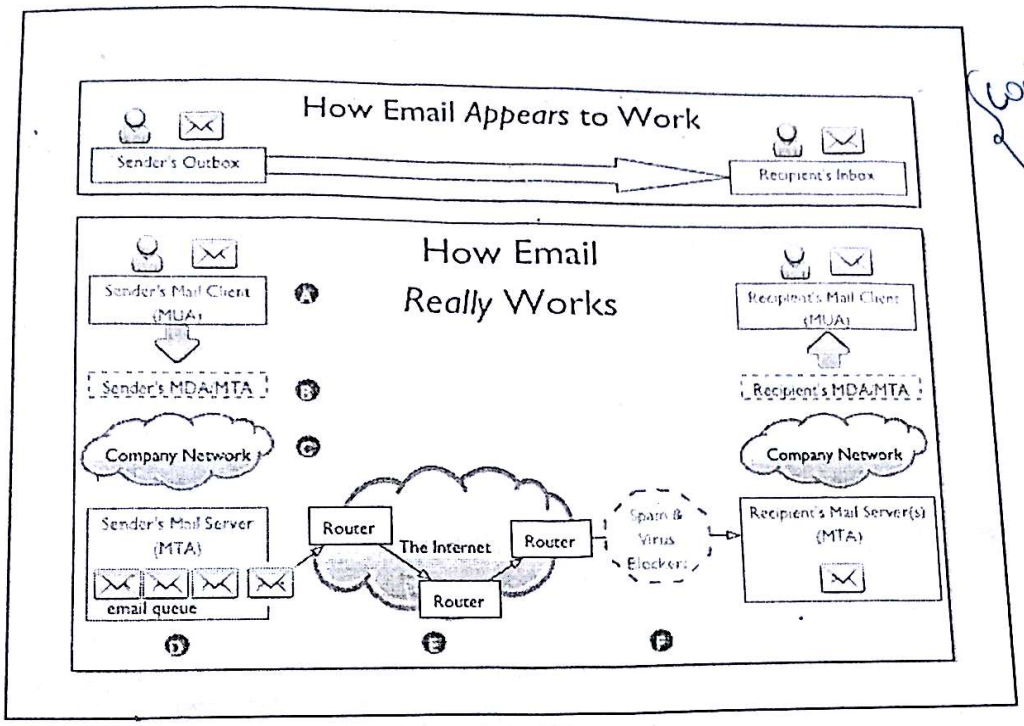
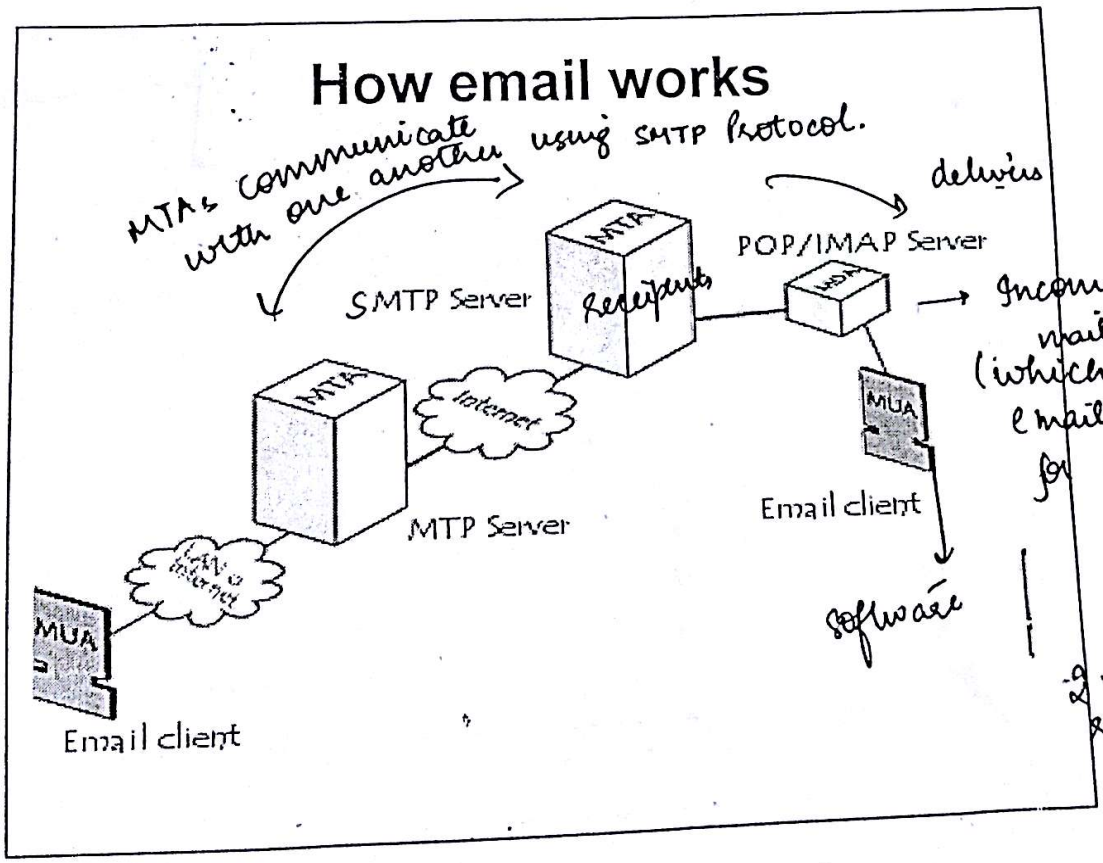
7/1

