File Transfer

Transferring files from one computer to another is one of the most common tasks expected from a networking or internetworking environment. As a matter of fact, the greatest volume of data exchange in the Internet today is due to file transfer. In this section, we discuss one popular protocol involved in transferring files: File Transfer Protocol *(FTP).*

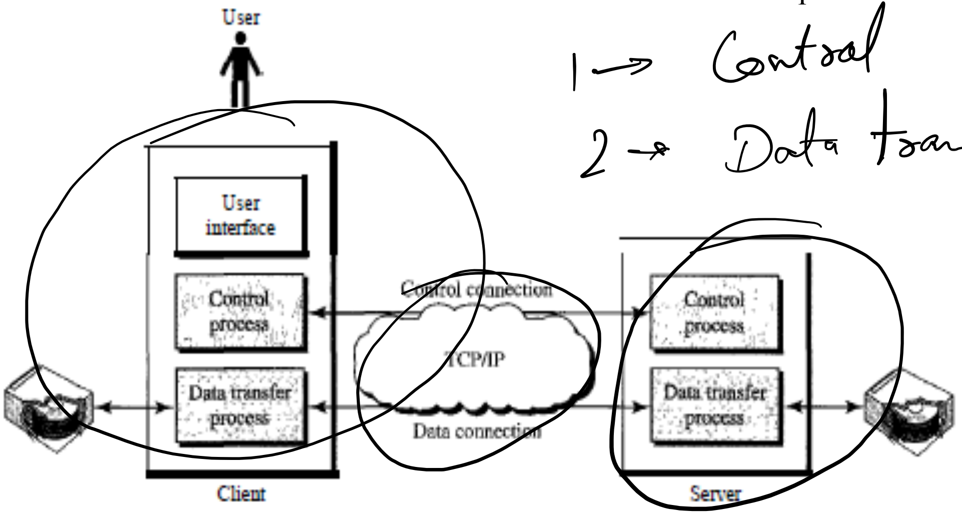
File Transfer Protocol (FTP)

File Transfer Protocol (FTP) is the standard mechanism provided by *TCP/IP* for copying a file from one host to another. Although transferring files from one system to another seems simple and straightforward, some problems must be dealt with first. For example, two systems may use different file name conventions. Two systems may have different ways to represent text and data. Two systems may have different directory structures. All these problems have been solved by FTP in a very simple and elegant approach.

FTP differs from other client/server applications in that it establishes two connections between the hosts. One connection is used for data transfer, the other for control information (commands and responses). Separation of commands and data transfer makes FTP more efficient. The control connection uses very simple rules of communication. We need to transfer only a line of command or a line of response at a time. The data connection, on the other hand, needs more complex rules due to the variety of data types transferred. However, the difference in complexity is at the FTP level, not TCP.

Note: For TCP, both connections are treated the same. FTP uses two well-known TCP ports: Port 21 is used for the control connection, and port 20 is used for the data connection.

Figure below shows the basic model of FTP. The client has three components: user interface, client control process, and the client data transfer process. The server has two components: the server control process and the server data transfer process. The control connection is made between the control processes. The data connection is made between the data transfer processes.



1 → Control

2 → Data transfer

Figure: FTP

The control connection remains connected during the entire interactive FTP session. The data connection is opened and then closed for each file transferred. It opens each time commands that involve transferring files are used, and it closes when the file is transferred. In other words, when a user starts an FTP session, the control connection opens. While the control connection is open, the data connection can be opened and closed multiple times if several files are transferred.

65-A, 97→a

Communication over Control Connection

FTP uses the 7-bit ASCII character set for the purpose of controlling the communication over the control connection. Communication is achieved through commands and responses. This simple method is adequate for the control connection because we send one command (or response) at a time. Each command or response is only one short line, so we need not worry about file format or file structure. Each line is terminated with a two-character (carriage return and line feed) end-of-line token.

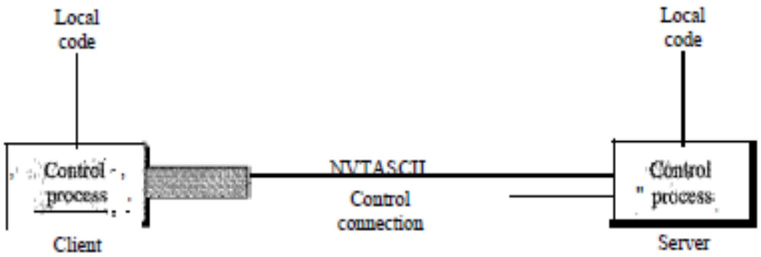


Figure: Using the control connection

Communication over Data Connection

The purpose of the data connection is different from that of the control connection. We want to transfer files through the data connection. File transfer occurs over the data connection under the control of the commands sent over the control connection. However, we should remember that file transfer in FTP means one of three things:

a) A file is to be copied from the server to the client. It is done under the supervision of the RETR command.

b) A file is to be copied from the client to the server. It is done under the supervision of the STOR command.

c) A list of directory or file names is to be sent from the server to the client. This is done under the supervision of the LIST command. Note that FTP treats a list of directory or file names as a file. It is sent over the data connection.

