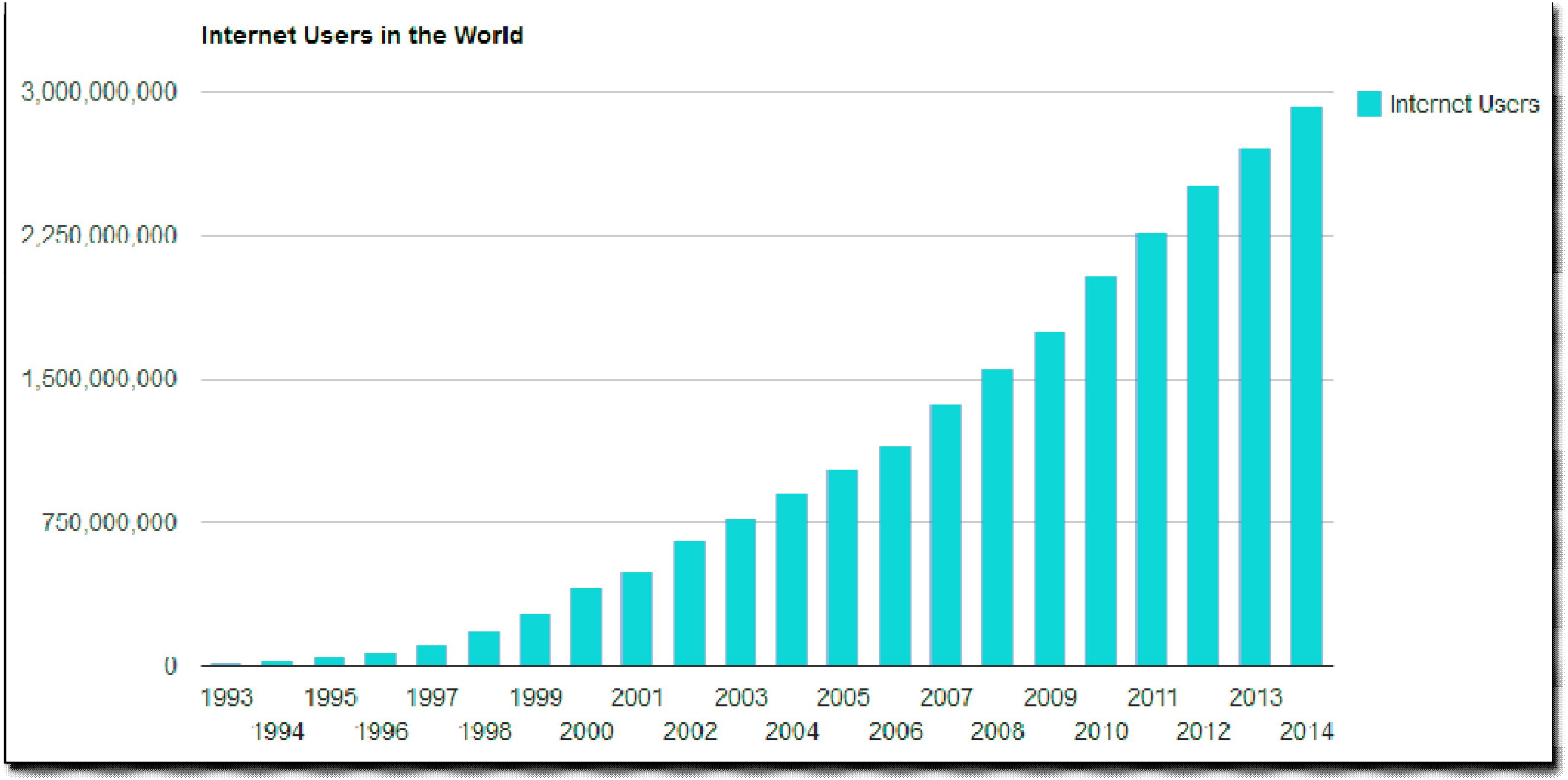
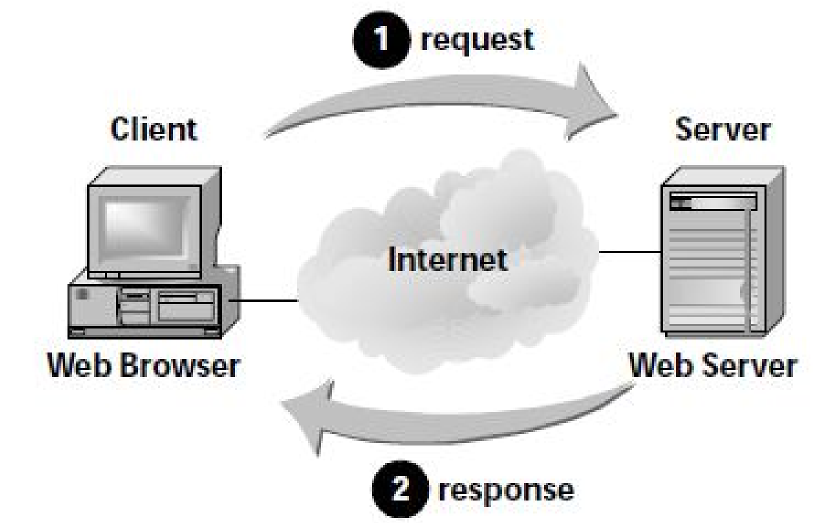
http://ww

Internet Status



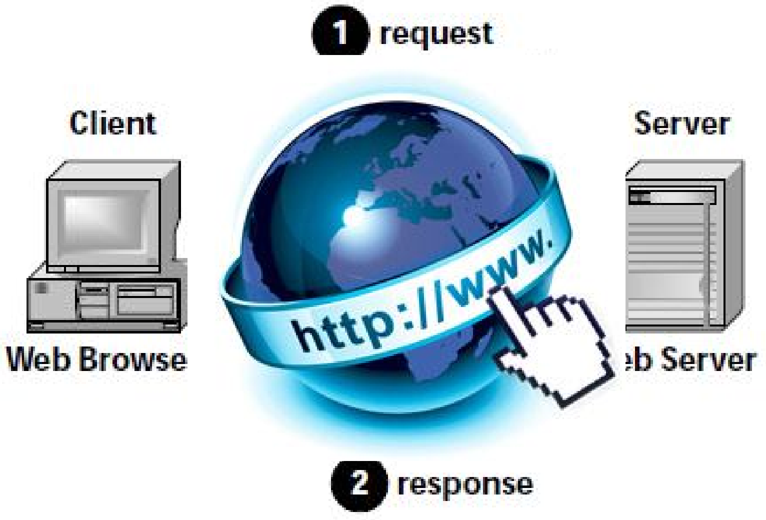
http:1 ARCHITECTURE

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.

