3. WWW and HTTP

The World Wide Web (WWW) is a repository of information linked together from points all over the world. The WWW has a unique combination of flexibility, portability, and user-friendly features that distinguish it from other services provided by the Internet. The WWW project was initiated by CERN (European Laboratory for Particle Physics) to create a system to handle distributed resources necessary for scientific research. In this chapter we first discuss issues related to the web. We then discuss a protocol, HTTP that is used to retrieve information from the Web.

3.1 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, as shown in fig. 3.1.

Fig. 3.1 Architecture of WWW

Each site holds one or more documents, referred to as web pages. Each web page can contain a link to other pages in the same site or at other sites. The pages can be retrieved and viewed by using browsers. As shown in fig. 3.1. The client needs to see some information that it knows belongs to site A. It sends a request through its browser, a program that is designed to fetch web documents. The request, among other information, includes the address of the site and the Web page, called the URL (Uniform
Resource Locator). The server at site A finds the document and sends it to the client. When the user views the document, the user finds some references to other documents, including a web page at site B. The reference has the URL for the new site. The user is also interested in seeing this document. The client sends another request to the new site, and the new page is retrieved.

3.1.1 Client (Browser)
A variety of vendors offer commercial browsers that interpret and display a web document, and all use nearly the same architecture. Each browser usually consists of three parts: a controller, client protocol, and interpreters. The controller receives input from the keyboard or the mouse and uses the client programs to access the document. After the document has been accessed, the controller uses one of the interpreters to display the document on the screen. The client protocol can be one of the protocols described previously such as FTP. The interpreter can be HTML, Java, or JavaScript, depending on the type of document as shown in fig.3.2.

![Fig.3.2 Browser](image)

3.1.2 Server
The web page is stored at the server. Each time a client request arrives, the corresponding document is sent to the client. To improve efficiency, servers normally store requested files in a cache in memory; memory is faster to access than disk. A server can also become more efficient through multithreading or multiprocessing. In this case, a server can answer more than one request at a time.
3.1.3 Uniform Resource Locator

A client that wants to access a web page needs the address. To facilitate the access of documents distributed throughout the world, HTTP uses locators. The Uniform Resource Locator (URL) is a standard for specifying any kind of information on the Internet. The URL defines four things: protocol, host computer, port, and path as shown in fig.3.3.

```
Protocol :// Host : Port / Path
```

Fig.3.3 URL

The protocol is the client/server program used to retrieve the document. Many different protocols can retrieve a document; among them are FTP or HTTP. The most common today is HTTP.

The host is the computer on which the information is located, although the name of the computer can be as alias. Web pages are usually stored in computers, and computers are given alias names that usually begin with the characters “WWW”. This is not mandatory, however, as the host can be any name given to the computer that hosts the Web page. 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. The path can contain slashes that separate the directories from the subdirectories and files.

3.1.4 Cookies

The World Wide Web was originally designed as a stateless entity. A client sends a request; a server responds. Their relationship is over. The original design of WWW is retrieving publicly available documents. Today the web has other functions; some are:
1. Some websites need to allow access to registered clients only.
2. Websites are being used as electronic stores that allow users to browse through the store, select wanted items, put them in an electronic cart, and pay at the end with a credit card.
3. Some websites are used as portals: the user selects the web pages that the user wants to see.
4. Some websites are just advertising.
For these purposes, the cookie mechanism was devised.
Creation and Storage of Cookies
The creation and storage 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 stored by the domain server 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 it is 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. The usage of the cookie for the four previously mentioned purposes:

1. 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.
2. An electronic store (e-commerce) can use a cookie for its client shoppers. When a client selects an item and inserts it into 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.
3. A web portal uses the cookie in a similar way. When a user selects the favorite pages, a cookie is made and sent. If the site is accessed again, the cookie is sent to the server to show what the client is looking for.
4. A cookie is also used by advertising agencies. An advertising agency can place banner ads on some main website that is often visited by users. The advertising agency supplies only a URL that gives the banner address instead of the banner itself. When a user visits the main website and clicks on the icon of an advertised corporation, a request is sent to the advertising agency. The advertising agency
sends the banner, a GIF file, for ex., but it also includes a cookie with the ID of the user. Any future use of the banners adds to the database that profiles the Web behavior of the user. The advertising agency has compiled the interests of the user and can sell this information to other parties. This use of cookies has made them very controversial.