The client must define the type of file to be transferred, the structure of the data, and the transmission mode. Before sending the file through the data connection, we prepare for transmission through the control connection. The heterogeneity problem is resolved by defining three attributes of communication: file type, data structure, and transmission mode.

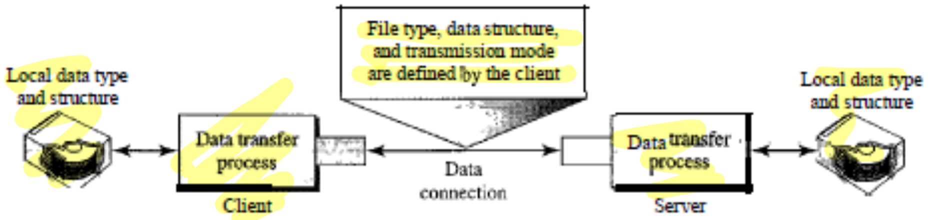

File type, data structure, and transmission mode are defined by the client
Local data type and structure
Data transfer process
Data connection
Data transfer process
Local data type and structure
Client
Server

Figure: Using the data connection

ASCII -> Text file
Image -> Binary File

File Type

FTP can transfer one of the following file types across the data connection: an ASCII file, EBCDIC file, or image file. The ASCII file is the default format for transferring text files. Each character is encoded using 7-bit ASCII. The sender transforms the file from its own representation into ASCII characters, and the receiver transforms the ASCII characters to its own representation. If one or both ends of the connection use EBCDIC encoding (the file format used by IBM), the file can be transferred using EBCDIC encoding. The image file is the default format for transferring binary files. The file is sent as continuous streams of bits without any interpretation or encoding. This is mostly used to transfer binary files such as compiled programs.

Data Structure

-> File
-> record -> text
-> page

FTP can transfer a file across the data connection by using one of the following interpretations about the structure of the data: file structure, record structure, and page structure. In the file structure format, the file is a continuous stream of bytes. In the record structure, the file is divided into records. This can be used only with text files. In the page structure, the file is divided into pages, with each page having a page number and a page header. The pages can be stored and accessed randomly or sequentially.

Transmission Mode

-> Stream mode
-> Block mode
-> compressed

FTP can transfer a file across the data connection by using one of the following three transmission modes: stream mode, block mode, and compressed mode. The stream mode is the default mode. Data are delivered from FTP to TCP as a continuous stream of bytes. TCP is responsible for chopping data into segments of appropriate size. If the data are simply a stream of bytes (file structure), no end-of-file is needed. End-of-file in this case is the closing of the data connection by the sender. If the data are divided into records (record structure), each record will have a I-byte end-of-record (EOR) character and the end of the file will have a I-byte end-of-file (EOF) character. In block mode, data can be delivered from FTP to TCP in blocks. In this case, each block is preceded by a 3-byte header. The first byte is called the *block descriptor;* the next 2 bytes define the size of the block in bytes. In the compressed mode, if the file is big, the data can be compressed. The compression method normally used is run-length encoding. In this method, consecutive appearances of a data unit are replaced by one occurrence and the number of repetitions. In a text file, this is usually spaces (blanks). In a binary file, null characters are usually compressed.

Anonymous FTP

To use FTP, a user needs an account (user name) and a password on the remote server. Some sites have a set of files available for public access, to enable anonymous FTP. To access these files, a user does not need to have an account or password. Instead, the user can use *anonymous* as the user name and *guest* as the password. User access to the system is very limited. Some sites allow anonymous users only a subset of commands. For example, most sites allow the user to copy some files, but do not allow navigation through the directories.

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. 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

→ Client initializes the transaction by sending a request message.

Figure 27.12 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 figure below. 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.