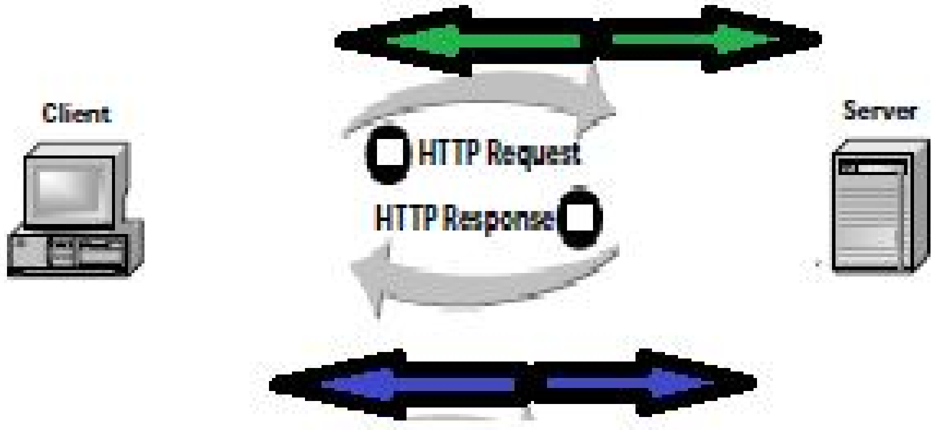


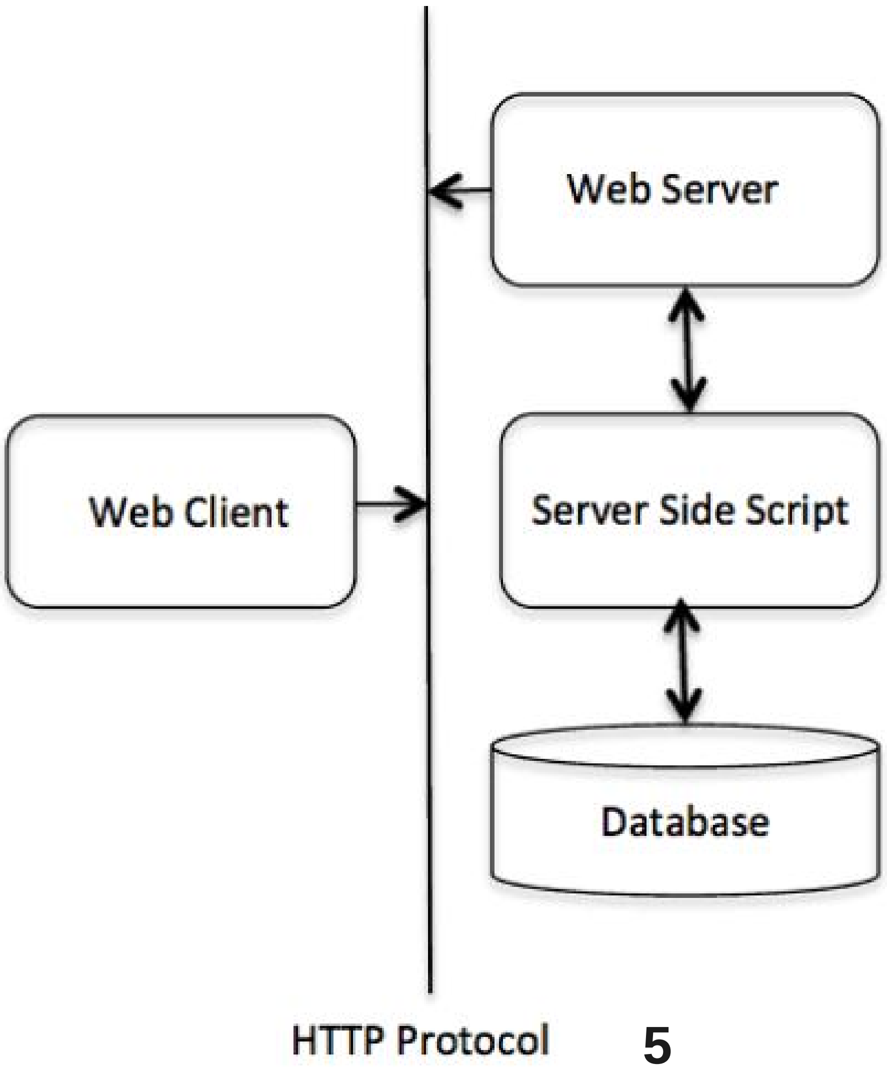
WWW and HTTP

- March **1989** that **Tim Berners-Lee** first outlined the advantages of a hypertext-based information
- **1990 Berners-Lee, & Robert Cailliau**, created the first Web browsers and servers. Called World Wide Web and later renamed Nexus.
- Browsers needed a protocol to regulate their communications;
- Berners-Lee and Cailliau designed the first version of http.

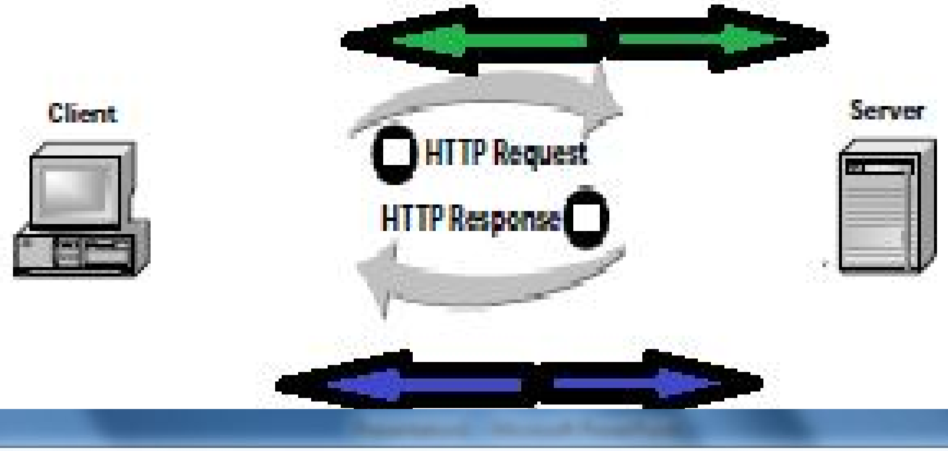


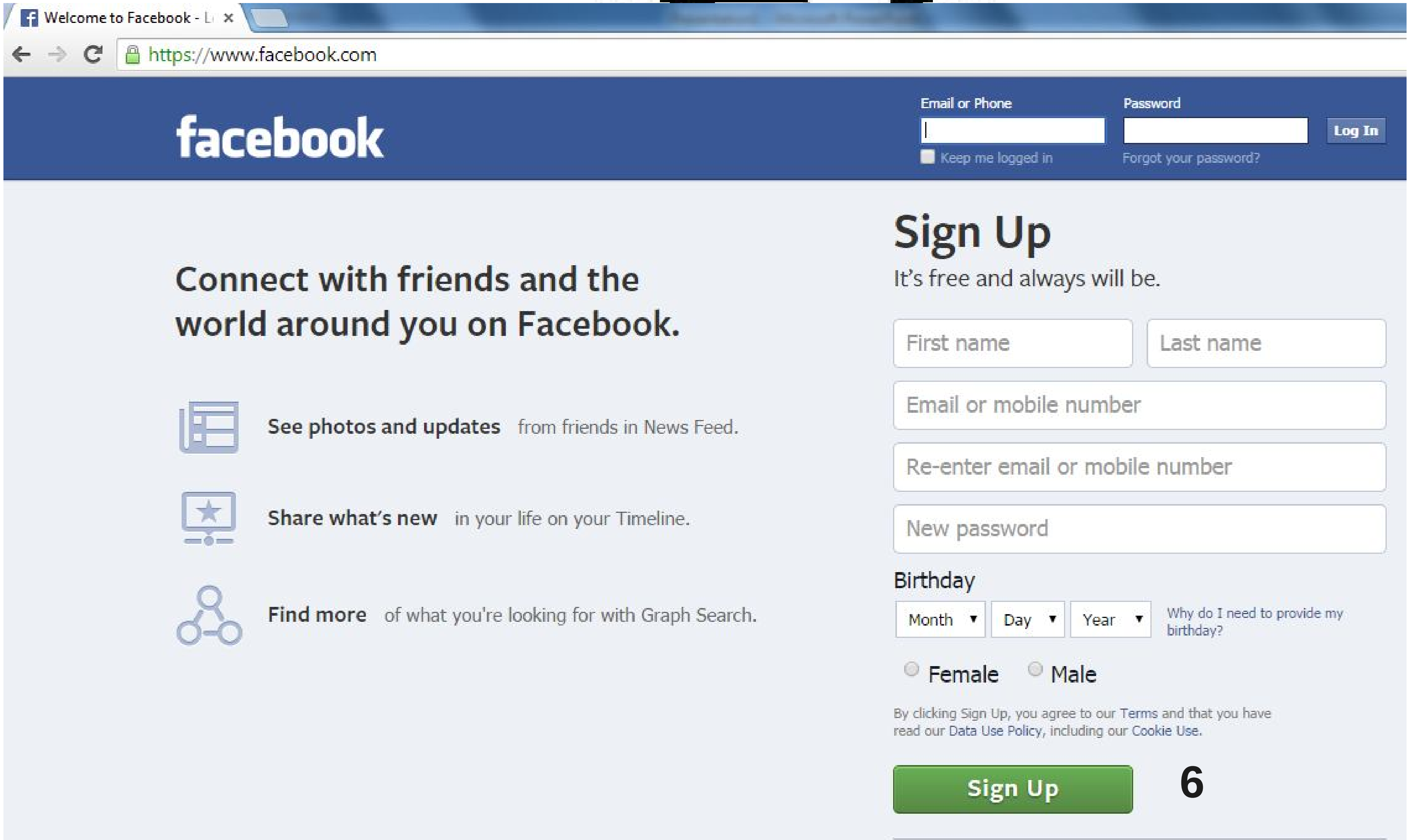
HTTP Architecture





HTTP





Email or Phone

Password

□ Keep me logged in

Forgot your password?

Connect with friends and the world around you on Facebook.

See photos and updates from friends in News Feed.

Share what's new in your life on your Timeline.

Find more of what you're looking for with Graph Search.

Sign Up

It's free and always will be.

First name

Last name

Email or mobile number

Re-enter email or mobile number

New password

Birthday

Month ▾ Day ▾ Year ▾

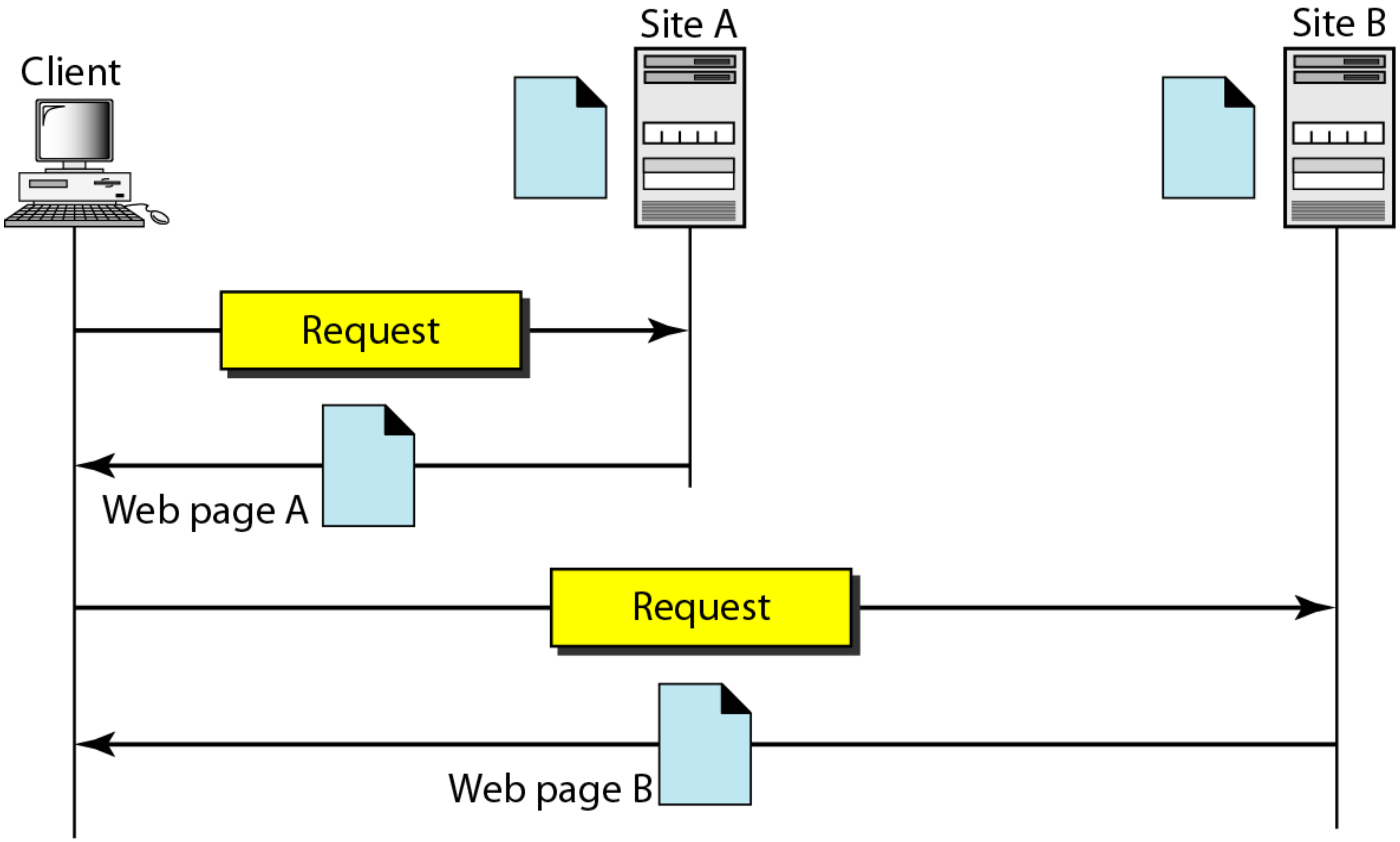
Why do I need to provide my birthday?

