Chapter 5

WWW and HTTP

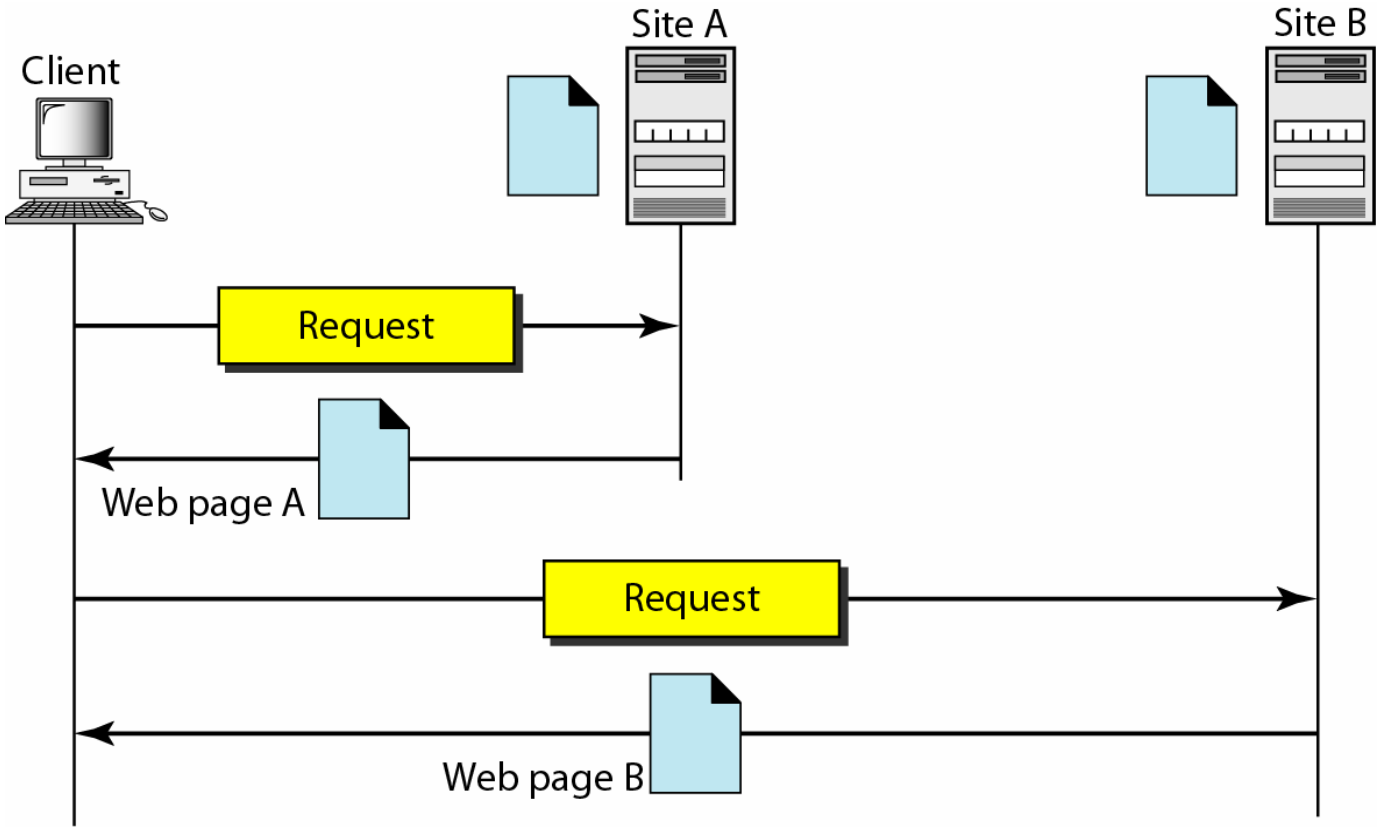
5-1 ARCHITECTURE

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.

Topics discussed in this section:

