UNIT-VI

Application layer (WWW and HTTP): ARCHITECTURE : Client (Browser) ,Server ,Uniform Resource Locator HTTP: HTTP Transaction, HTTP Operational Model and Client/Server Communication, HTTP Generic Message Format, HTTP Request Message Format, HTTP Response Message Format. **The wireless web :** WAP—The Wireless Application Protocol

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.

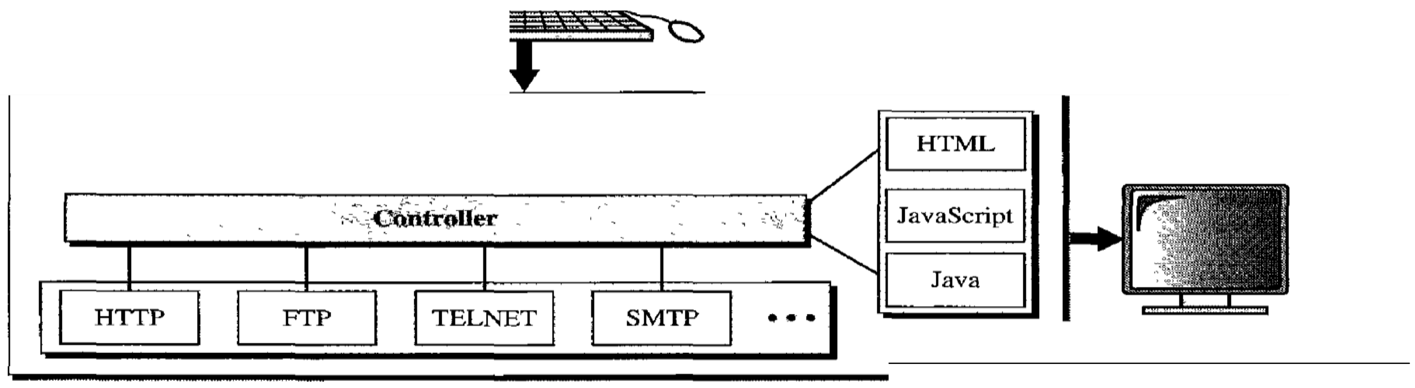
ARCHITECTURE

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 the figure.

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. Let us go through the scenario shown in Figure. 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. The server at site A finds the document and sends it to the client. When the user views the document, she 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.

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** or HTIP . The interpreter can be HTML, Java, or JavaScript, depending on the type of document.



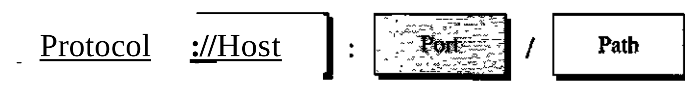
Server

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. Cache 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.

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.

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 an 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. Note that the path can itself contain slashes that, in the UNIX operating system, separate the directories from the subdirectories and files.

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, retrieving publicly available documents, exactly fits this purpose. Today the Web has other functions; some are listed here.

I. 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 he wants to see.

4. Some websites are just advertising.

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 sorted 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 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. Now let us see how a cookie is used 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 her 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.

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 FfP because it transfers files and uses the services of TCP. However, it is much simpler than FTP because it uses only one TCP connection. 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 look like SMTP messages. In addition, the format of the messages is controlled by MIME-like headers. Unlike SMTP, 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 commands from the client to the server are embedded in a request message. The contents of the requested file or other information are embedded in a response message. HTTP uses the services of TCP on well-known port 80.

HTTP Transaction

Figure illustrates the HTTP transaction between the client and 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.

Messages

The formats of the request and response messages are similar; both are shown in Fig• ure 27.13. A request message consists of a request line, a header, and sometimes a body. A response message consists of a status line, a header, and sometimes a body.

HTTP transaction

Request and response messages

Request type: This field is used in the request message. In version 1.1 of HTTP, several request types are defined. The request type is categorized into *methods* as defined in Table.

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

URL. We discussed the URL earlier in the chapter.
Version. The most current version of HTTP is 1.1.
Status code: This field is used in the response message. 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. We list the most common codes in Table.
Status phrase: This field is used in the response message. It explains the status code in text form. Table also gives the status phrase.

Table *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.
		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.

Header

The header exchanges additional information between the client and the server. For example, the client can request that the document be sent in a 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 (see Figure 27.15). We will show some header lines in the examples at the end of this chapter. 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 headers. A response message, on the other hand, can contain only general, response, and entity headers.

Header format

<u>General header</u>: the general header gives general information about the message and can be present in both a request and a response. Table 27.3 lists some general headers with their descriptions.

General headers

Header	*Description*
Cache-control	Specifies infonnation 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

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. See Table 27.4 for a list of some request headers and their descriptions.

Request headers

Header	*Description*
Accept	Shows the medium fonnat 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 pennissions 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

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 for a list of some response headers with their descriptions.

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

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 for a list of some entity headers and their descriptions.

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

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

Example 27.3

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), 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.

```
$ teinet www.mhhe.com 80
Trying 198.45.24.104 ...
Connected to www.mhhe.com (198.45.24.104).
Escape character is 11\]'.
GET /engcslcompscilforouzan HTTP/I.t
From: forouzanbehrouz@fbda.edu

HTTP/t.l 200 OK
Date: Thu, 28 Oct 2004 16:27:46 GMT
Server: Apache/l.3.9 (Unix) ApacheJServ/1.1.2 PHP/4.1.2 PHP/3.0.18
MIME-version: 1.0
Content-Type: text/html
        Last-modified: Friday, 15-0ct-04 02:11:31
        GMT Content-length: 14230
Connection closed by foreign host
```

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.