8/10/2011



(Or More)



When an e-mail is sent message is

2 mail protocols to retrieve mail on an MDA

POP3 or (Post office Protocol)

IMAP (Internet Message access Protocol)

7.1 - E-mail

8/10/2011

Every email provider supplies you with three fundamental services:

- A way to send email
- A way to receive email
- A way to store received email, at least temporarily

Electronic Mail

- Electronic Mail is a method of sending a message from a user at a computer to a recipient on another computer.
- E-mail systems are based on a **store-and-forward model** in which e-mail computer server systems accept, forward, deliver and store messages on behalf of users, who only need to connect to the e-mail infrastructure, typically an e-mail server, with a network-enabled device (e.g., a personal computer) for the duration of message submission or retrieval.

① accept
② forward
③ deliver

Email addresses

Email addresses (both for senders and recipients) are two strings separated by the character "@" (the "at sign"):

user@domain

The right-hand part describes the domain name involved, and the left-hand part refers to the user who belongs to that domain.

An email address can be up to 255 characters long and can include the following characters:

- Lowercase letters from a to z;
- Digits
- The characters ".","_" and "-" (full stop, underscore, and hyphen)

In practice, an email address often looks something like this:

firstname.lastname@provider.domain

- Email is based around the use of electronic mailboxes.
- When an email is sent, the message is routed from server to server, all the way to the recipient's email server.
- More precisely, the message is sent to the mail server tasked with transporting emails (called the **MTA**, for *Mail Transport Agent*) to the recipient's MTA.
- On the Internet, MTAs communicate with one another using the protocol SMTP, and so are logically called **SMTP servers** (or sometimes *outgoing mail servers*).

The recipient's MTA then delivers the email to the incoming mail server (called the **MDA**, for *Mail Delivery Agent*), which stores the email as it waits for the user to accept it. There are two main protocols used for retrieving email on an MDA:

- **POP3** (***Post Office Protocol***), the older of the two, which is used for retrieving email and, in certain cases, leaving a copy of it on the server.
- **IMAP** (***Internet Message Access Protocol***), which is used for coordinating the status of emails (read, deleted, moved) across multiple email clients. With IMAP, a copy of every message is saved on the server, so that this synchronization task can be completed.

For this reason, incoming mail servers are called **POP servers** or **IMAP servers**, depending on which protocol is used.

To use a real-world analogy, MTAs act as the post office (the sorting area and mail carrier, which handle message transportation), while MDAs act as mailboxes, which store messages (as much as their volume will allow) until the recipients check the box. This means that it is not necessary for recipients to be connected in order for them to be sent email.

- To keep everyone from checking other users' emails, MDA is protected by a user name called a **login** and by a **password**.

- Retrieving mail is done using a software program called an **MUA** (*Mail User Agent*).

- When the MUA is a program installed on the user's system, it is called an **email client** (such as Mozilla Thunderbird, Microsoft Outlook, Eudora Mail, Incredimail or Lotus Notes).

- When it is a web interface used for interacting with the incoming mail server, it is called **webmail**.

Types of programs used for E-mail.