○ Female ○ Male

By clicking Sign Up, you agree to our Terms and that you have read our Data Use Policy, including our Cookie Use.

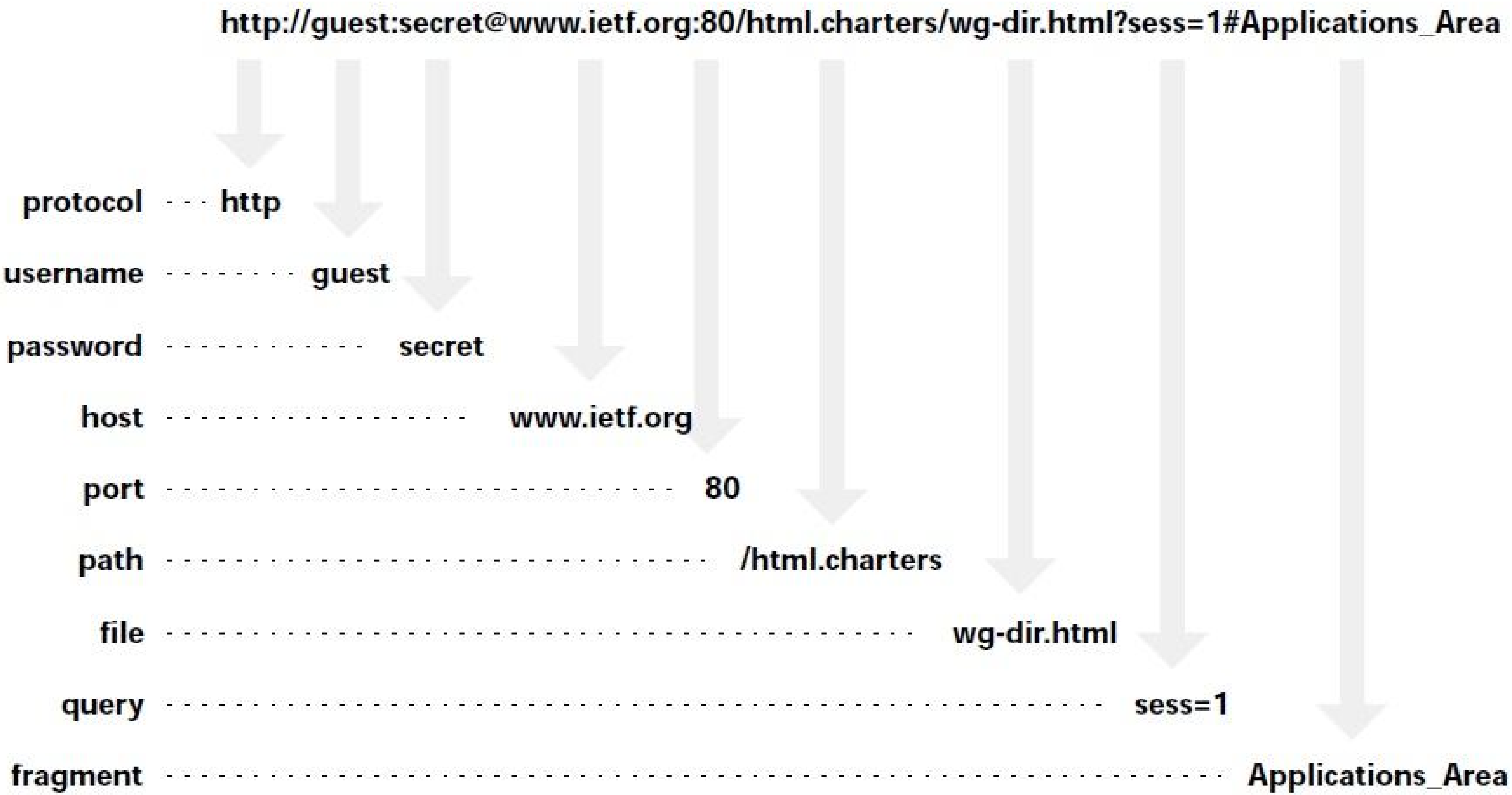
Sign Up

Figure http.1 *Architecture of WWW*



URL

Uniform Resource Identifiers



http://guest:secret@www.ietf.org:80/html.charters/wg-dir.html?sess=1#Applications_Area

protocol	http
username	guest
password	secret
host	www.ietf.org
port	80
path	/html.charters
file	wg-dir.html
query	sess=1
fragment	Applications_Area

Browser Communication

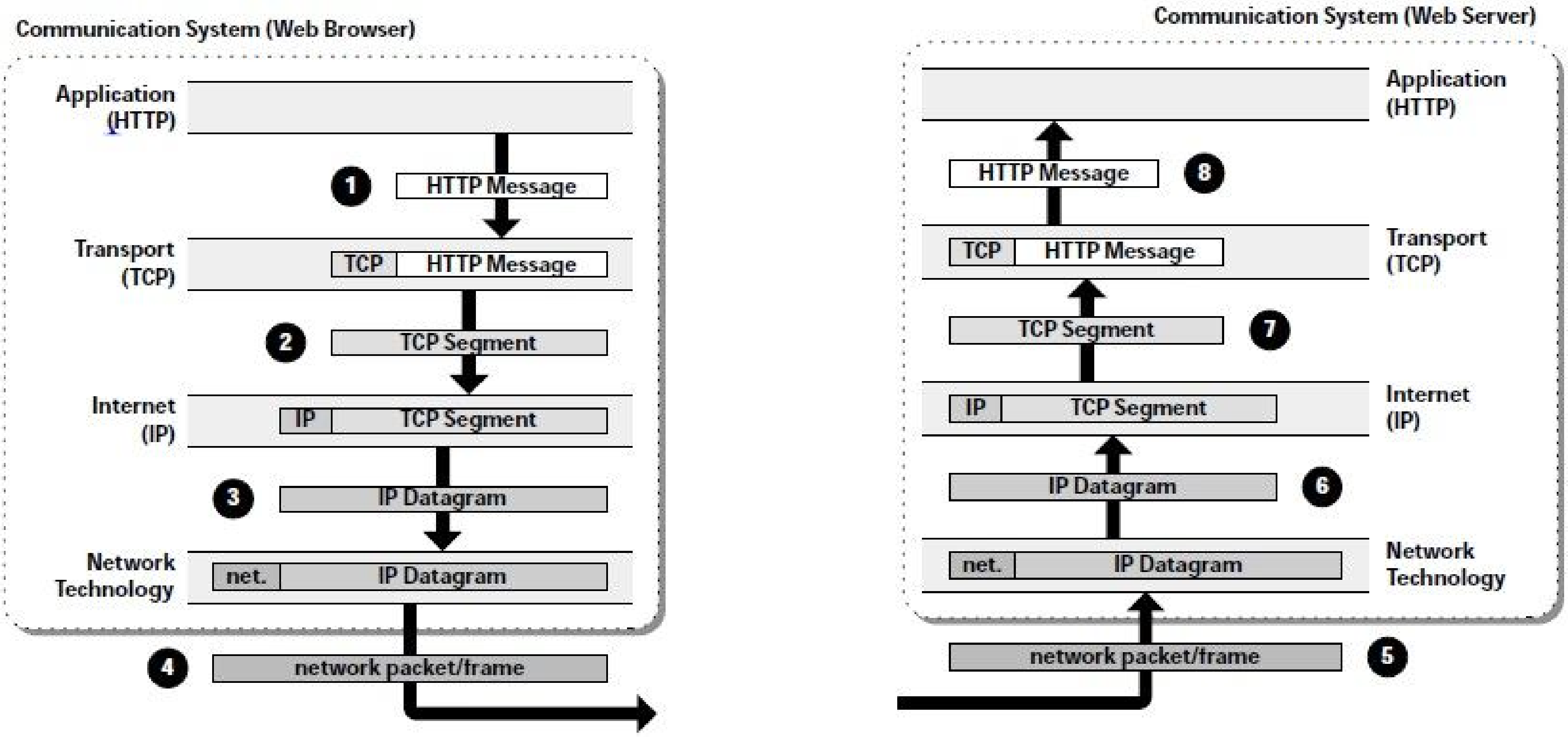
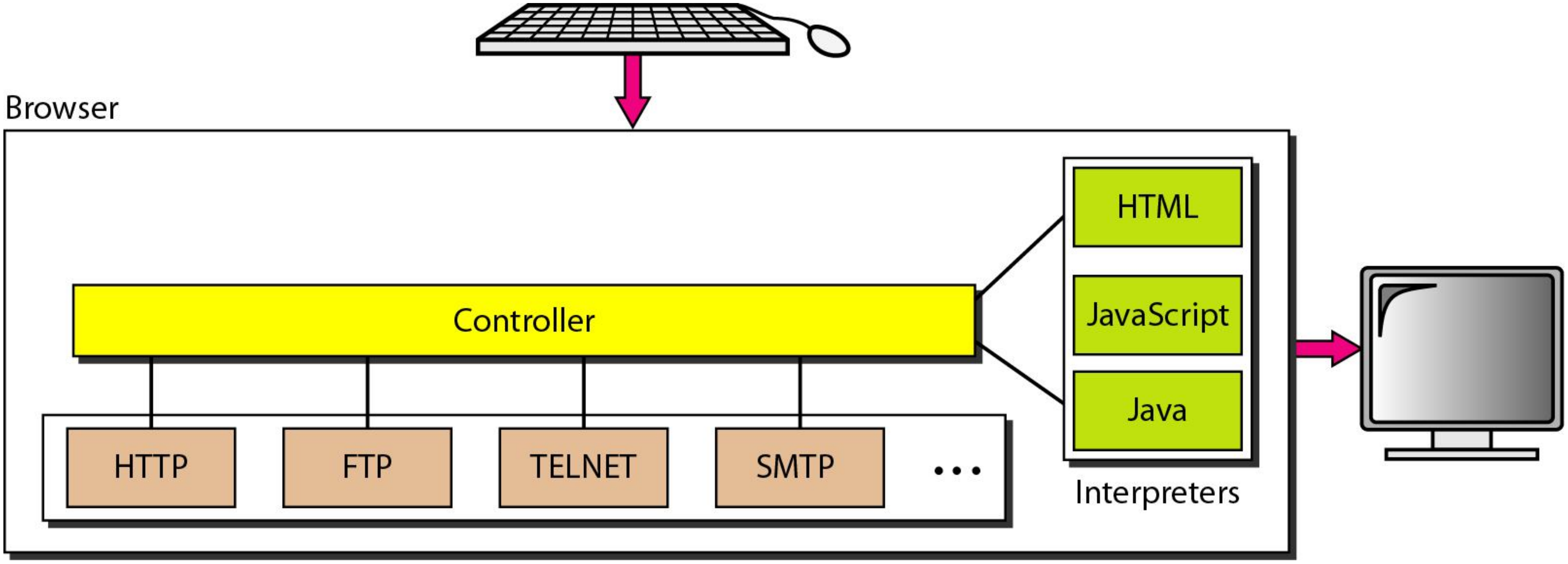


