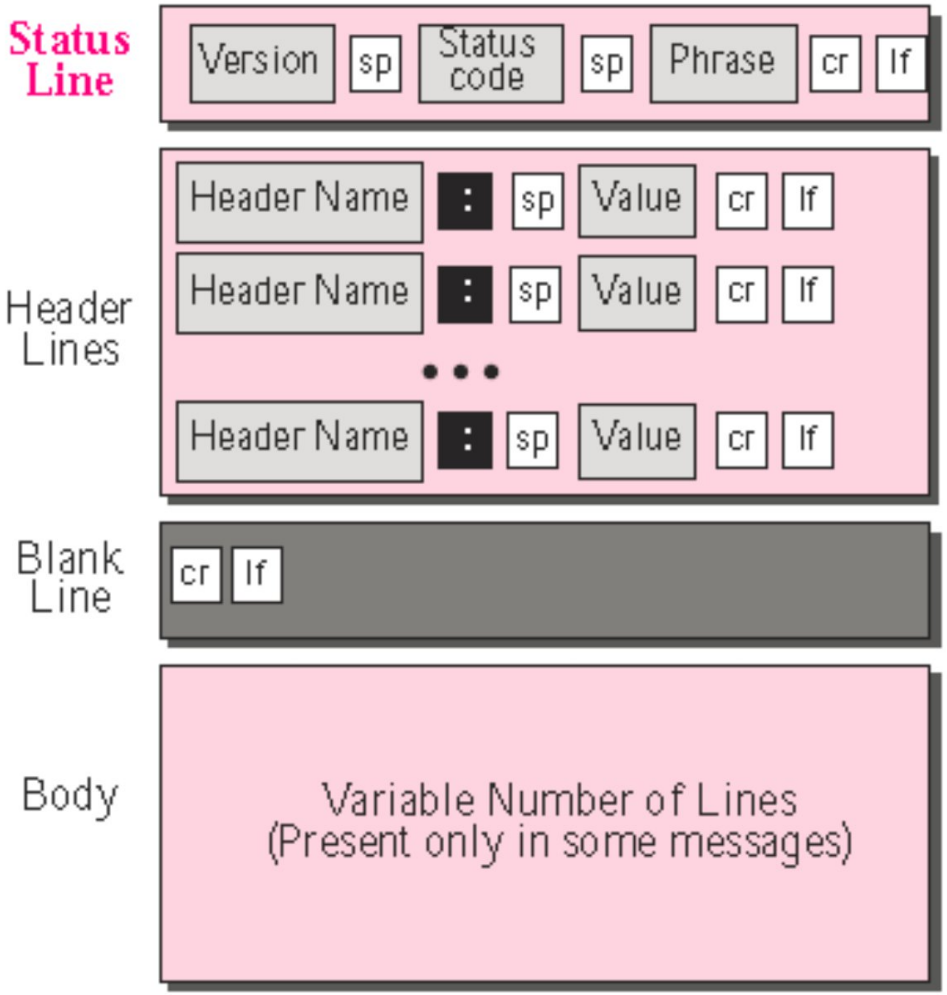
Header Lines In Response Message

- 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

Body

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.