- **Email Clients.** Email clients are computer programs that **run on your local computer** which enable you to perform actions such as composing and sending email messages, retrieving your new email messages, and looking at your old email messages.
- Examples of email clients include: Microsoft Outlook, Microsoft Outlook Express, Eudora, Pine, Netscape Communicator, and Entourage.
- Email clients are characterized by being installed on your local machine and **requiring you to configure** them **with details of your email provider** so that they can send and receive your email.

(Tenenbaum)

- **WebMail.** WebMail is like an email client in that it allows you to compose and send email messages, view new and old email messages, and perform other email activities.
- However, unlike an email client, WebMail programs **run on your email provider's web servers and are accessible via any computer connected to the Internet that has a compatible web browser, such as a recent version of Internet Explorer or Netscape Navigator.**
- WebMail programs **do not require configuration**, you simply login and they work.

Structure of an email

An email has two basic parts:

- The **message** header Fields
- The **message** body

The **header**

- Header — Structured into fields such as summary, sender, receiver, and other information about the e-mail.
- The header begins with a *From* line and is changed each time it passes through an intermediary server.
- Using headers, you can see the exact path taken by the email, and how long it took each server to process.

HTTP overview (continued)

Uses TCP:

- client initiates TCP connection (creates socket) to server, port 80
- server accepts TCP connection from client
- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)
- TCP connection closed

HTTP is "stateless"

- server maintains no information about past client requests

aside

Protocols that maintain "state" are complex!

- past history (state) must be maintained
- if server/client crashes, their views of "state" may be inconsistent, must be reconciled

HTTP connections

Nonpersistent HTTP

- At most one object is sent over a TCP connection.
- HTTP/1.0 uses nonpersistent HTTP

Persistent HTTP

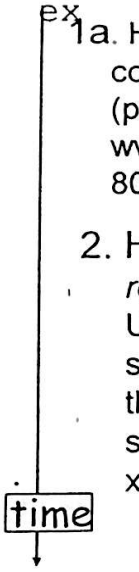
- Multiple objects can be sent over single TCP connection between client and server.
- HTTP/1.1 uses persistent connections in default mode

Nonpersistent HTTP

Suppose user enters URL

`www.someSchool.edu/someDepartment/home.index`

(contains text, references to 10 jpeg images)



1a. HTTP client initiates TCP connection to HTTP server (process) at www.someSchool.edu on port 80

1b. HTTP server at host www.someSchool.edu waiting for TCP connection at port 80. "accepts" connection, notifying client

2. HTTP client sends HTTP *request message* (containing URL) into TCP connection socket. Message indicates that client wants object someDepartment/home.index

3. HTTP server receives request message, forms *response message* containing requested object, and sends message into its socket

Nonpersistent HTTP (cont.)

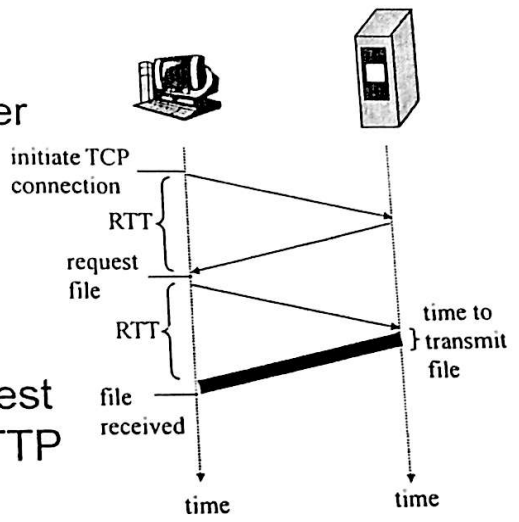
4. HTTP server closes TCP connection.

5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects

6. Steps 1-5 repeated for each of 10 jpeg objects

Non-Persistent HTTP: Response time

Definition of RTT: time to send a small packet to travel from client to server and back.