Figure http.2 *Browser*



http-2 WEB DOCUMENTS

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

Figure http.4 *Static document*

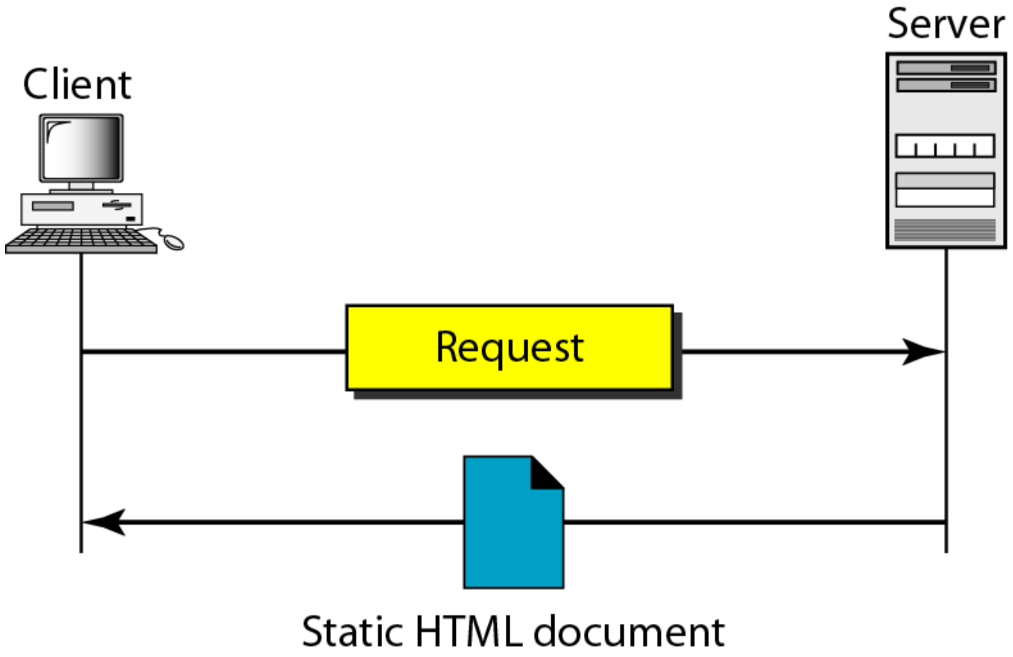
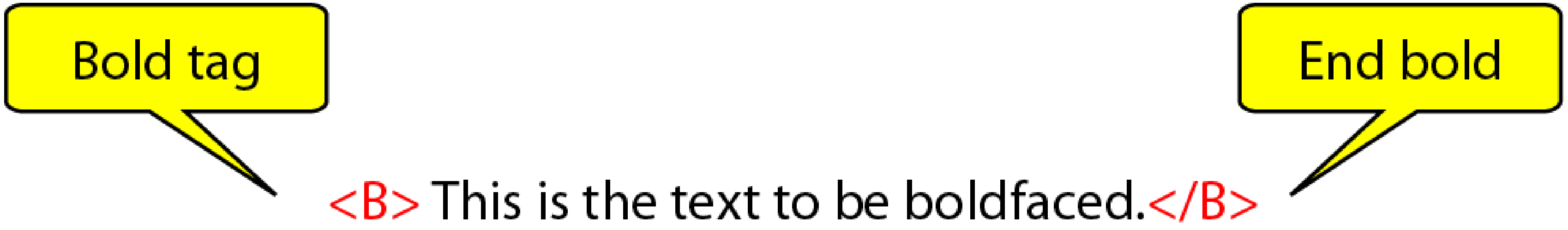
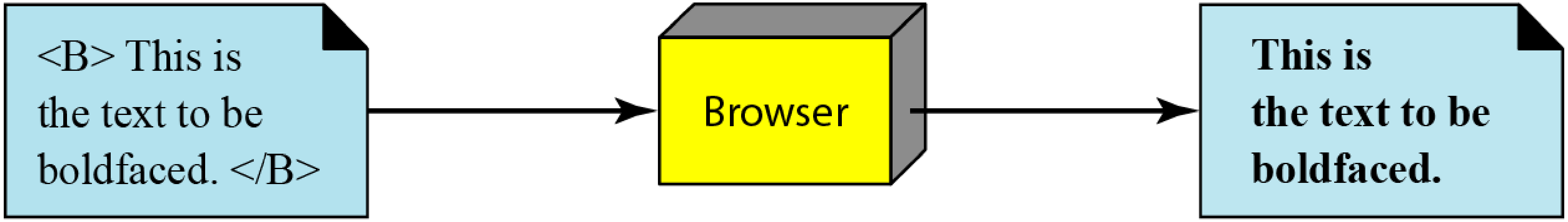