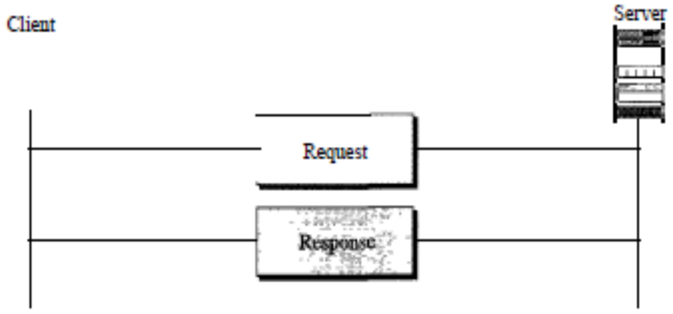
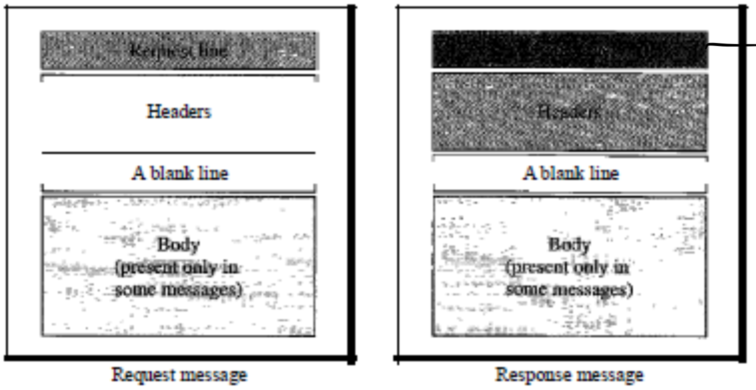


Figure: HTTP transaction



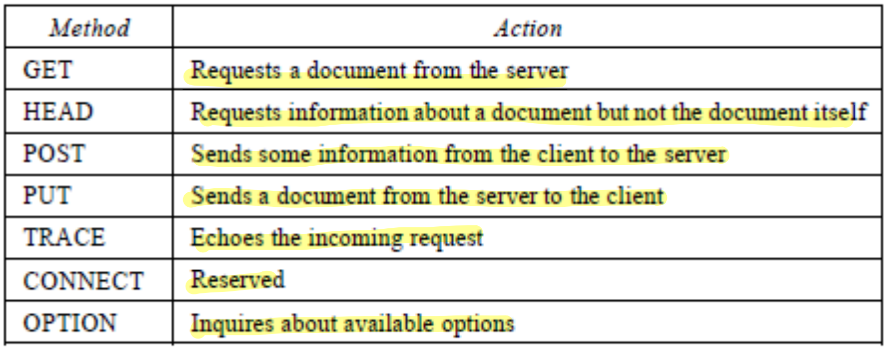
→ status line

Figure: Request and response messages

Request and Status Lines

The first line in a request message is called a request line; the first line in the response message is called the status line.

a) 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 below.



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

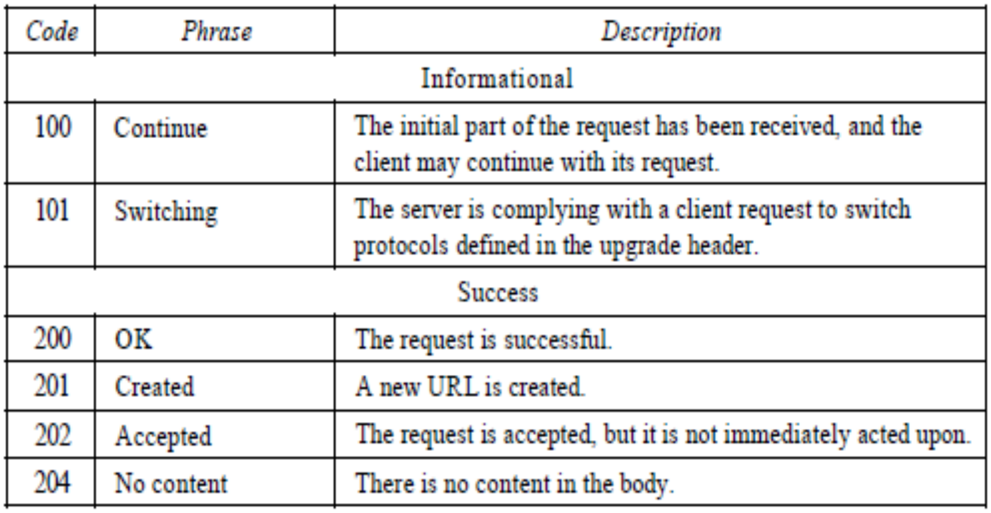
b) URL: URL is an acronym for Uniform Resource Locator and is a reference (an address) to a resource on the Internet. A URL has two main components:
   - Protocol identifier: For the URL http://example.com, the protocol identifier is *http*.
   - Resource name: For the URL http://example.com, the resource name is *example.com*.

c) Version: The most current version of HTTP is 1.1.