Client (Browser)
Server
Uniform Resource Locator
Cookies

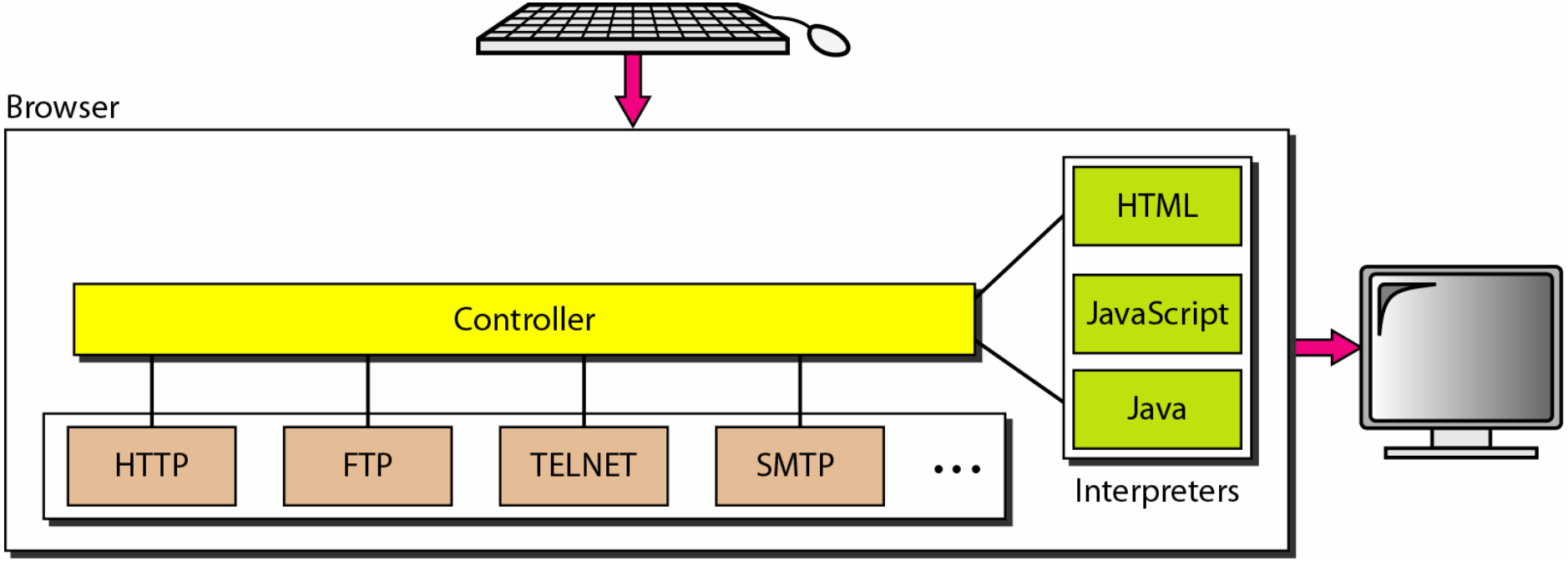
Figure 5.1 *Architecture of WWW*



Note

Client: Each browser usually consists of three parts: a controller, client protocol, and interpreters.

Figure 5.2 *Browser*



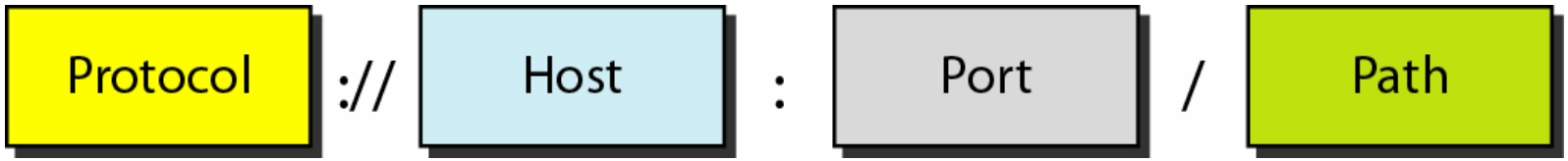
Note

Server: The Web page is stored at the server. Each time a client request arrives, document is sent to the client.

Note

Uniform Resource Locator (URL): The URL defines four things: protocol, host computer, port, and path

Figure 5.3 *URL*



Note

Creation and Storage of Cookies include three steps:

1. When a server receives a request from a client, it stores information about the client in a file or a string.

Note

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.

Note

Using Cookies: When a client sends a request to a server, the browser looks in the cookie directory to include in the request. When the server receives the request, it knows that this is an old client

5-2 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.

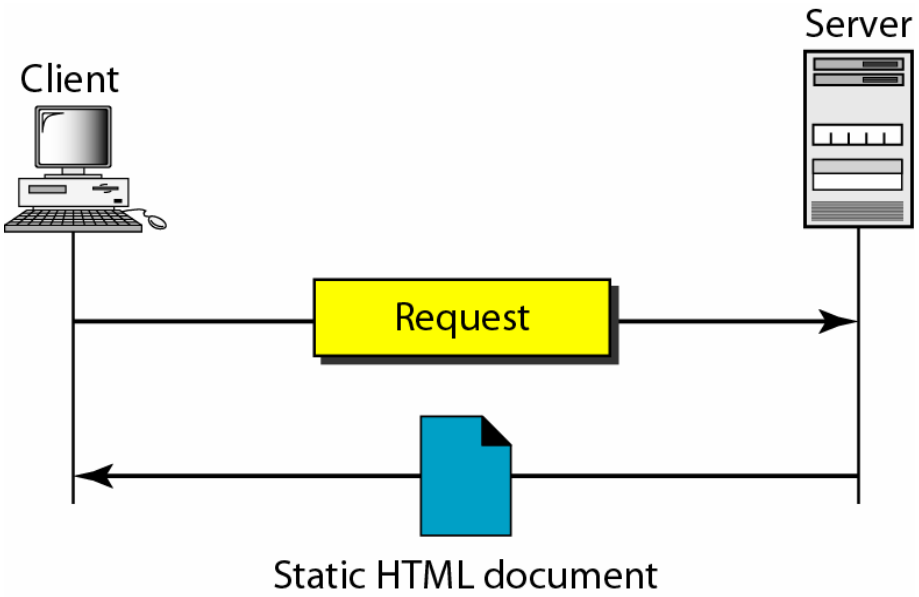
Topics discussed in this section:

Static Documents
Dynamic Documents
Active Documents

Note

Static Documents: Static documents are fixed-content documents. The contents of the file are determined when the file is created, not when it is used.

Figure 5.4 *Static document*



Note

Hypertext Markup Language (HTML) is a language for creating Web pages.