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

3.2.1 Static Documents
Static documents are fixed-content documents that are created and stored in a server. The client can get only a copy of the document. In other words, the contents of the file are determined when the file is created, not when it is used. Of course, the contents in the server can be changed, but the user cannot change them. When a client accesses the document, a copy of the document is sent. The user can then use a browsing program to display the document. As shown in Fig. 3.4.

![Fig. 3.4 Static Document](image)

**HTML**
Hypertext Markup Language (HTML) is a language for creating Web pages. The term markup language comes from the book publishing industry. Before a book is typeset and
printed, a copy editor reads the manuscript and puts marks on it. These marks tell the
compositor how to format the text. For ex., if the copy editor wants part of a line to be
printed in boldface, the user draws a wavy line under that part. In the same way, data
for a web page are formatted for interpretation by a browser.
To make part of a text displayed in boldface with HTML, we put beginning and ending
boldface tags (marks) in the text, as shown in Fig.3.5.

![Fig.3.5 Boldface tags](image)

The two tags <B> and </B> are instructions for the browser. When the browser sees
these two marks, it knows that the text must be boldfaced (as in fig.3.6). A markup
language such as HTML allows us to embed formatting instructions in the file itself. The
instructions are included with the text. In this way, any browser can read the
instructions and format the text according to the specific workstation. One might ask
why we do not use the formatting capabilities of word processors to create and save
formatted text. The answer is that different word processors use different techniques or
procedures for formatting text. For ex., imagine that a user creates formatted text on a
Macintosh computer and stores it in a Web page. Another user who is on an IBM
computer would not be able to receive the Web page because the two computers use
different formatting procedures.

HTML lets us use only ASCII characters for both the main text and formatting
instructions. In this way, every computer can receive the whole document as an ASCII
document. The main text is the data, and the formatting instructions can be used by the
browser to format the data.
A Web page is made up of two parts: the head and the body. The head is the first part
of a Web page. The head contains the title of the page and other parameters that the
browser will use. The actual contents of a page are in the body, which includes the text
and the tags. Whereas the text is the actual information contained in a page, the tags
define the appearance of the document. Every HTML tag is a name followed by an
optional list of attributes, all enclosed between less-than and greater-than symbols (< and >).
An attribute, if present, is followed by an equal sign and the value of the attribute. Some tags can be used alone; others must be used in pairs. Those that are used in pairs are called beginning and ending tags. The beginning tag can have attributes and values and starts with the name of the tag. The ending tag cannot have attributes or values but must have a slash before the name of the tag.

The browser makes a decision about the structure of the text based on the tags, which are embedded into the text. Fig.3.7 shows the format of a tag.

One commonly used tag category is the text formatting tags such as <B> and </B>, which make the text bold; <I> and </I>, which make the text italic; and <U> and </U>, which underline the text.

Another interesting tag category is the image tag. Non textual information such as digitizing photos or graphic images is not a physical part of an HTML document. But we can use an image tag to point to the file of a photo or image. The image tag defines the address (URL) of the image to be retrieved. It also specifies how the image can be inserted after retrieval. We can choose from several attributes. The most common are...
SRC (source), which defines the source (address), and ALIGN, which defines the alignment of the image. The SRC attribute is required. Most browsers accept images in the GIF or JPEG formats. For ex., the following tag can retrieve an image stored as image 1.gif in the directory /bin/images.

```
<IMG SRC="/bin/images/image1.gif" ALIGN=MIDDLE>
```

A third interesting category is the hyperlink tag, which is needed to link documents together. Any item (word, phrase, paragraph, or image) can refer to another document through a mechanism called an anchor. The anchor is defined by <A...> and </A> tags, and the anchored item uses the URL to refer to another document.

When the document is displayed, the anchored item is underlined, blinking, or boldfaced. The user can click on the anchored item to go to another document, which may or may not be stored on the same server as the original document. The reference phrase is embedded between the beginning and ending tags. The beginning tag can have several attributes, but the one required is HREF (hyperlink reference), which defines the address (URL) of the linked document. For ex., the link to the author of a book can be:

```
<A HREF="http://www.deanza.edu/forouzan">Author</A>
```

What appears in the text is the word Author, on which the user can click to go to the author’s Web page.