Response time:

- one RTT to initiate TCP connection
- one RTT for HTTP request and first few bytes of HTTP response to return
- file transmission time

total = 2RTT+transmit time

Persistent HTTP

Nonpersistent HTTP issues:

- requires 2 RTTs per object
- OS overhead for *each* TCP connection
- browsers often open parallel TCP connections to fetch referenced objects

Persistent HTTP

- server leaves connection open after sending response
- subsequent HTTP messages between same client/server sent over open connection

Persistent ***without*** pipelining:

- client issues new request only when previous response has been received
- one RTT for each referenced object

Persistent ***with*** pipelining:

- default in HTTP/1.1
- client sends requests as soon as it encounters a referenced object
- as little as one RTT for all the referenced objects

HTTP request message

- two types of HTTP messages: *request*, *response*
- HTTP request message:
  - ASCII (human-readable format)

request line (GET, POST, HEAD commands)

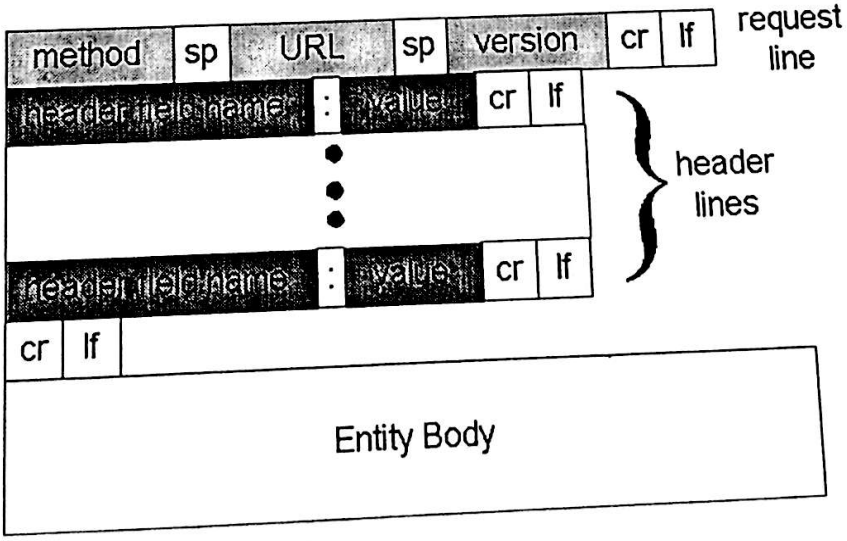
```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language:fr
```

header lines

Carriage return, line feed indicates end of message

(extra carriage return, line feed)

HTTP request message: general format



Uploading form input

Post method:

- Web page often includes form input
- Input is uploaded to server in entity body

URL method:

- Uses GET method
- Input is uploaded in URL field of request line:

```
www.somesite.com/animalsearch?monkeys&banana
```

Method types

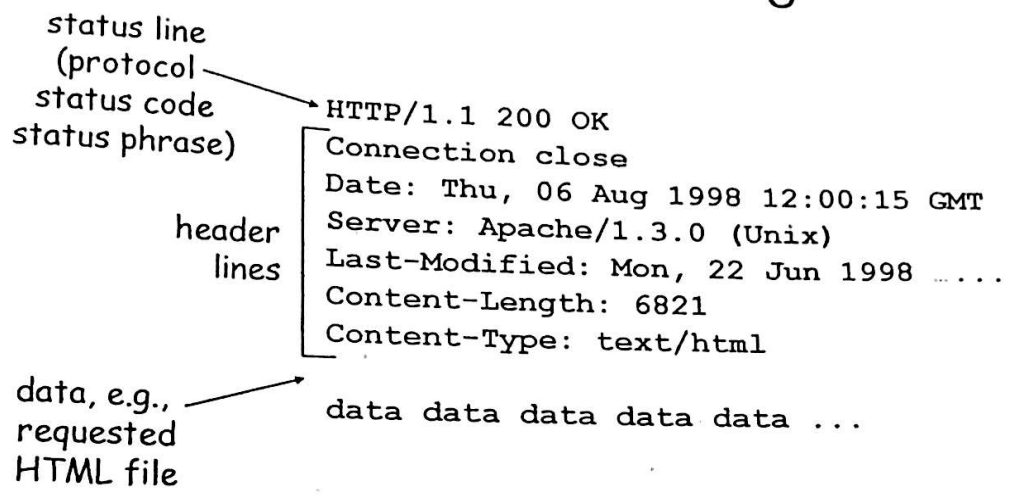
HTTP/1.0

- GET
- POST
- HEAD
  - asks server to leave requested object out of response

HTTP/1.1

- GET, POST, HEAD
- PUT
  - uploads file in entity body to path specified in URL field
- DELETE
  - deletes file specified in the URL field

HTTP response message



HTTP response status codes

In first line in server->client response message.