Figure http.5 *Boldface tags*



<B> This is the text to be boldfaced.</B>

Figure http.6 ***Effect of boldface tags***

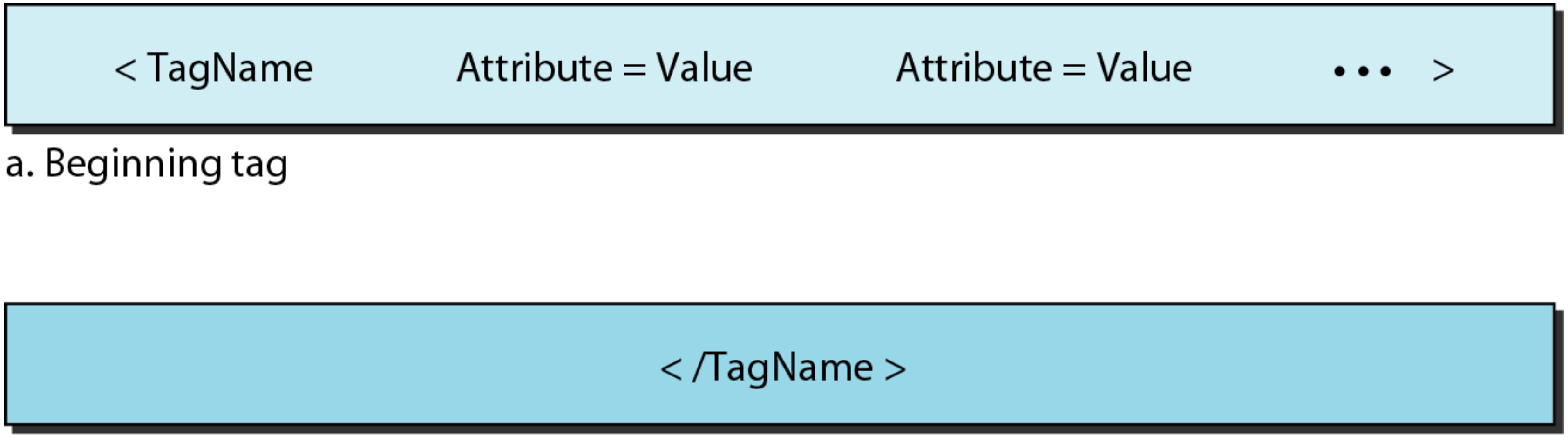


<B> This is
the text to be
boldfaced. </B>

Browser

**This is
the text to be
boldfaced.**

Figure http.7 *Beginning and ending tags*



a. Beginning tag

b. Ending tag

Figure http.8 *Dynamic document using CGI*

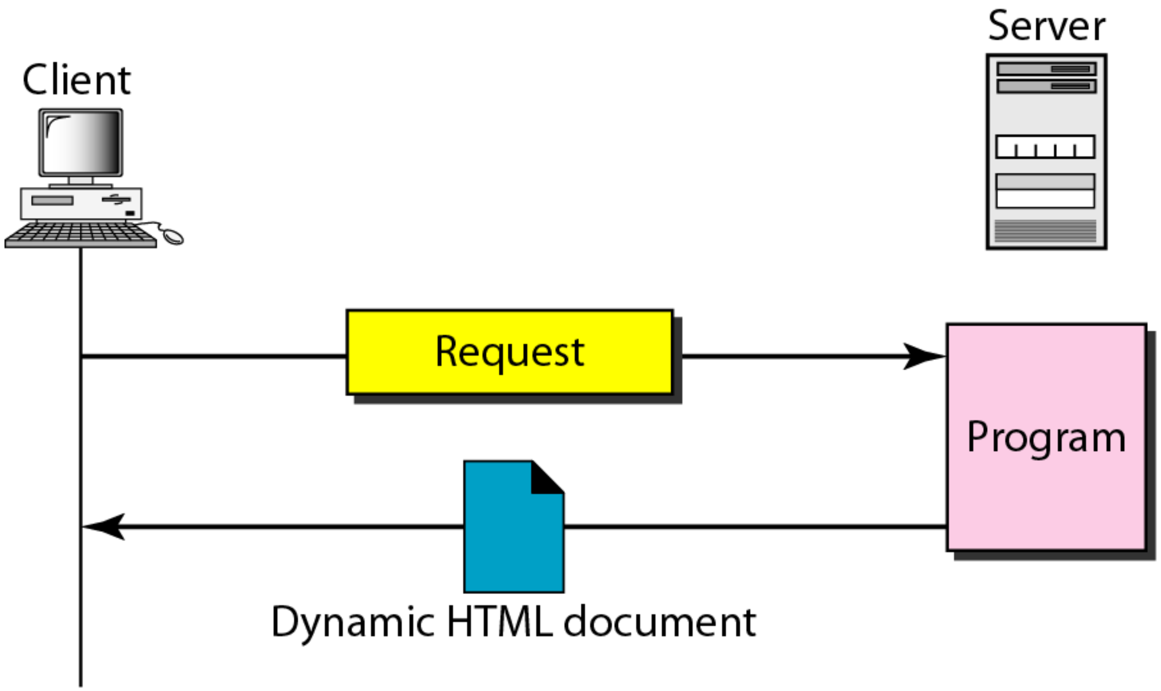
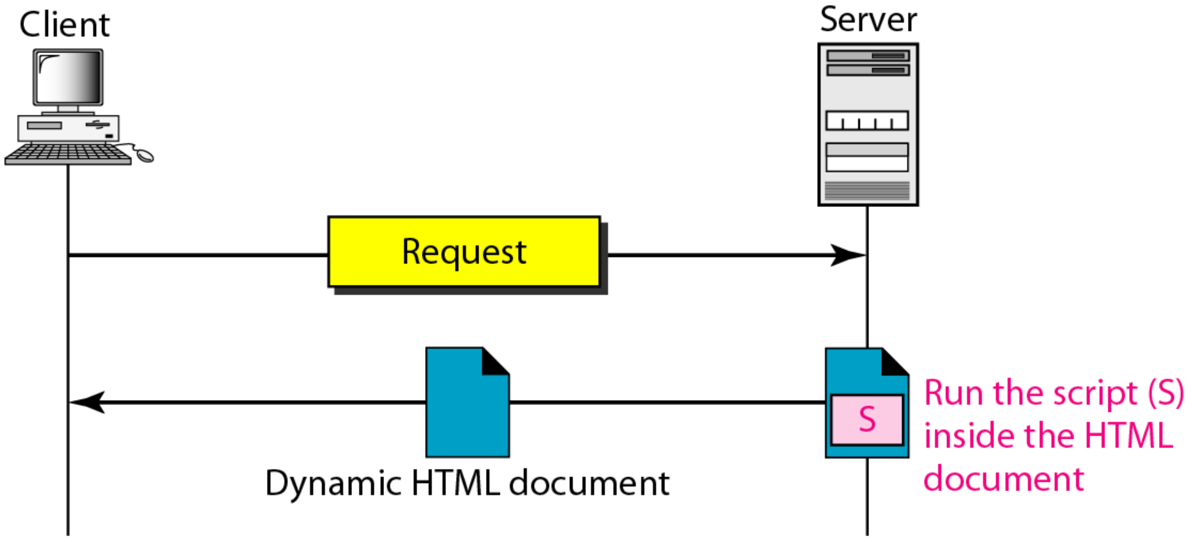


Figure http.9 *Dynamic document using server-site script*



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

Figure http.10 *Active document using Java applet*

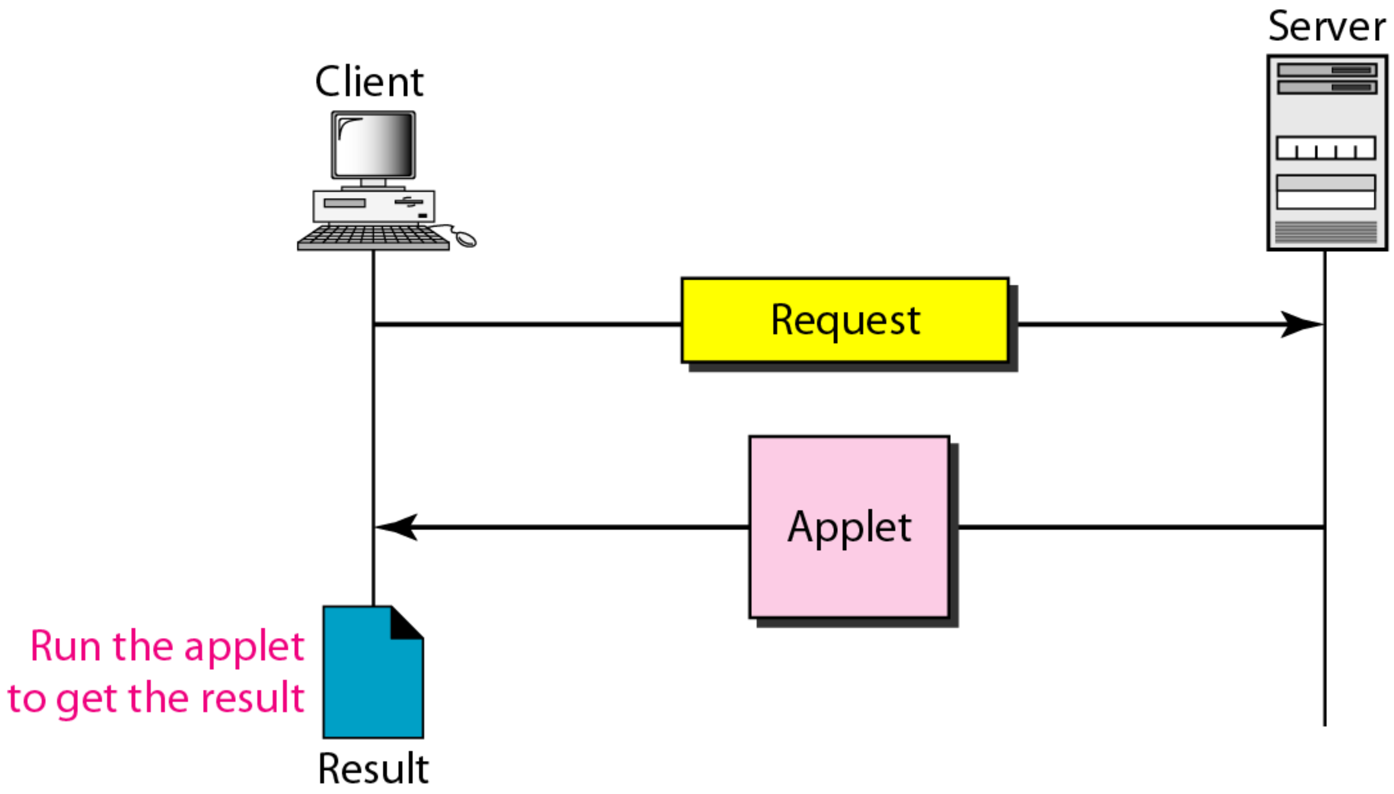
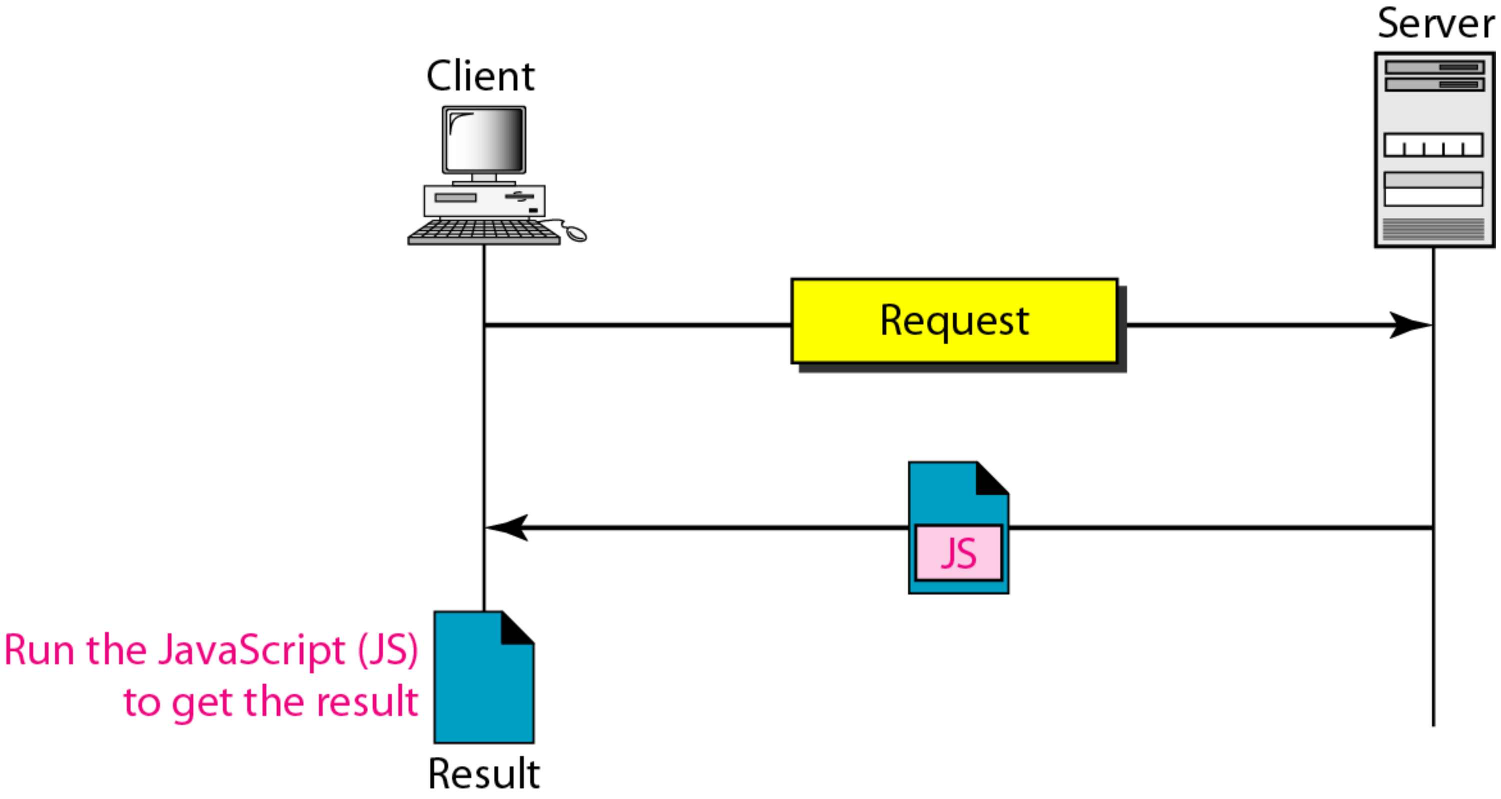