Figure 5.5 *Boldface tags*

Figure 5.6 *Effect of boldface tags*

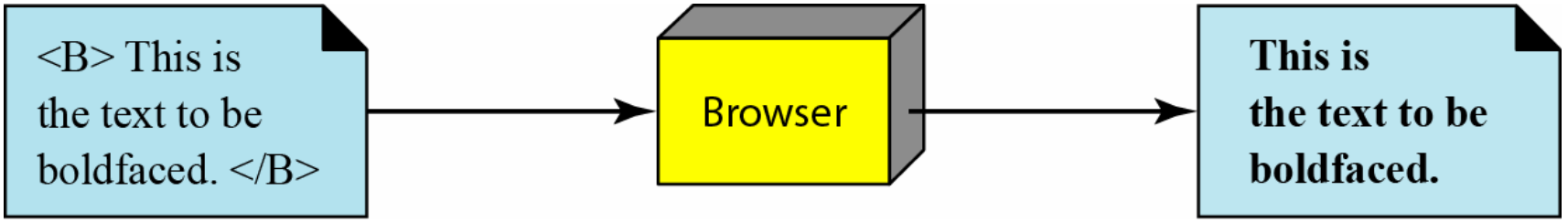
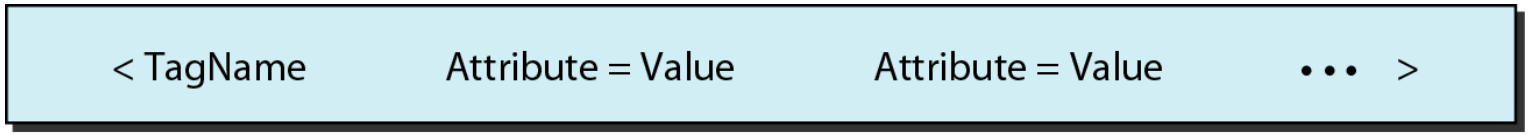


Figure 5.7 *Beginning and ending tags*



a. Beginning tag

b. Ending tag

Note

Dynamic Documents: A dynamic document is created by a Web server whenever a browser requests the document. The contents of a dynamic document can vary from one request to another.

Note

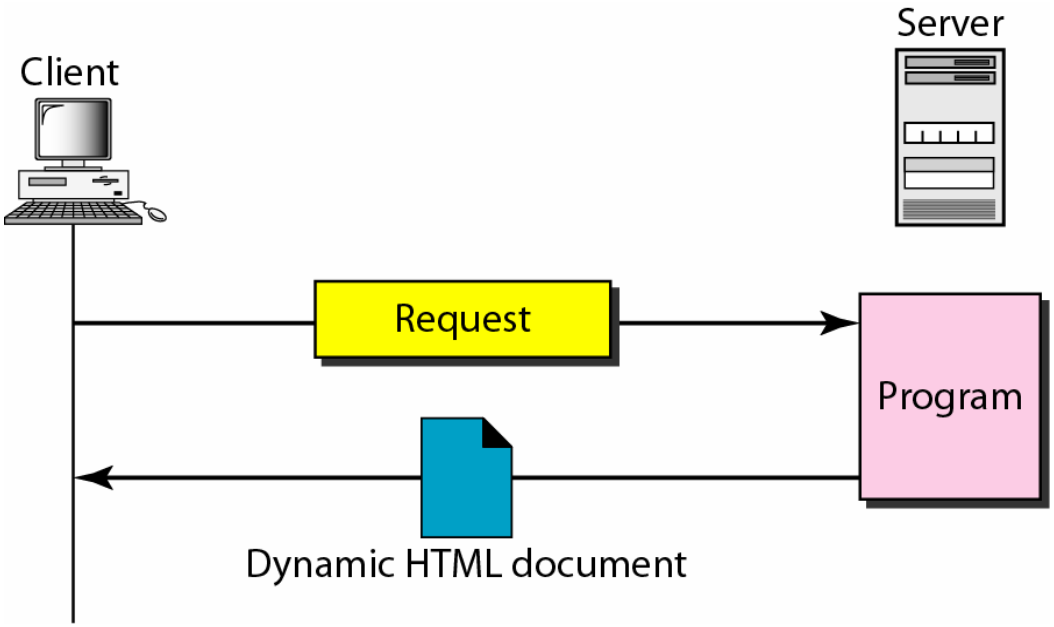
Common Gateway Interface (CGI): is a technology that creates and handles dynamic documents.

Note

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.