A few sample codes:

200 OK

- request succeeded, requested object later in this message

301 Moved Permanently

- requested object moved, new location specified later in this message (Location:)

400 Bad Request

- request message not understood by server

404 Not Found

- requested document not found on this server

505 HTTP Version Not Supported

2.5 MESSAGE FORMAT

As mentioned earlier, HTTP is a request–response protocol [Figure 2.2]. It specifies a set of rules that clients and servers use to communicate:

- An HTTP server process is created on a port (usually 80), which waits for clients to establish a TCP connection.
- An HTTP client initiates a TCP connection with the HTTP server (process) at the designated port.
- The HTTP server accepts this connection.
- The HTTP client then sends a *request* for a resource to the server.
- Upon receiving the *request*, the server processes the request, performs the desired task, and sends a *response* back to the client.
- The HTTP server closes the TCP connection.
- The HTTP client receives the *response* containing information and processes it.

Note that every time the HTTP client wants to get resource from the server, it has to follow these steps. This makes the HTTP *stateless*. This means that web server treats every request as a new request. There is no way to specify that some requests are related.

The advantage of this mechanism is that the client/server need not retain information between successive requests. However, keeping the information about successive related requests is sometimes necessary. As HTTP protocol is inherently stateless, designer of the web pages must use alternative methods such as Cookies, URL rewriting, etc. to remember the previous request(s).

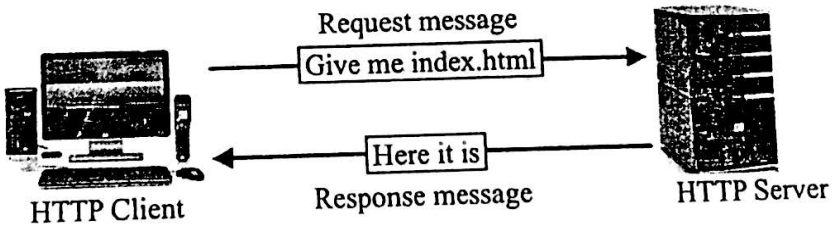


Figure 2.2 HTTP request and response

These requests and responses are encapsulated via HTTP messages [Figure 2.3]. There are two types of messages: *request message* and *response message*.

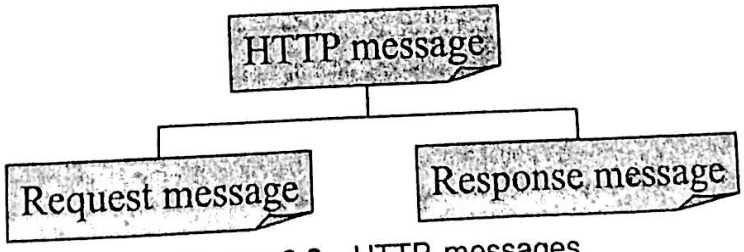


Figure 2.3 HTTP messages

The following section describes the format of each of these two messages.

2.5.1 Request Message

A *request message* is sent by a web client to the web server. It consists of the following parts [Figure 2.4]:

- A request line
- A header
- An empty line
- An optional body

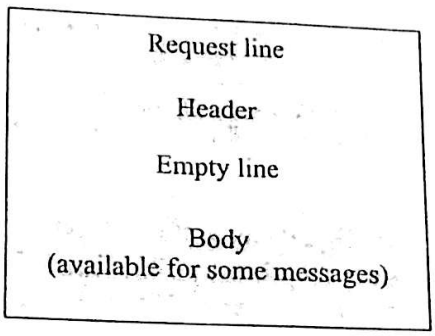


Figure 2.4 HTTP request message format

2.5.1.1 Request line

A request line consists of three parts: *request type*, *URL*, and *HTTP version* [Figure 2.5]. Two consecutive parts are separated by a space.

Figure 2.5 HTTP request line

2.5.1.2 Request type (method)

It indicates the type of the request, a client wants to send. They are also called *methods*. A method makes a message either a request or a command to the server. Request messages are used to retrieve data from the server whereas a command tells the server to do a specific task. Some of the HTTP methods are discussed here:

GET

This is the most frequently used method in the WWW. It is specified when a client wants to retrieve (GET) a resource (document) from the server. The URL in the request line identifies the resource. If the URL specified is a valid one, the server reads the content of the resource and sends the content back to the client; otherwise an error message is sent back to the client. If the resource being requested is a server-side program such as a CGI script, or ASP or JSP, the result generated by that program is returned instead of the content of the resource. The message body is empty for the GET method.