### 3.2.2 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. Because a fresh document is created for each request, the contents of a dynamic document can vary from one request to another. A very simple example of a dynamic document is the retrieval of the time and date from a server. Time and date are kinds of information that are dynamic in that they change from moment to moment. The client can ask the server to run a program such as the date program in UNIX and send the result of the program to the client.
Common Gateway Interface (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 is not a new language; instead, it allows programmers to use any of several languages such as C, C++, Bourne Shell, C Shell, or Perl. The only thing that CGI defines is a set of rules and terms that the programmer must follow. The term common in CGI indicates that the standard defines a set of rules that is common to any language or platform. The term gateway here means that a CGI program can be used to access other resources such as database, graphical 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. A CGI program in its simplest form is code written in one of the languages supporting CGI. Any programmer who can encode a sequence of thoughts in a program and knows the syntax of one of the above mentioned languages can write a simple CGI program. Fig.3.8 illustrates the steps in creating a dynamic program using CGI technology.

Fig.3.8 Dynamic document using CGI

Input
In traditional programming, when a program is executed, parameters can be passed to the program. Parameter passing allows the programmer to write a generic program that can be used in different situations. For example, a generic copy program can be written to copy any file to another. A user can use the program to copy a file named x to another file named y by passing x as y parameters.
The input from a browser to a server is sent by using a form. If the information in a form is small (such as a word), it can be appended to the URL after a question mark. For example, the following URL is carrying form information (23, as a value):

```
http://www.deanza/cgi-bin/prog.pl?23
```

When the server receives the URL, it uses the part of the URL before the question mark to access the program to be run, and it interprets the part after the question mark (23) as the input sent by the client. It stores this string in a variable. When the CGI program is executed, it can access this value.

If the input from a browser is too long to fit in the query string, the browser can ask the server to send a form. The browser can then fill the form with the input data and send it to the server. The information in the form can be used as the input to the CGI program.

**Output**

The whole idea of CGI is to execute a CGI program at the server site and send the output to the client (browser). The output is usually plain text or a text with HTML structures; however, the output can be a variety of other things. It can be graphics or binary data, a status code, instructions to the browser to cache the result, or instructions to the server to send an existing document instead of the actual output. To let the client know about the type of document sent, a CGI program creates headers. As a matter of fact, the output of the CGI program always consists of two parts: a header and a body. The header is separated by a blank line from the body. This means any CGI program creates first the header, then a blank line, and then the body. Although the header and the blank line are not shown on the browser screen, the header is used by the browser to interpret the body.

**Scripting Technologies for Dynamic Documents**

The problem with CGI technology is the inefficiency that results if part of the dynamic document that is to be created is fixed and not changing from request. For example, assume that we need to retrieve a list of spare parts, their availability, and prices for a specific car brand. Although the availability and prices vary from time to time, the name, description, and the picture of the parts are fixed. If we use CGI, the program must create an entire document each time a request is made. 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 varying availability and price section. Fig.3.9 shows the idea.
A few technologies have been involved in creating dynamic documents using scripts. Among the most common are Hypertext Preprocessor (PHP), which uses the Perl language; Java Server Page (JSP), which uses Java language for scripting; Active Server Page (ASP), a Microsoft product which uses Visual Basic language for scripting; and ColdFusion, which embeds SQL database queries in the HTML document.

### 3.2.3 Active Documents

For many applications, we need a program or a script to be run at the client site. These are called active documents. For example, suppose we want to run a program that creates animated graphics on the screen or a program that interacts with the user. The program definitely needs to be run at the client site where the animation or interaction takes pace. 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.

**Java Applets**

One way to create an active document is to use Java applets. Java is a combination of a high-level programming language, a run-time environment, and a browser to run it. It can also be a stand-alone program that doesn’t use a browser.

An applet is a program written in Java on the server. It is compiled and ready to be run. The document is in byte-code (binary) format. The client process (browser) creates an instance of this applet and runs it. A java applet can be run by the browser in two ways. In the first method, the browser can directly request the Java applet program in the URL and receive the applet in binary form. In the second method, the browser can retrieve
and run an HTML file that has embedded the address of the applet as a tag. Fig. 3.10 shows how Java applets are used in the first method; the second is similar but needs two transactions.