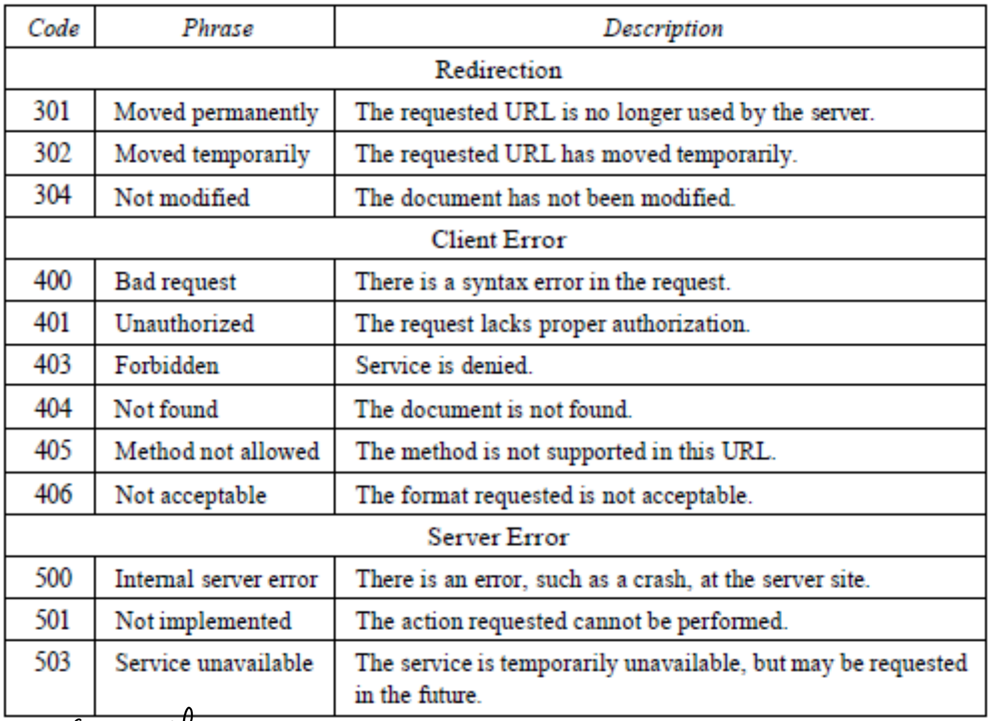
d) Status code: This field is used in the response message. The status code field is similar to those in the FTP protocol. It consists of three digits. Whereas the codes in the 100 range are only informational, the codes in the 200 range indicate a successful request. The codes in the 300 range redirect 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

e) most common codes in table below.

f) Status phrase: This field is used in the response message. It explains the status code in text form. Table below also gives the status phrase.



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.



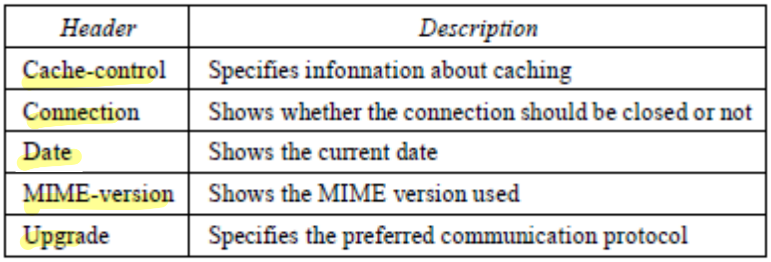
Code	Phrase	Description
		Redirection
301	Moved permanently	The requested URL is no longer used by the server.
302	Moved temporarily	The requested URL has moved temporarily.
304	Not modified	The document has not been modified.
		Client Error
400	Bad request	There is a syntax error in the request.
401	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.