2.5 MESSAGE FORMAT

As mentioned earlier, HTTP is a request–response protocol [Figure 2.2]. It specifies a set of rules that clients and servers use to communicate:

- An HTTP server process is created on a port (usually 80), which waits for clients to establish a TCP connection.
- An HTTP client initiates a TCP connection with the HTTP server (process) at the designated port.
- The HTTP server accepts this connection.
- The HTTP client then sends a *request* for a resource to the server.
- Upon receiving the *request*, the server processes the request, performs the desired task, and sends a *response* back to the client.
- The HTTP server closes the TCP connection.
- The HTTP client receives the *response* containing information and processes it.

Note that every time the HTTP client wants to get resource from the server, it has to follow these steps. This makes the HTTP *stateless*. This means that web server treats every request as a new request. There is no way to specify that some requests are related.

The advantage of this mechanism is that the client/server need not retain information between successive requests. However, keeping the information about successive related requests is sometimes necessary. As HTTP protocol is inherently stateless, designer of the web pages must use alternative methods such as Cookies, URL rewriting, etc. to remember the previous request(s).

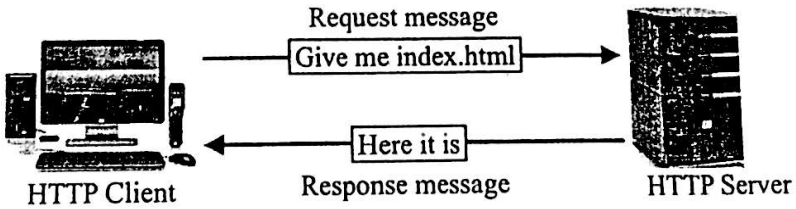


Figure 2.2 HTTP request and response

These requests and responses are encapsulated via HTTP messages [Figure 2.3]. There are two types of messages: *request message* and *response message*.

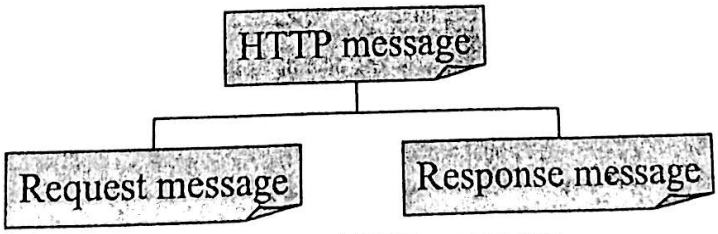


Figure 2.3 HTTP messages

The following section describes the format of each of these two messages.

2.5.1 Request Message

A *request message* is sent by a web client to the web server. It consists of the following parts [Figure 2.4]:

- A request line
- A header
- An empty line
- An optional body

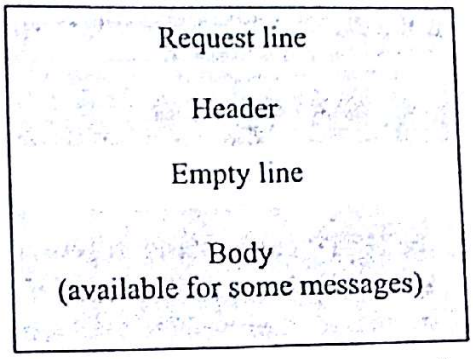


Figure 2.4 HTTP request message format

2.5.1.1 Request line

A request line consists of three parts: *request type*, *URL*, and *HTTP version* [Figure 2.5]. Two consecutive parts are separated by a space.

Figure 2.5 HTTP request line

2.5.1.2 Request type (method)

It indicates the type of the request, a client wants to send. They are also called *methods*. A method makes a message either a request or a command to the server. Request messages are used to retrieve data from the server whereas a command tells the server to do a specific task. Some of the HTTP methods are discussed here:

GET

This is the most frequently used method in the WWW. It is specified when a client wants to retrieve (GET) a resource (document) from the server. The URL in the request line identifies the resource. If the URL specified is a valid one, the server reads the content of the resource and sends the content back to the client; otherwise an error message is sent back to the client. If the resource being requested is a server-side program such as a CGI script, or ASP or JSP, the result generated by that program is returned instead of the content of the resource. The message body is empty for the GET method.