Fig. 3.10 Active document using Java applet

**JavaScript**

The idea of scripts in dynamic documents can also be used for active documents. If the active part of the document is small, it can be written in a scripting language; then it can be interpreted and run by the client at the same time. The script is in source code (text) and not in binary form. The scripting technology used in this case is usually JavaScript. JavaScript, which bears a small resemblance to Java, is a very high level scripting language developed for this purpose. Fig. 3.11 shows JavaScript is used to create an active document.

Fig. 3.11 Active document using client-site script
3.3 HTTP

Hyper Text Transfer Protocol is a file transfer protocol specifically designed to facilitate access to the WWW. This protocol transfers data in the form of plain text, hypertext, audio, video, and so on. It is called HTTP because it is used in an environment where there are rapid jumps from one document to another.

HTTP functions like a combination of FTP and SMTP. It is similar to FTP because it transfers files and uses the services of TCP. However, it is much simpler than FTP because it uses only one TCP connection (well-known port 80). There is no separate control connection: only data are transferred between the client and the server.

HTTP is like SMTP because the data transferred between the client and the server is similar to SMTP messages. In addition, the format of the messages is controlled by MIME-like headers. However, HTTP differs from SMTP in the way the messages are sent from the client to the server and from the server to the client. Unlike SMTP messages, the HTTP messages are not destined to be read by humans; they are read and interpreted by the HTTP server and HTTP client (browser). SMTP messages are stored and forwarded, but HTTP messages are delivered immediately.

The idea of HTTP is very simple. A client sends a request, which looks like mail, to the server. The server sends the response, which looks like a mail reply to the client. The request and response messages carry data in the form of a letter with a MIME-like format. The commands from the client to the server are embedded in a letter-like request message. The contents of the requested file or other information are embedded in a letter-like response message.

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

3.3.1 Transaction

Fig.3.12 illustrates the HTTP transaction between the client and the server. Although HTTP uses the services of TCP, HTTP itself is a stateless protocol. The client initializes the transaction by sending a request message. The server replies by sending a response. There are two general types of HTTP messages: request and response. Both message types almost the same format.
3.3.2 Request AND Response Messages
The first line in a request message is called a request line; the first line in a response message is called the status line. There are some common fields, as shown in Fig.3.13.

- **Request and Status Line:** defines the request type, resource (URL), and HTTP version as in Fig.3.14.
  1. **Request Type:** In version 1.1 HTTP, several request types are defined. The request type categorizes the request messages into several methods.
These methods are defined as several kinds of messages. The request method is the actual command or request that a client issues to the server. Below are the purposes of some methods as listed in table 3.1:

<table>
<thead>
<tr>
<th>Method</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>Requests a document from the server</td>
</tr>
<tr>
<td>HEAD</td>
<td>Requests information about a document but not the document itself</td>
</tr>
<tr>
<td>POST</td>
<td>Sends some information from the client to the server</td>
</tr>
<tr>
<td>PUT</td>
<td>Sends a document from the server to the client</td>
</tr>
<tr>
<td>TRACE</td>
<td>Echoes the incoming request</td>
</tr>
<tr>
<td>CONNECT</td>
<td>Reserved</td>
</tr>
<tr>
<td>OPTION</td>
<td>Inquires about available options</td>
</tr>
</tbody>
</table>

Table 3.1 Methods

2. **Uniform Resource Locator (URL)**: As described previously.
3. **Version** although the most current version of **HTTP** of **HTTP** is 1.1, **HTTP** version 1.0 and 0.9 are still in use.

- **Response Message**:
  A response message consists of a status line, a header, and sometimes a body.

**Status Line**: The status line defines the status of the response message. It consists of the **HTTP** version, a space, a status code, a space, and a status phrase.
- **HTTP version**: This field is the same as the corresponding field in the request line.
- **Status code**: The status code field is similar to those in the **FTP** and the **SMTP** protocols. It consists of three digits. Whereas the codes in the 100 range are only informational, the codes in the 200 range indicate a successful request. The codes in the 300 range redirect the client to another **URL**, and the codes in the 400
range indicate an error at the client site. Finally, the codes in the 500 range indicate an error at the server site. Other most common codes are listed in Table 3.2.

- **Status Phrase:** This field explains the code in text form. Table 3.2 also gives the status phrase.

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

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

Table 3.2 Status codes
• Header

The header exchanges additional information between the client and the server. For example, the client can request that the document be sent in special format, or the server can send extra information about the document. The header can consist of one or more header lines. Each header line has a header name, a colon, a space, and a header value (As in Fig.3.15). A header line belongs to one of four categories: general header, request header, response header, and entity header. A request message can contain only general, request, and entity header. A response message, on the other hand, can contain only general, response, and entity headers.