Figure http.11 *Active document using client-site script*



Note

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

http-3 HTTP

The Hypertext Transfer Protocol (HTTP) is a protocol used mainly to access data on the World Wide Web. HTTP functions as a combination of FTP and SMTP.

Note

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

Figure http.12 *HTTP transaction*

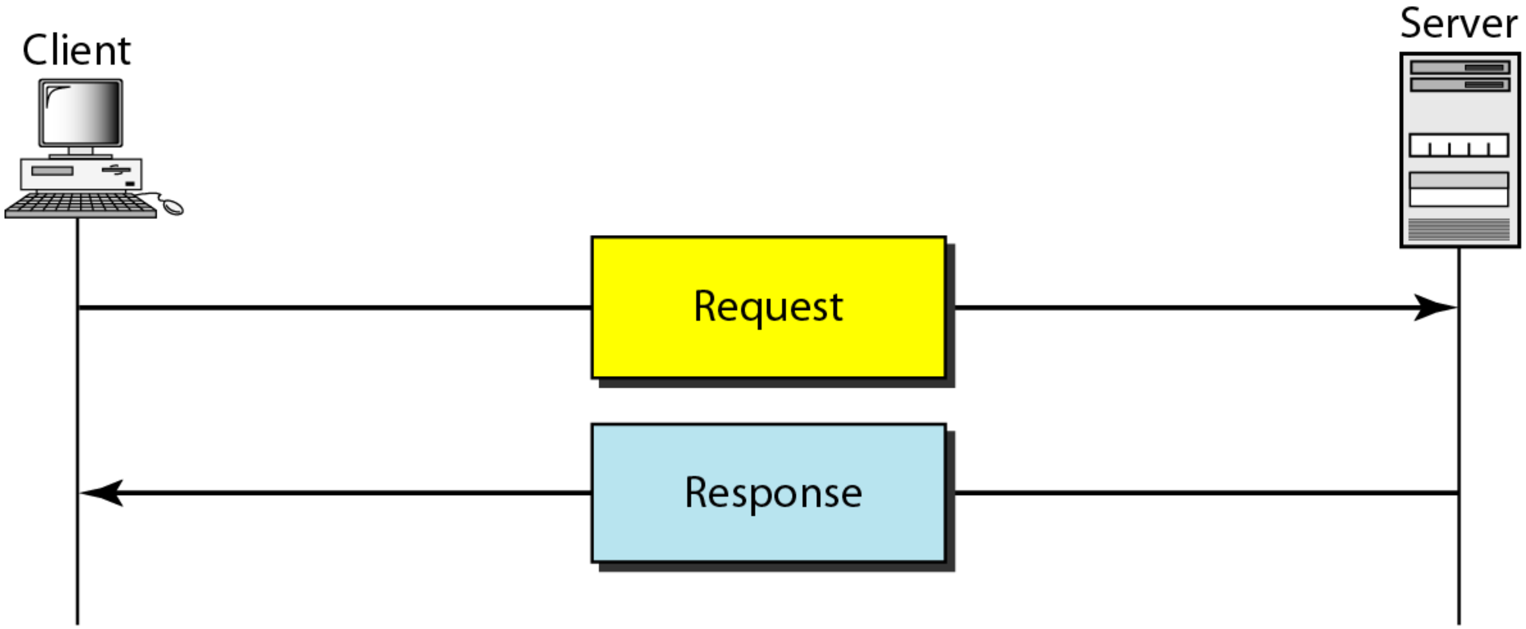


Figure http.13 *Request and response messages*

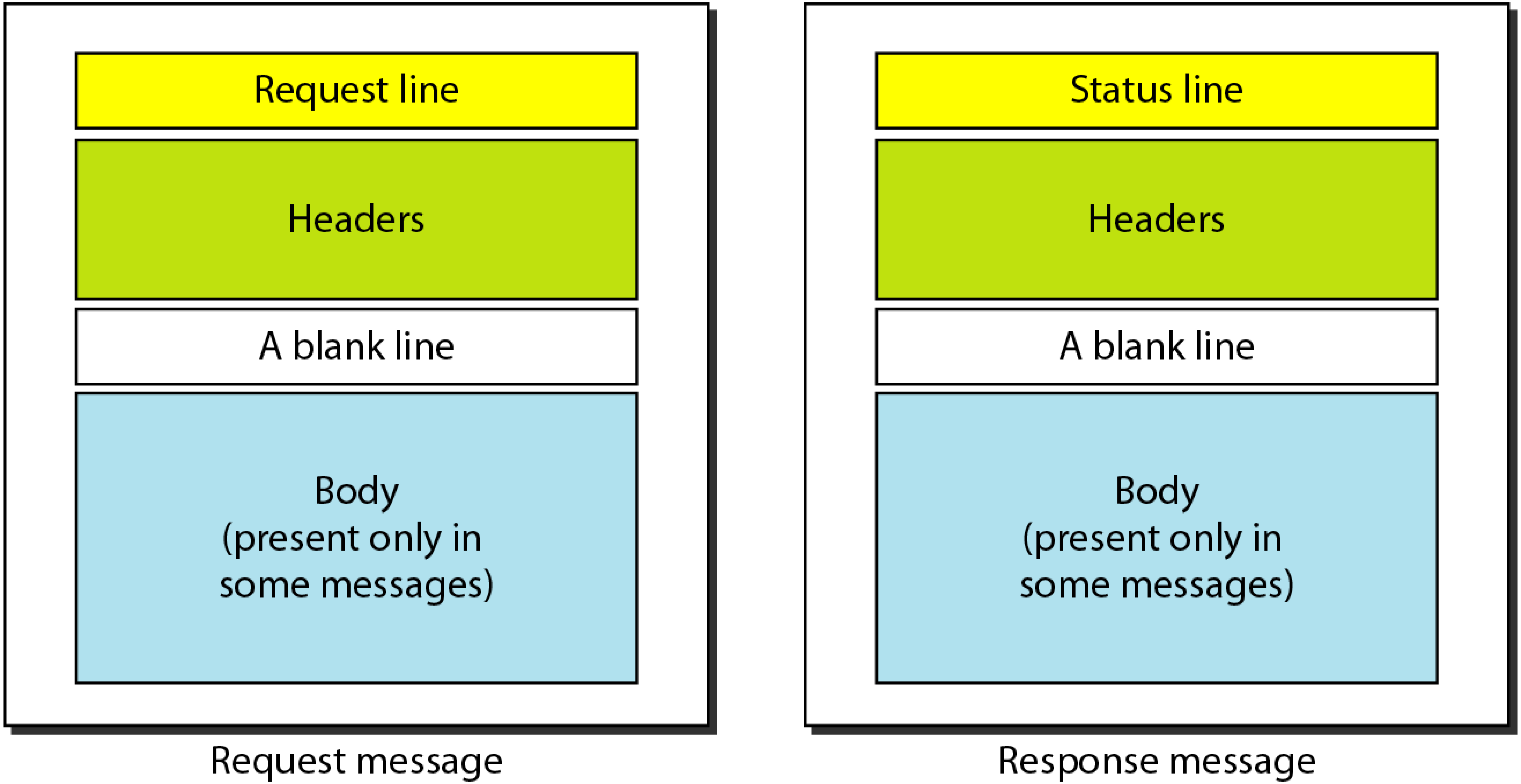