http://www.deanzalcgi-binlprog.pl?23

Figure 5.8 *Dynamic document using CGI*



Note

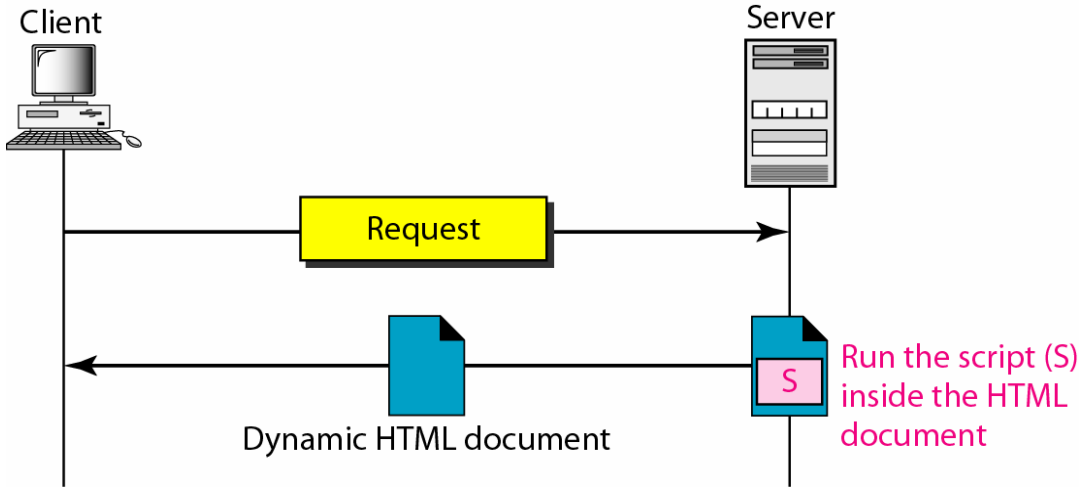
Scripting Technologies for Dynamic Documents

The problem with CGI technology is the inefficiency if part of the dynamic document is not changing from request to request.

Note

The 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 to provide the dynamic.

Figure 5.9 *Dynamic document using server-site script*



Note

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

Note

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

Figure 5.10 *Active document using Java applet*

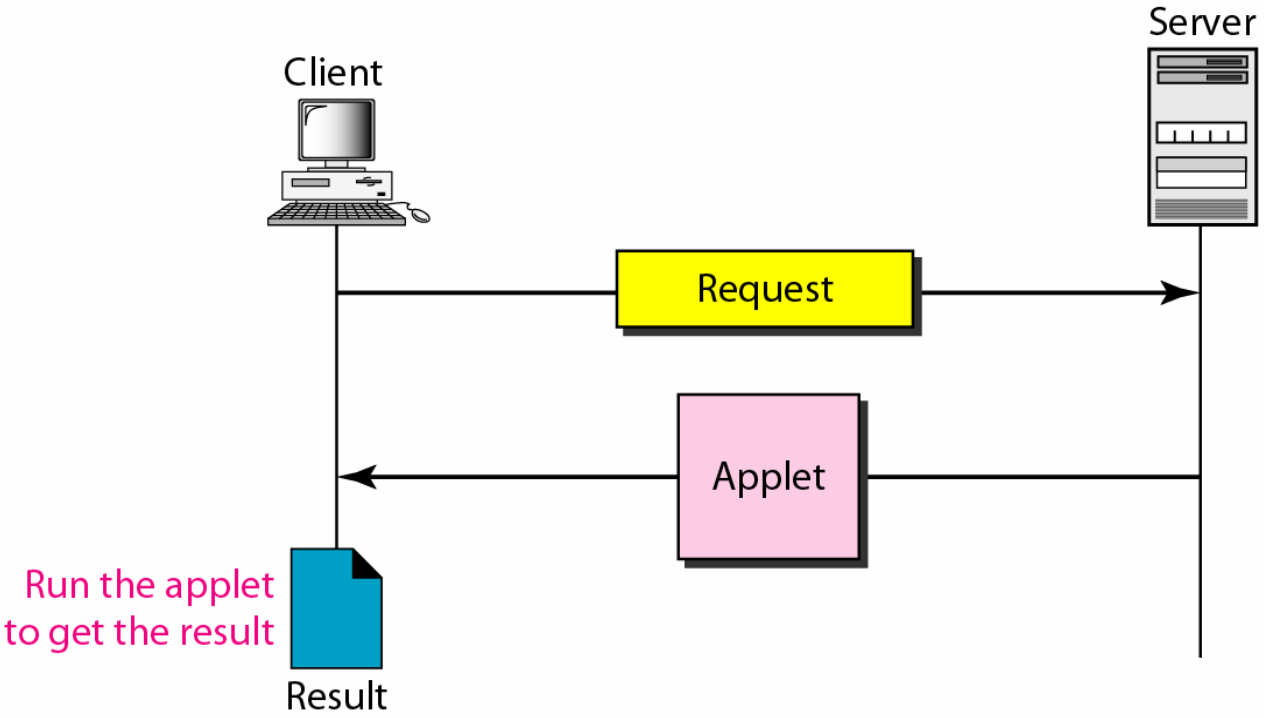
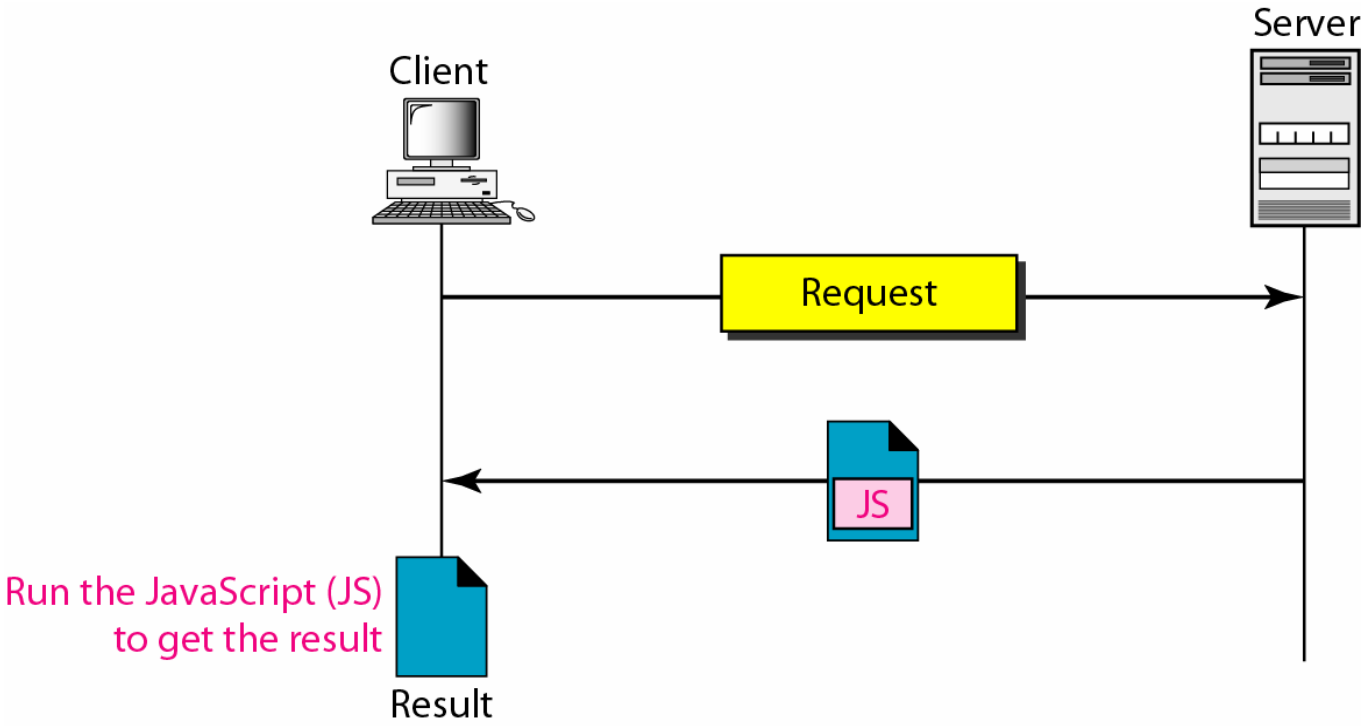