The Wireless Web

There is considerable interest in small portable devices capable of accessing the Web via a wireless link. We will focus on the first two wide area wireless Web systems to hit the market: WAP and i-mode.

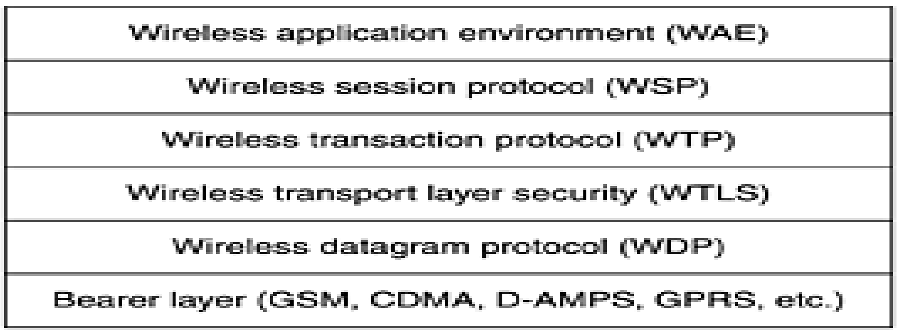
WAP-The Wireless Application Protocol

Once the Internet and mobile phones had become commonplace, it did not take long before somebody got the idea to combine them into a mobile phone with a built-in screen for wireless access to e-mail and the Web. The "somebody" in this case was a consortium initially led by Nokia, Ericsson, Motorola, and phone.com (formerly Unwired Planet) and now boasting hundreds of members. The system is called **WAP** (**Wireless Application Protocol**).

A WAP device may be an enhanced mobile phone, PDA, or notebook computer without any voice capability. The specification allows all of them and more. The basic idea is to use the existing digital wireless infrastructure. Users can literally call up a WAP gateway over the wireless link and send Web page requests to it. The gateway then checks its cache for the page requested. If present, it sends it; if absent, it fetches it over the wired Internet. In essence, this means that WAP 1.0 is a circuit-switched system with a fairly high per-minute connect charge. To make a long story short, people did not like accessing the Internet on a tiny screen and paying by the minute, so WAP was something of a flop (although there were other problems as well). However, WAP and its competitor, i-mode (discussed below), appear to be converging on a similar technology, so WAP 2.0 may yet be a big success. Since WAP 1.0 was the first attempt at wireless Internet, it is worth describing it at least briefly.

WAP is essentially a protocol stack for accessing the Web, but optimized for low-bandwidth connections using wireless devices having a slow CPU, little memory, and a small screen. These requirements are obviously different from those of the standard desktop PC scenario, which leads to some protocol differences. The layers are shown in Fig.

The WAP protocol stack.

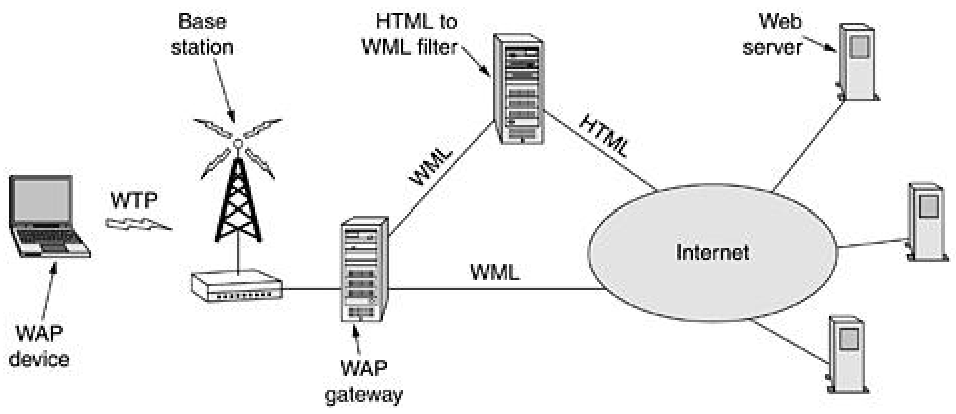


Wireless application environment (WAE)
Wireless session protocol (WSP)
Wireless transaction protocol (WTP)
Wireless transport layer security (WTLS)
Wireless datagram protocol (WDP)
Bearer layer (GSM, CDMA, D-AMPS, GPRS, etc.)

The lowest layer supports all the existing mobile phone systems, including GSM, D-AMPS, and CDMA. The WAP 1.0 data rate is 9600 bps. On top of this is the datagram protocol, **WDP** (**Wireless Datagram Protocol**), which is essentially UDP. Then comes a layer for security, obviously needed in a wireless system. WTLS is a subset of Netscape's SSL, which we will look at in Chap. 8. Above this is a transaction layer, which manages requests and responses, either reliably or unreliably. This layer replaces TCP, which is not used over the air link for efficiency reasons. Then comes a session layer, which is similar to HTTP/1.1 but with some restrictions and extensions for optimization purposes. At the top is a microbrowser (WAE).

Besides cost, the other aspect that no doubt hurt WAP's acceptance is the fact that it does not use HTML. Instead, the WAE layer uses a markup language called **WML** (**Wireless Markup Language**), which is an application of XML. As a consequence, in principle, a WAP device can only access those pages that have been converted to WML. However, since this greatly restricts the value of WAP, the architecture calls for an on-the-fly filter from HTML to WML to increase the set of pages available. This architecture is illustrated in Fig..

The WAP architecture.



In all fairness, WAP was probably a little ahead of its time. When WAP was first started, XML was hardly known outside W3C and so the press reported its launch as **WAP DOES NOT USE HTML.** A more accurate headline would have been: **WAP ALREADY USES THE NEW HTML STANDARD.** But once the damage was done, it was hard to repair and WAP 1.0 never caught on. We will revisit WAP after first looking at its major competitor.