Header

General
request
response

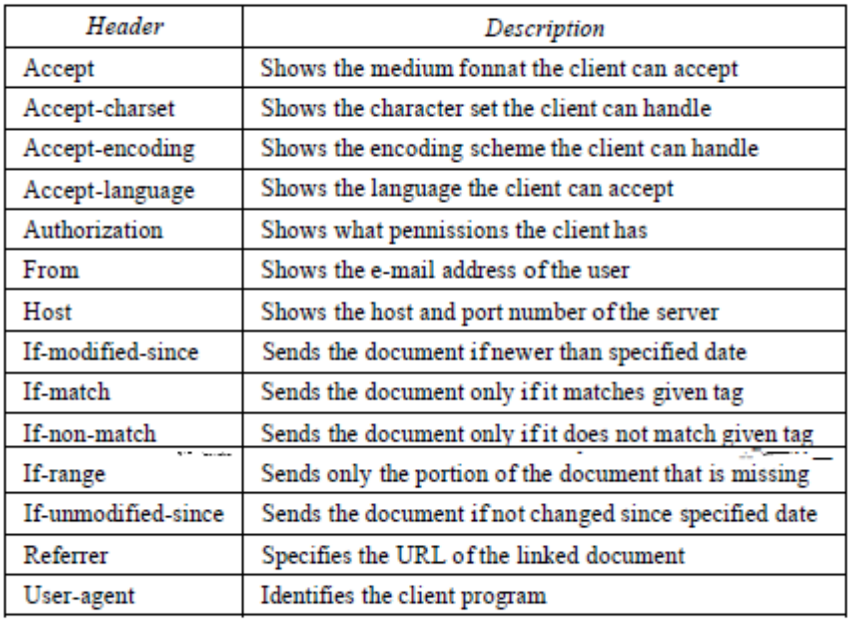
The header exchanges additional information between the client and the server. For example, the client can request that the document be sent in a special format, or the server can send extra information about the document. The header can consist of one or more header lines. Each header line has a header name, a colon, a space, and a header value. 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.

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



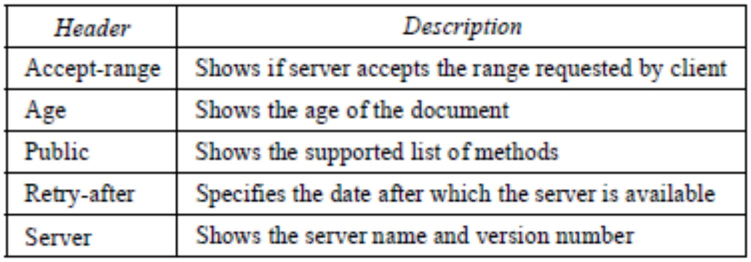
Header	Description
Cache-control	Specifies infonnation about caching
Connection	Shows whether the connection should be closed or not
Date	Shows the current date
MIME-version	Shows the MIME version used
Upgrade	Specifies the preferred communication protocol

b) 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 below for a list of some request headers and their descriptions.



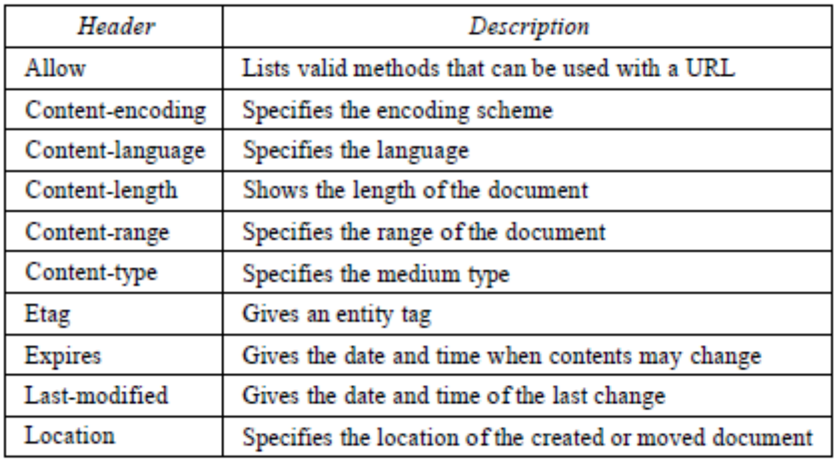
Header	*Description*
Accept	Shows the medium fonnat the client can accept
Accept-charset	Shows the character set the client can handle
Accept-encoding	Shows the encoding scheme the client can handle
Accept-language	Shows the language the client can accept
Authorization	Shows what pennissions the client has
From	Shows the e-mail address ofthe user
Host	Shows the host and port number ofthe server
If-modified-since	Sends the document ifnewer than specified date
If-match	Sends the document only ifit matches given tag
If-non-match	Sends the document only ifit does not match given tag
If-range	Sends only the portion of the document that is missing
If-unmodified-since	Sends the document ifnot changed since specified date
Referrer	Specifies the URL ofthe linked document
User-agent	Identifies the client program

c) 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 below for a list of some response headers with their descriptions.



Header	*Description*
Accept-range	Shows if server accepts the range requested by client
Age	Shows the age ofthe 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

d) 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 below for a list of some entity headers and their descriptions.



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 ofthe document
Content-range	Specifies the range ofthe 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 ofthe last change
Location	Specifies the location ofthe 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.

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:
a) The client opens a TCP connection and sends a request.
b) The server sends the response and closes the connection.
c) 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. HTTP version 1.1 specifies a persistent connection by default.

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.