Figure 5.11 *Active document using client-site script*



Note

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

5-3 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.

Topics discussed in this section:

HTTP Transaction

Persistent Versus Nonpersistent Connection

Note

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

Note

A HTTP transaction includes a request and a response. The client initializes the transaction by sending a request message. The server replies by sending a response.

Figure 5.12 *HTTP transaction*

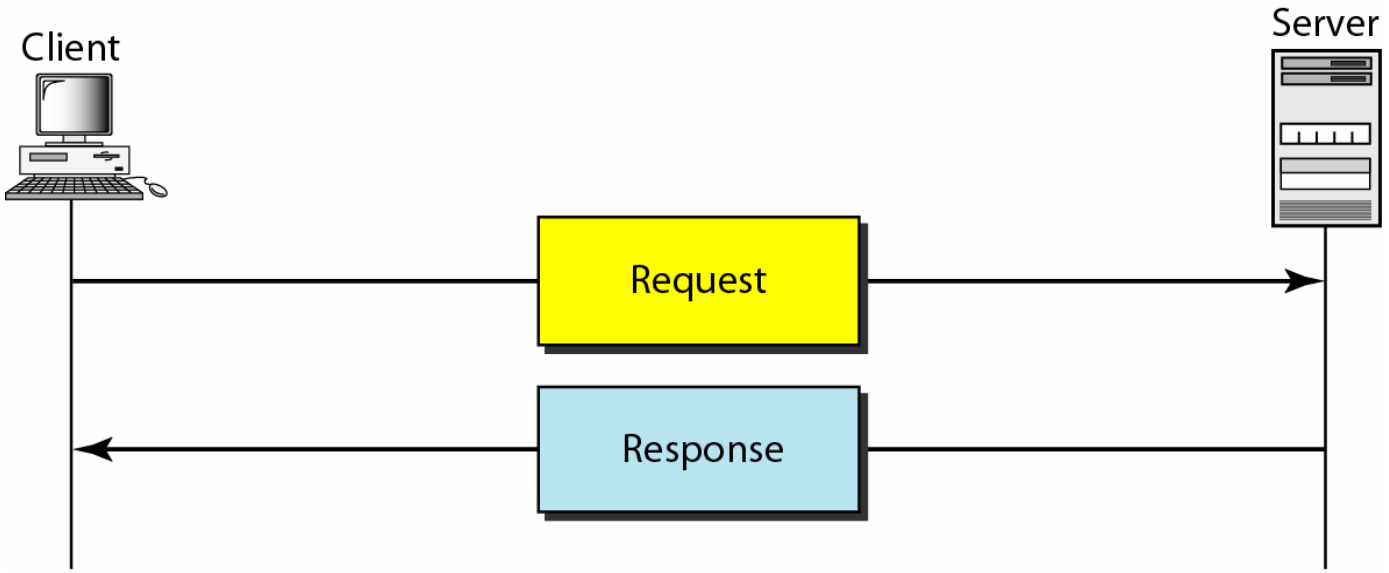


Figure 5.13 *Request and response messages*

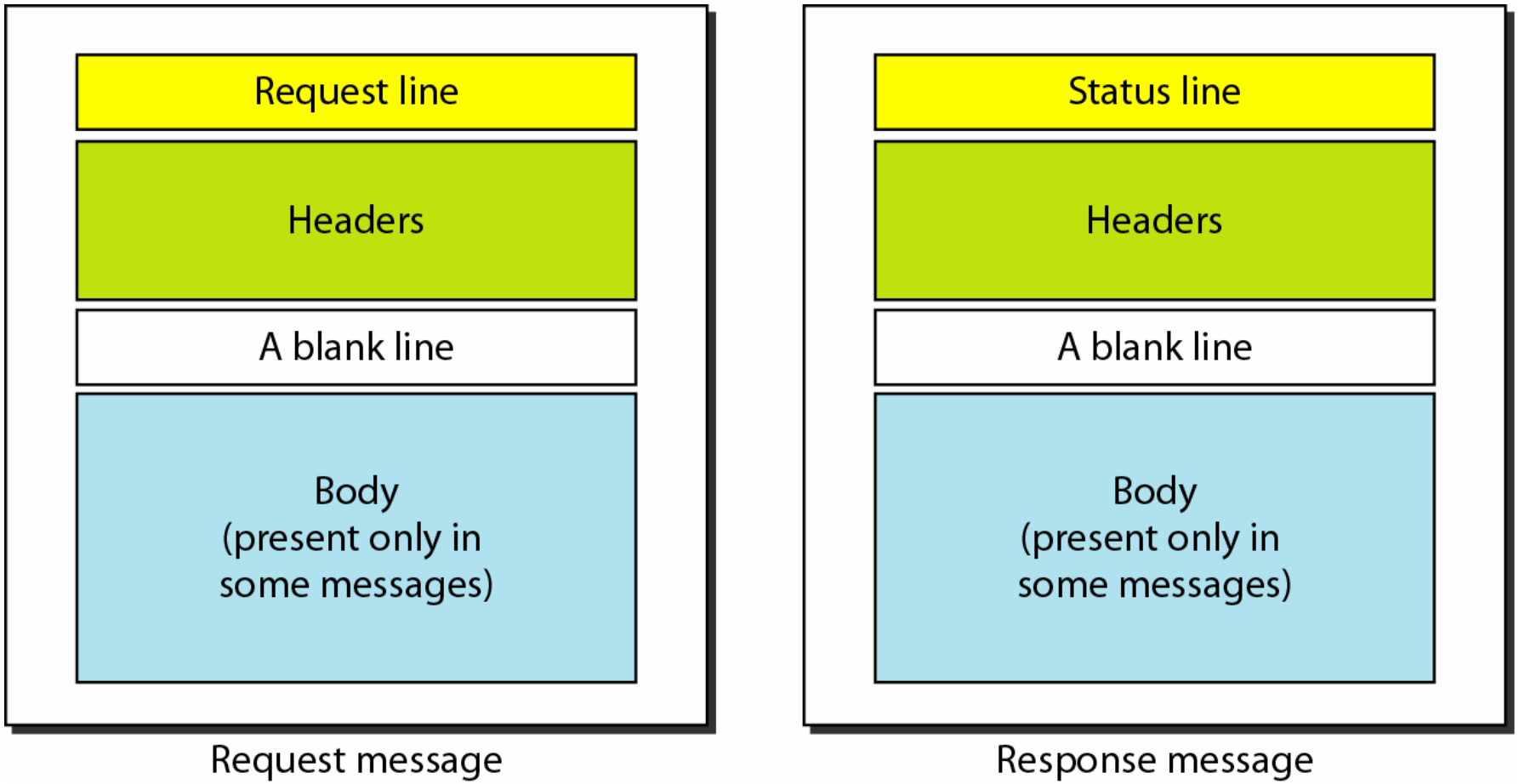
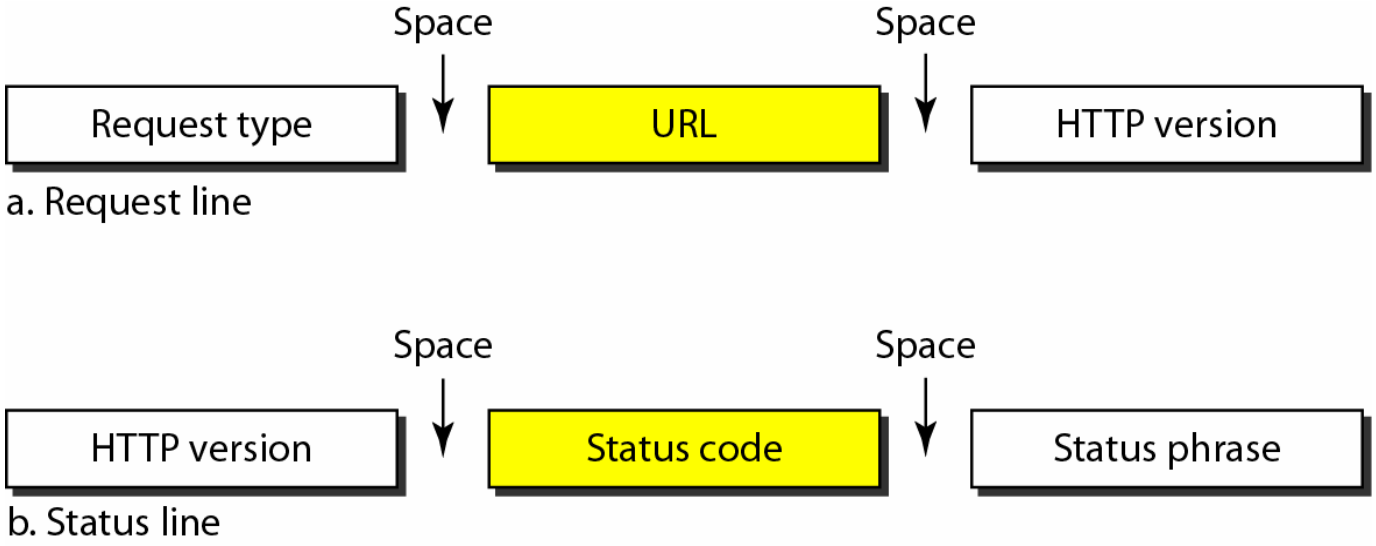