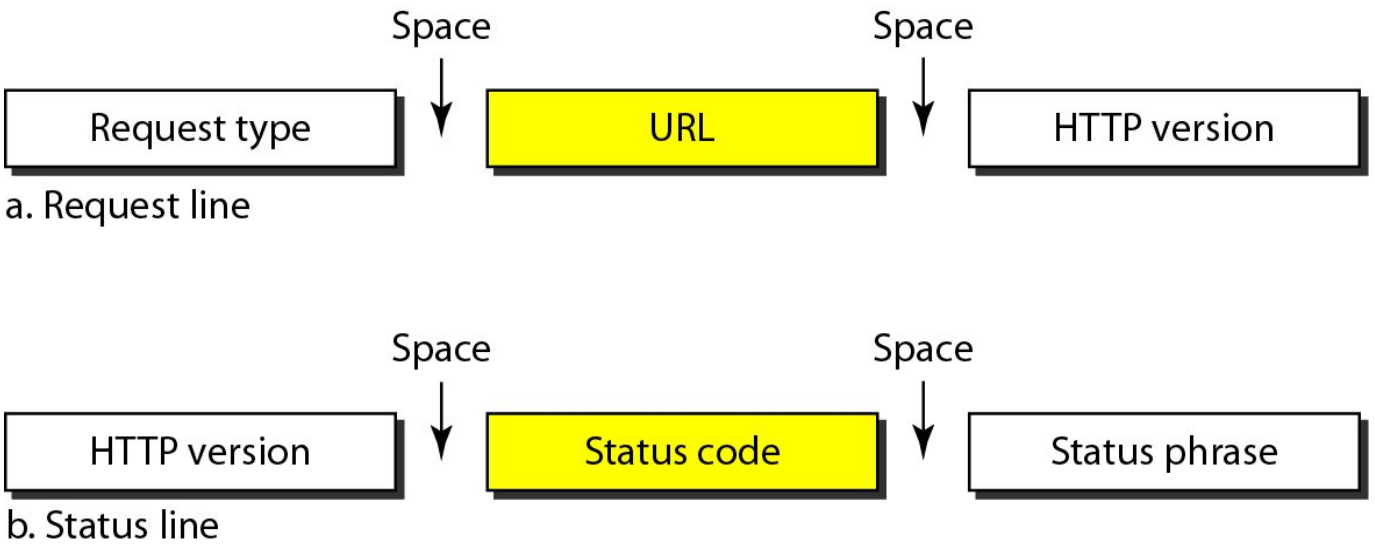
Format of the response message



Legend

sp: Space
cr: Carriage Return
lf: Line Feed

Figure 27.14 *Request and status lines*



In version 1.1 of HTTP, several methods are defined, as shown in Table below

Table 27.1 ***Methods***

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

Table 27.2 *Status codes*

Code	*Phrase*	*Description*
	Informational	
100	Continue	The initial part of the request has been received, and the client may continue with its request.
101	Switching	The server is complying with a client request to switch protocols defined in the upgrade header.
	Success	
200	OK	The request is successful.
201	Created	A new URL is created.
202	Accepted	The request is accepted, but it is not immediately acted upon.
204	No content	There is no content in the body.

Table 27.2 *Status codes (continued)*

Code	*Phrase*	*Description*
	Redirection	
301	Moved permanently	The requested URL is no longer used by the server.
302	Moved temporarily	The requested URL has moved temporarily.
304	Not modified	The document has not been modified.
	Client Error	
400	Bad request	There is a syntax error in the request.
401	Unauthorized	The request lacks proper authorization.
403	Forbidden	Service is denied.
404	Not found	The document is not found.
405	Method not allowed	The method is not supported in this URL.
406	Not acceptable	The format requested is not acceptable.
	Server Error	
500	Internal server error	There is an error, such as a crash, at the server site.
501	Not implemented	The action requested cannot be performed.
503	Service unavailable	The service is temporarily unavailable, but may be requested in the future.

Figure 27.15 *Header format*

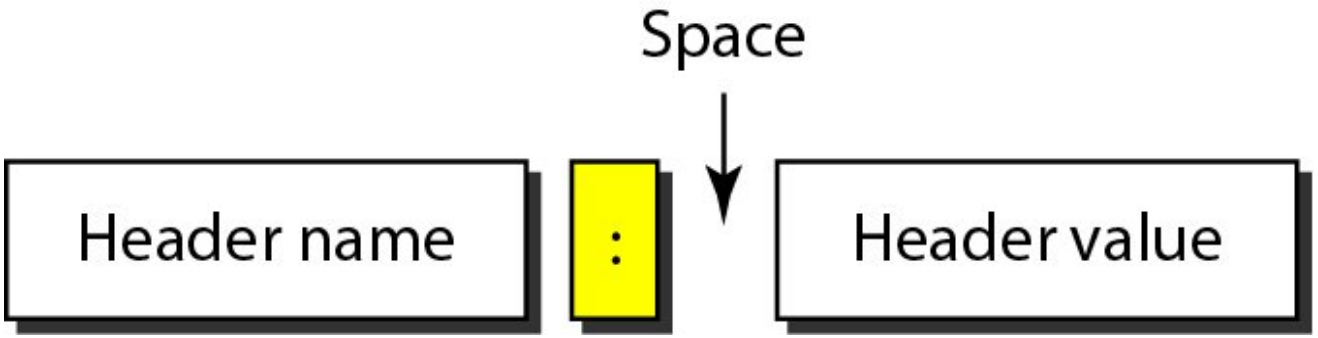


Table 27.3 *General headers*

Header	*Description*
Cache-control	Specifies information about caching
Connection	Shows whether the connection should be closed or not
Date	Shows the current date
MIME-version	Shows the MIME version used
Upgrade	Specifies the preferred communication protocol