![Header Format Diagram](Image)

Fig.3.15 Header format

1) General header: The general header gives general information about the message and can be present in both a request and a response. Table 3.3 lists some general headers with their descriptions.

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache-control</td>
<td>Specifies information about caching</td>
</tr>
<tr>
<td>Connection</td>
<td>Shows whether the connection should be closed or not</td>
</tr>
<tr>
<td>Date</td>
<td>Shows the current date</td>
</tr>
<tr>
<td>MIME-version</td>
<td>Shows the MIME version used</td>
</tr>
<tr>
<td>Upgrade</td>
<td>Specifies the preferred communication protocol</td>
</tr>
</tbody>
</table>

Table 3.3 General Headers

2) Request header: The request header can be present only in a request message. It specifies the client’s configuration and the client’s preferred document format. As in Table 3.4 which shows a list of some request headers and their descriptions.
3) **Response header**: The response header can be present only in a response message. It specifies the server's configuration and special information about the request. See Table 3.5 for a list of some response headers with their descriptions.

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept</td>
<td>Shows the medium format the client can accept</td>
</tr>
<tr>
<td>Accept-charset</td>
<td>Shows the character set the client can handle</td>
</tr>
<tr>
<td>Accept-encoding</td>
<td>Shows the encoding scheme the client can handle</td>
</tr>
<tr>
<td>Accept-language</td>
<td>Shows the language the client can accept</td>
</tr>
<tr>
<td>Authorization</td>
<td>Shows what permissions the client has</td>
</tr>
<tr>
<td>From</td>
<td>Shows the e-mail address of the user</td>
</tr>
<tr>
<td>Host</td>
<td>Shows the host and port number of the server</td>
</tr>
<tr>
<td>If-modified-since</td>
<td>Sends the document if newer than specified date</td>
</tr>
<tr>
<td>If-match</td>
<td>Sends the document only if it matches given tag</td>
</tr>
<tr>
<td>If-non-match</td>
<td>Sends the document only if it does not match given tag</td>
</tr>
<tr>
<td>If-range</td>
<td>Sends only the portion of the document that is missing</td>
</tr>
<tr>
<td>If-unmodified-since</td>
<td>Sends the document if not changed since specified date</td>
</tr>
<tr>
<td>Referrer</td>
<td>Specifies the URL of the linked document</td>
</tr>
<tr>
<td>User-agent</td>
<td>Identifies the client program</td>
</tr>
</tbody>
</table>

Table 3.5 Response header

4) **Entity header**: The entity header gives information about the body of the document. Although it is mostly present in response messages, some request messages, such as POST or PUT methods, that contain a body also use this type of header. See Table 3.6 for a list of some entity headers and their descriptions.
Table 3.4 Entity Headers

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

- **Body**
  The body can be present in a request or response message. Usually, it contains the document to be sent or received.

**Example 3.1**
This example retrieves a document. We use the **GET** method to retrieve an image with the path `/usr/bin/imagel`. 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 documents. The body of the document follows the header (see Figure 3.16).
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 Fig.3.17).
**Example 3.3**

HTTP uses ASCII characters. A client can directly connect to a server using TELNET, which logs into port 80. 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), and 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.

```
$ telnet www.mhhe.com 80
Trying 198.45.24.104
Connected to www.mhhe.com (198.45.24.104).
Escape character is '9'.
GET lengcslcompsci/forouzan HTTP/1.1
From: forouzanbehrouz @ thdaedu
HTTP/1.1 200 OK
Date: Thu, 28 Oct 2004 16:27:46 GMT
Server: Apache/1.39 (Unix) ApacheJServ/112 PHP/4.1,2 PHP/3.0.18
MIMEversion: 1.0
Content-Type: text/html
Last-modified: Friday, lSaOCtsO4 02:11:31 GMT
Content-length: 14230
Connection closed by foreign host.
```
3.4 Persistent Versus Nonpersistent Connection

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

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

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

In this strategy, for \( N \) different pictures in different files, the connection must be opened and closed \( N \) times. The nonpersistent strategy imposes high overhead on the server because the server needs \( N \) different buffers and requires a slow start procedure each time a connection is opened.

- **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. 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.

3.5 Proxy Server

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