Figure 5.14 *Request and status lines*



Note

Request type. This field is used in the request message. The request type is categorized into methods

Table 5.1 *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

Note

Status code. This field is used in the response message. It consists of three digits.

Note

Status phrase. This field is used in the response message. It explains the status code in text form.

Table 5.2 *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.

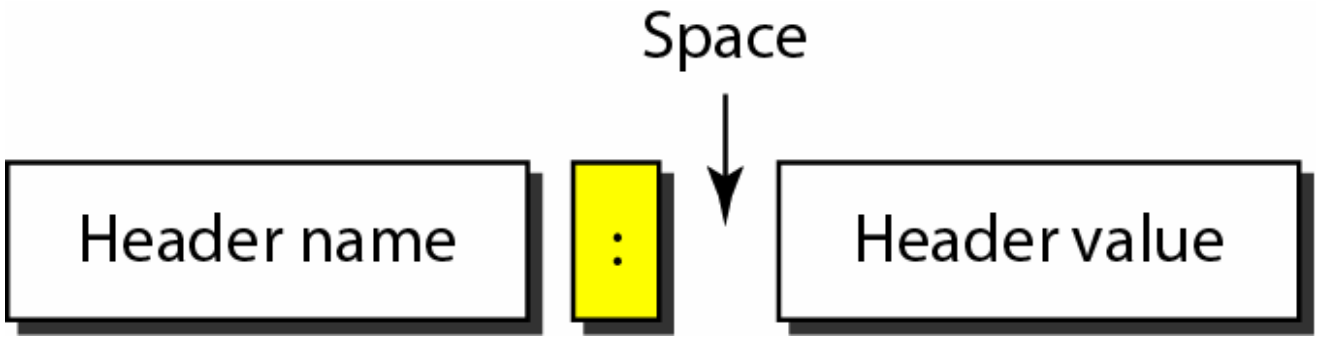
Table 5.2 *Status codes (continued)*

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.

Note

The header exchanges additional information between the client and the server. The header can consist of one or more header lines.

Figure 5.15 *Header format*



Note

A header line belongs to one of four categories: general header, request header, response header, and entity header.

Table 5.3 *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 5.4 *Request headers*

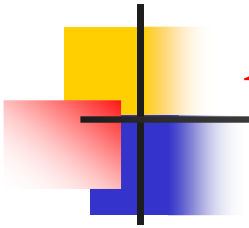
Header	*Description*
Accept	Shows the medium format the client can accept
Accept-charset	Shows the character set the client can handle
Accept-encoding	Shows the encoding scheme the client can handle
Accept-language	Shows the language the client can accept
Authorization	Shows what permissions the client 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 5.5 ***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 5.6 *Entity headers*

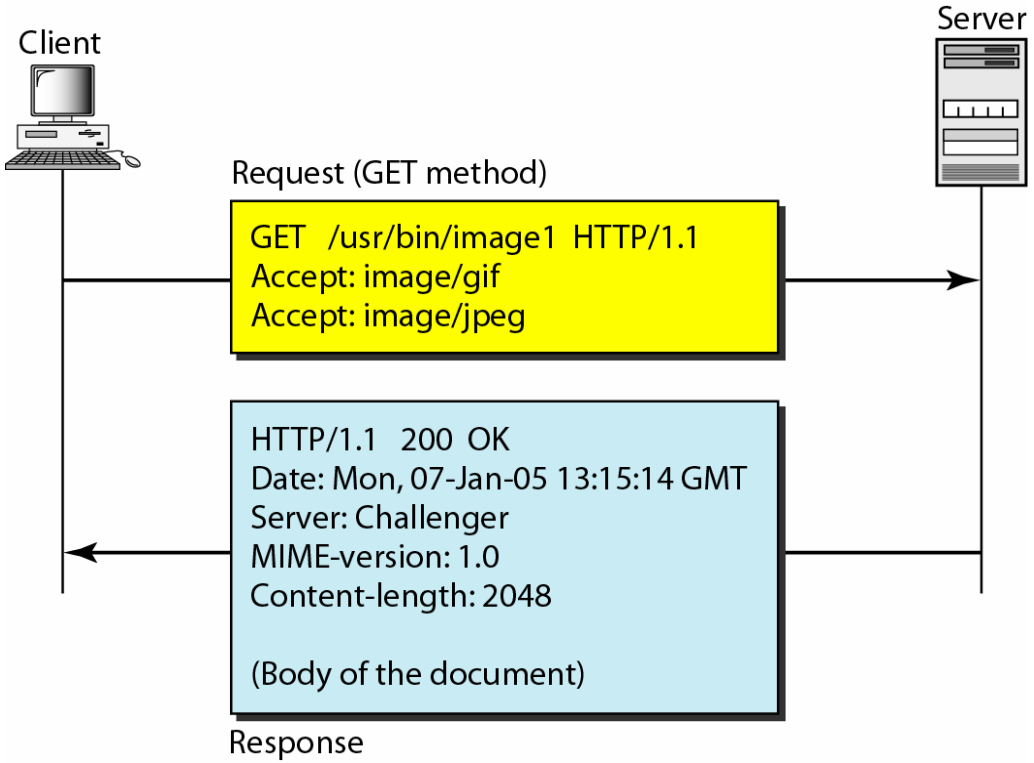
Header	*Description*
Allow	Lists valid methods that can be used with a URL
Content-encoding	Specifies the encoding scheme
Content-language	Specifies the language
Content-length	Shows the length of the document
Content-range	Specifies the range of the document
Content-type	Specifies the medium type
Etag	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