Table 27.4 *Request headers*

Header	*Description*
Accept	Shows the medium 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
From	Shows the e-mail address of the user
Host	Shows the host and port number of the server
If-modified-since	Sends the document if newer than specified date
If-match	Sends the document only if it matches given tag
If-non-match	Sends the document only if it does not match given tag
If-range	Sends only the portion of the document that is missing
If-unmodified-since	Sends the document if not changed since specified date
Referrer	Specifies the URL of the linked document
User-agent	Identifies the client program

Table 27.5 *Response headers*

Header	*Description*
Accept-range	Shows if server accepts the range requested by client
Age	Shows the age of the document
Public	Shows the supported list of methods
Retry-after	Specifies the date after which the server is available
Server	Shows the server name and version number

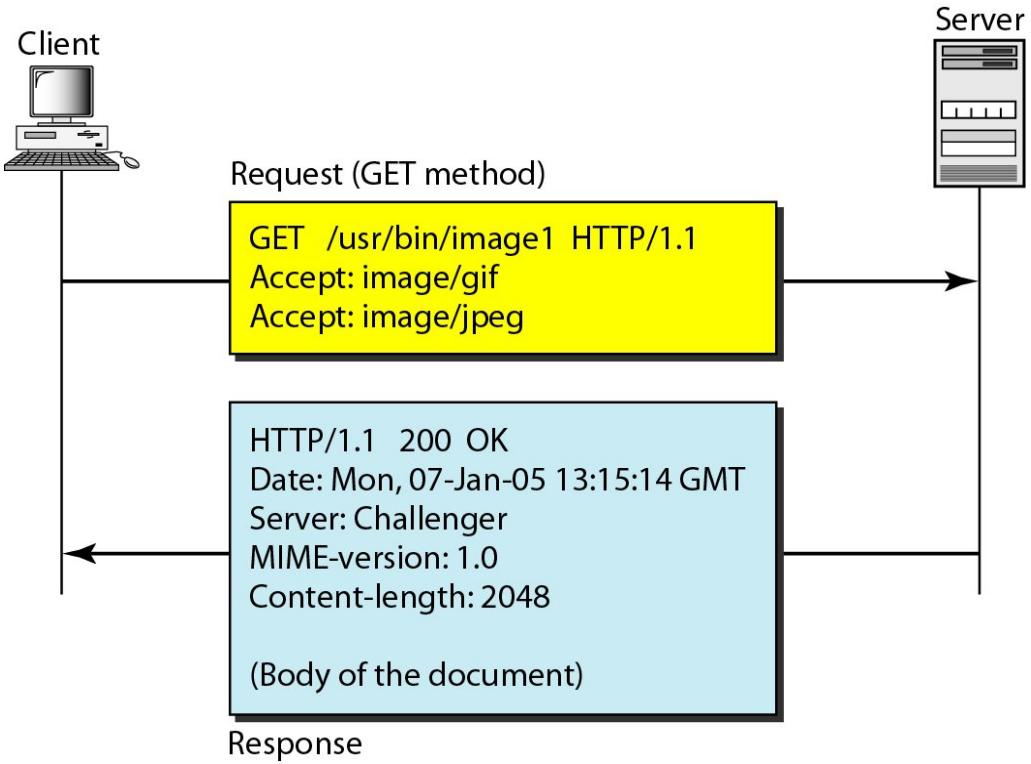
Table 27.6 *Entity headers*

Header	*Description*
Allow	Lists valid methods that can be used with a URL
Content-encoding	Specifies the encoding scheme
Content-language	Specifies the language
Content-length	Shows the length of the document
Content-range	Specifies the range of the document
Content-type	Specifies the medium type
Etag	Gives an entity tag
Expires	Gives the date and time when contents may change
Last-modified	Gives the date and time of the last change
Location	Specifies the location of the created or moved document

Example 27.1

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, MIME version, and length of the document. The body of the document follows the header (see Figure 27.16).