Figure http.14 *Request and status lines*

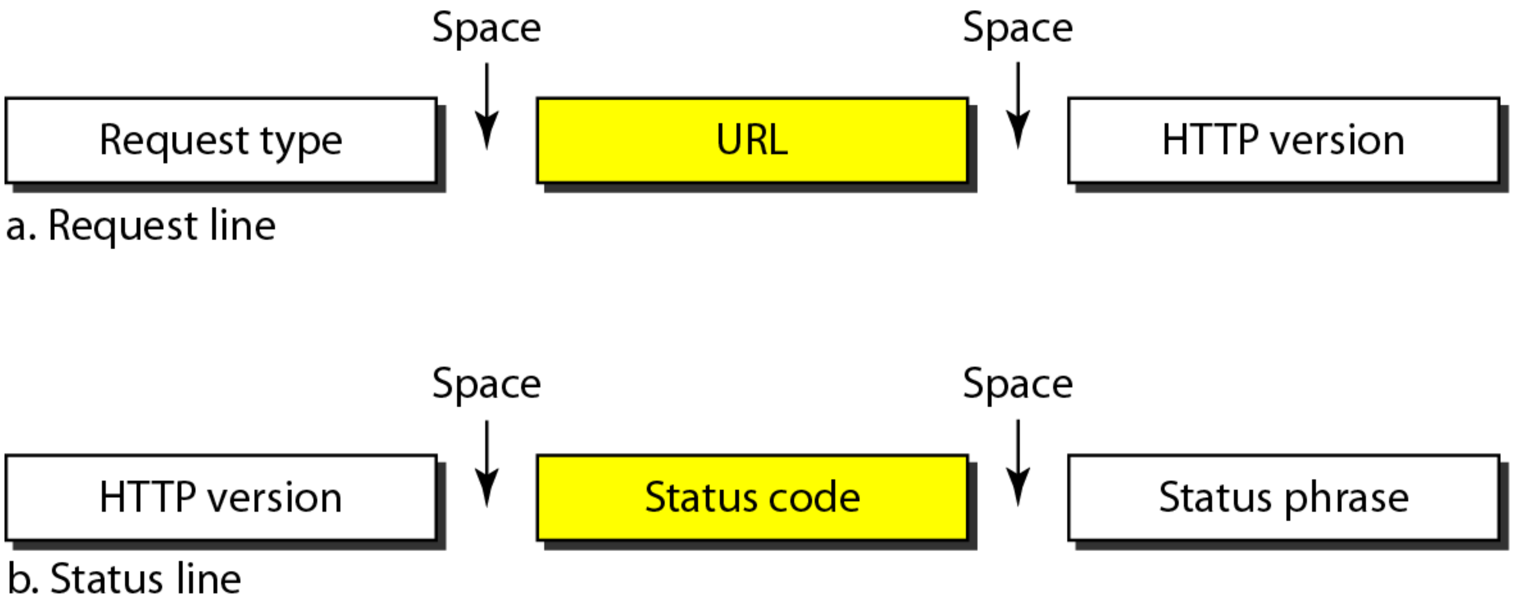
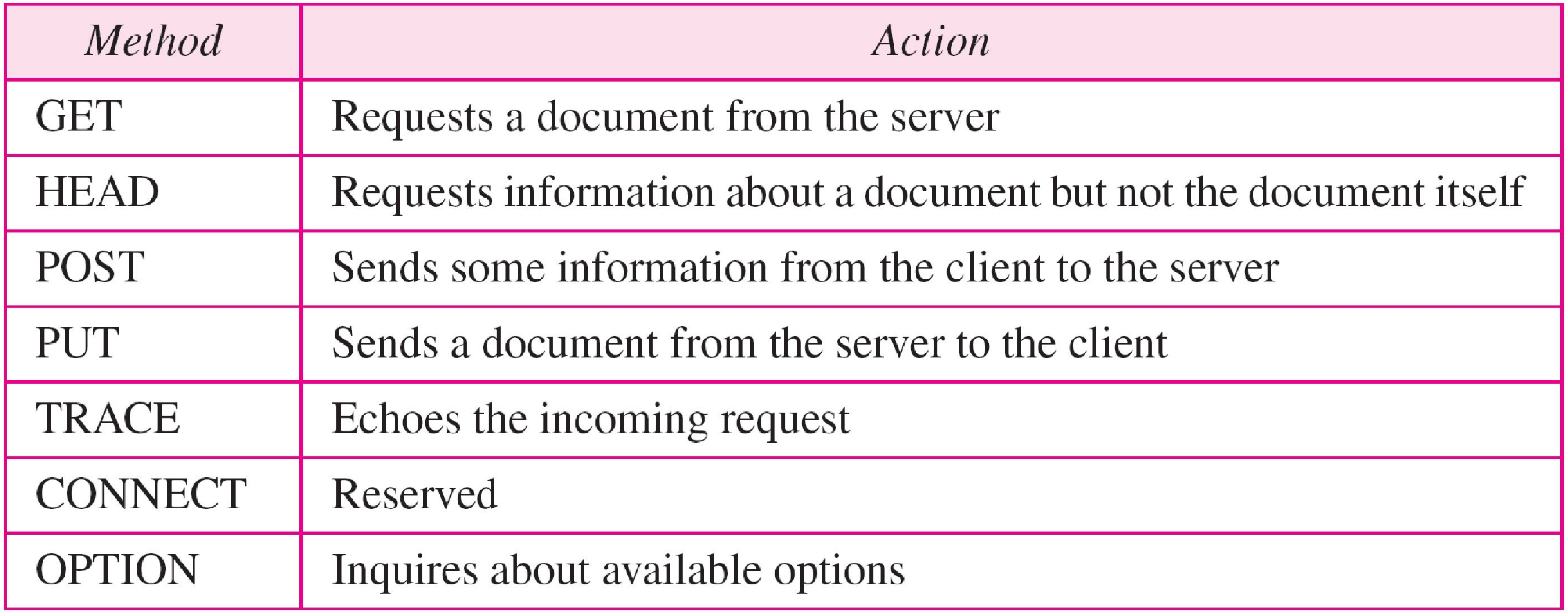
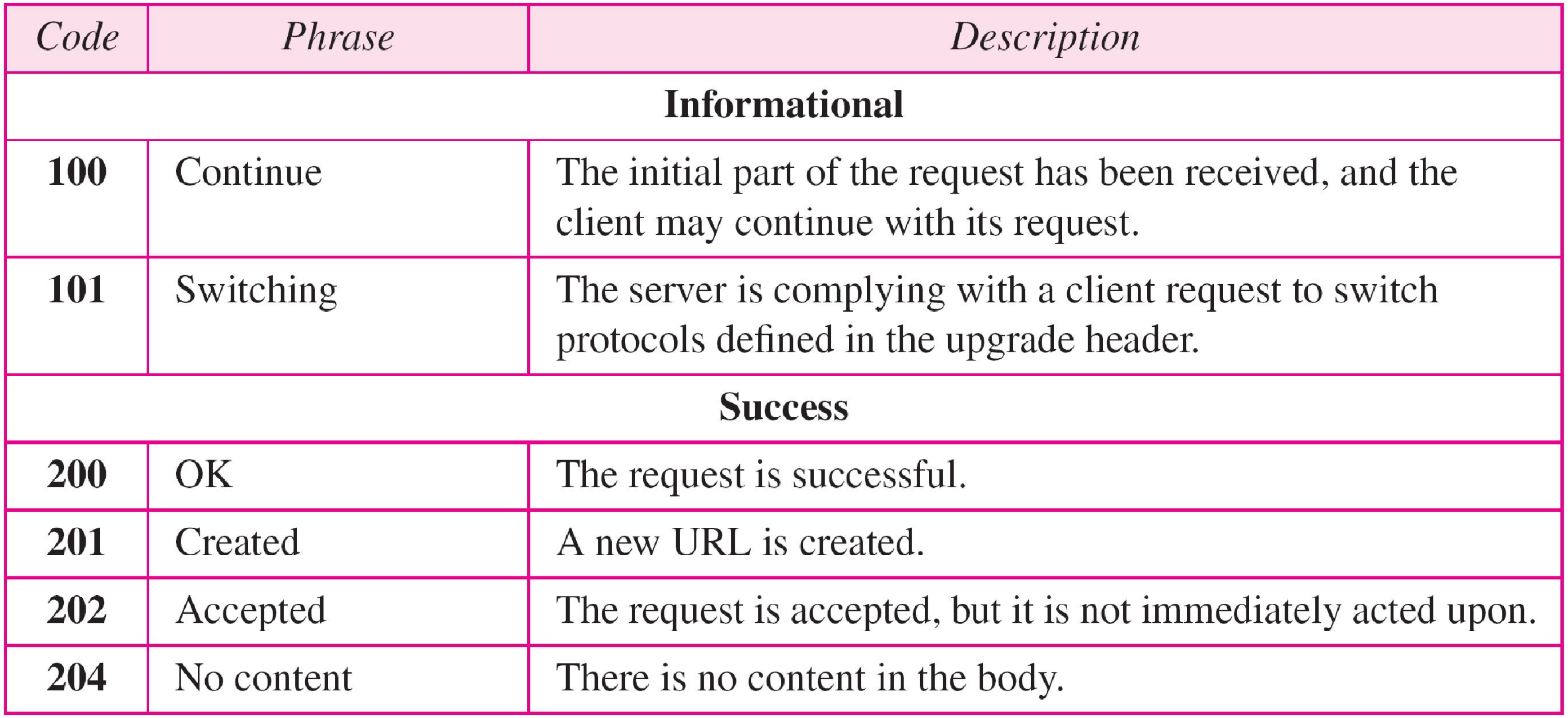


Table http.1 *Methods*



Method	*Action*
GET	Requests a document from the server
HEAD	Requests information about a document but not the document itself
POST	Sends some information from the client to the server
PUT	Sends a document from the server to the client
TRACE	Echoes the incoming request
CONNECT	Reserved
OPTION	Inquires about available options