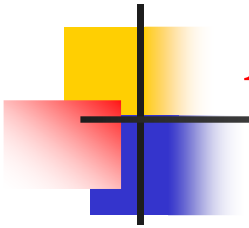


Example 5.1

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 5.16).

Figure 5.16 *Example 5.1*

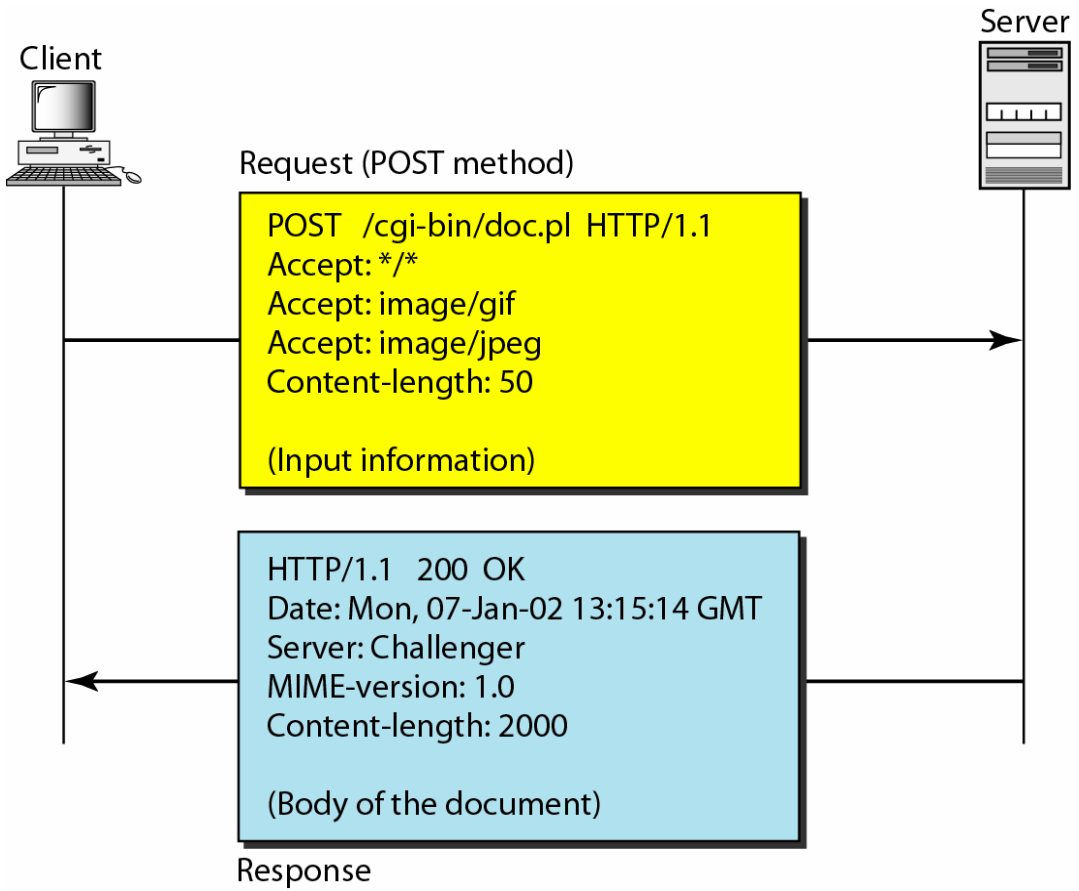


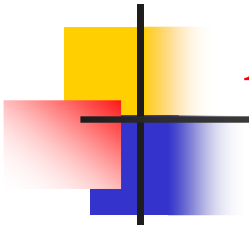


Example 5.2

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 5.17).

Figure 5.17 *Example 5.2*





Example 5.3

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 5.3 (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
```

Note

Persistent Versus Nonpersistent Connection. A HTTP connection can be persistent or non-persistent.

Note

HTTP version 1.1 specifies a persistent connection by default.

Note

Nonpersistent Connection: In a nonpersistent connection, one TCP connection is made for each request/response.

Note

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.

Note

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.

Note

Proxy Server: HTTP supports proxy servers. A proxy server is a computer that keeps copies of responses to recent requests.