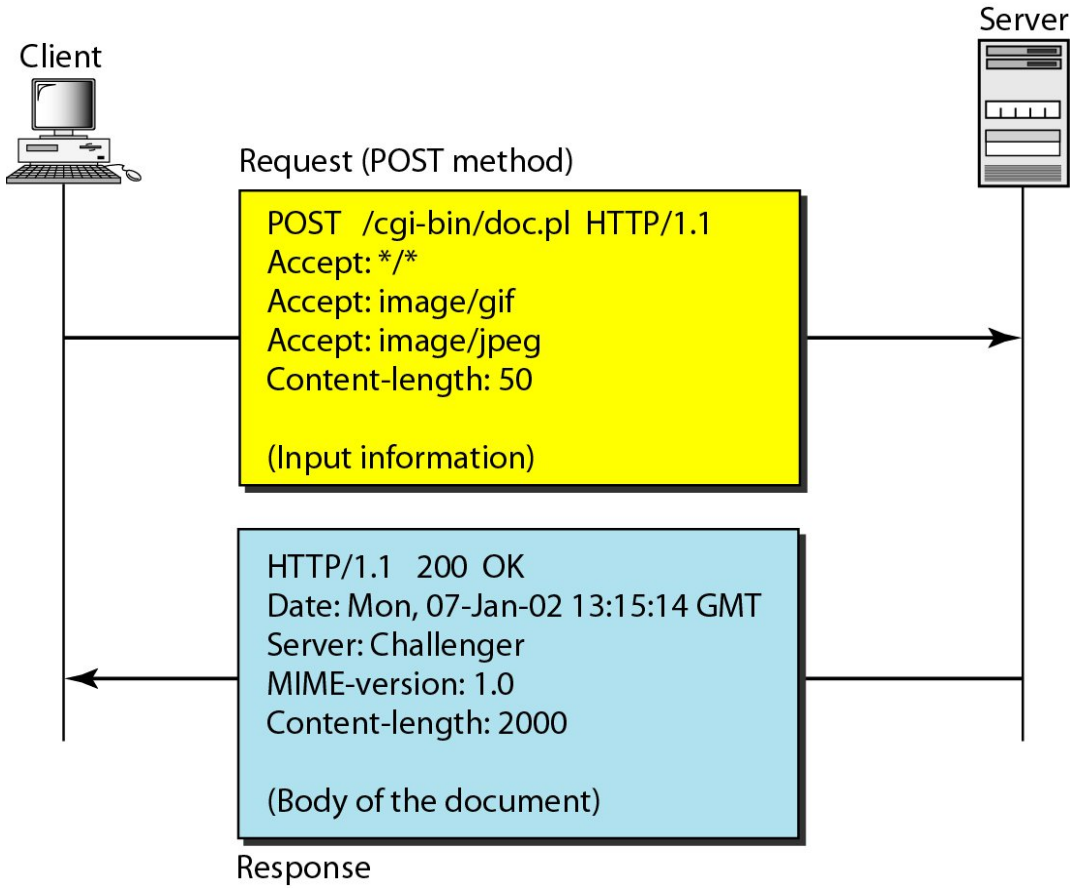
Figure 27.16 *Example 27.1*



Example 27.2

In this example, the client wants to send data to the server. We use the POST method. The request line shows the method (POST), URL, and HTTP version (1.1). There are four lines of headers. The request body contains the input information. 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 27.17).

Figure 27.17 *Example 27.2*



Example 27.3

HTTP uses ASCII characters. A client can directly connect to a server using TELNET, which logs into port 80 (see next slide). The next three lines show that the connection is successful. We then type three lines. The first shows the request line (GET method), the second is the header (defining the host), the third is a blank, terminating the request. The server response is seven lines starting with the status line. The blank line at the end terminates the server response. The file of 14,230 lines is received after the blank line (not shown here). The last line is the output by the client.

Example 27.3 (continued)

```
$ telnet www.mhhe.com 80
Trying 198.45.24.104 . . .
Connected to www.mhhe.com (198.45.24.104).
Escape character is '^]'.
GET /engcs/compsci/forouzan HTTP/1.1
From: forouzanbehrouz@fhda.edu

HTTP/1.1 200 OK
Date: Thu, 28 Oct 2004 16:27:46 GMT
Server: Apache/1.3.9 (Unix) ApacheJServ/1.1.2 PHP/4.1.2 PHP/3.0.18
MIME-version:1.0
Content-Type: text/html
```

Persistence

HTTP, prior to version 1.1, specified a non-persistent connection, while a persistent connection is the default in version 1.1.

Non-persistent Connection

In a **non-persistent 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.

Persistent Connection

- 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.

HTTP version 1.1 specifies a persistent connection by default.