Table http.2 *Status codes*

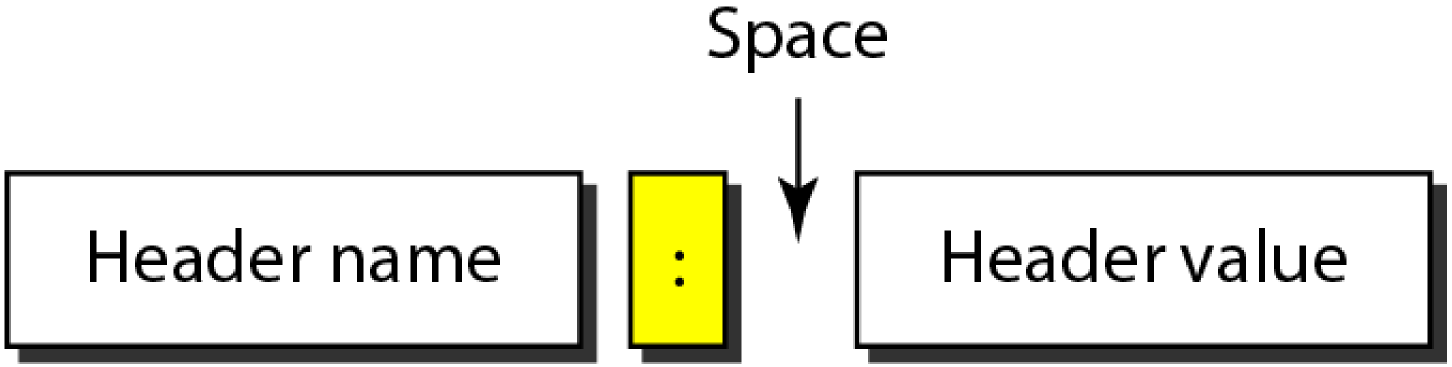


Code	*Phrase*	*Description*
Informational		
100	Continue	The initial part of the request has been received, and the client may continue with its request.
101	Switching	The server is complying with a client request to switch protocols defined in the upgrade header.
Success		
200	OK	The request is successful.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

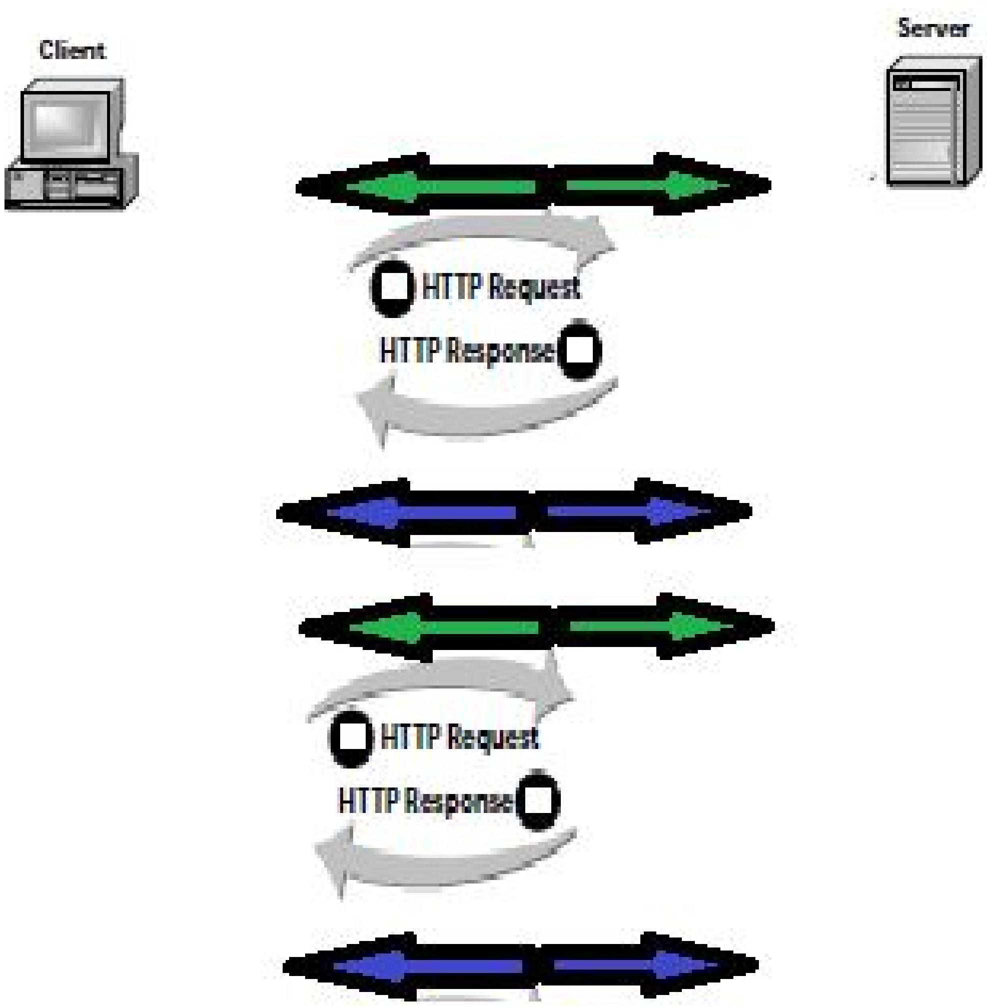
Table http.2 *Status codes (continued)*

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

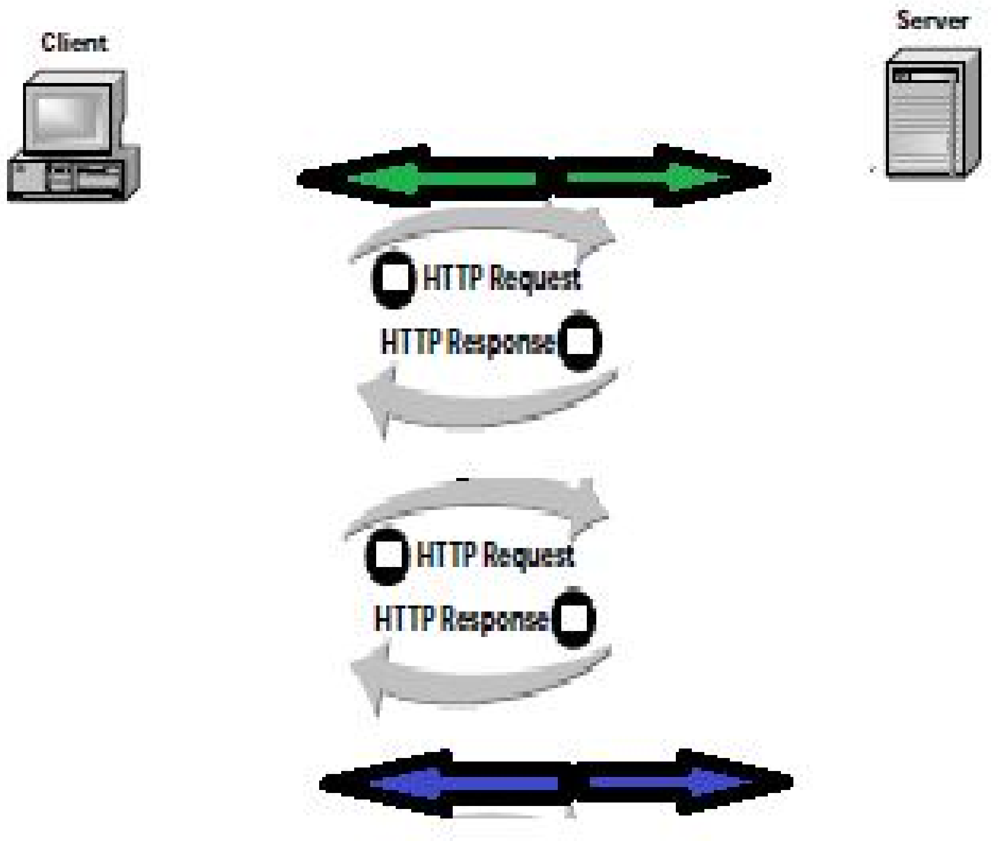
Figure http.15 *Header format*



HTTP1.0 Non-persistence

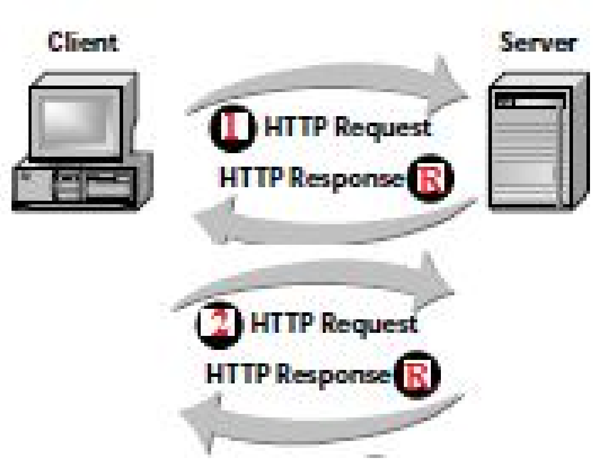


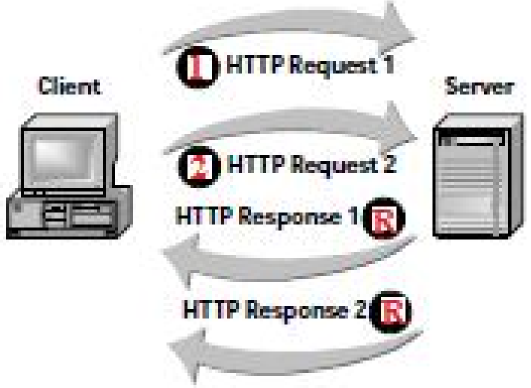
HTTP1.1 Persistence



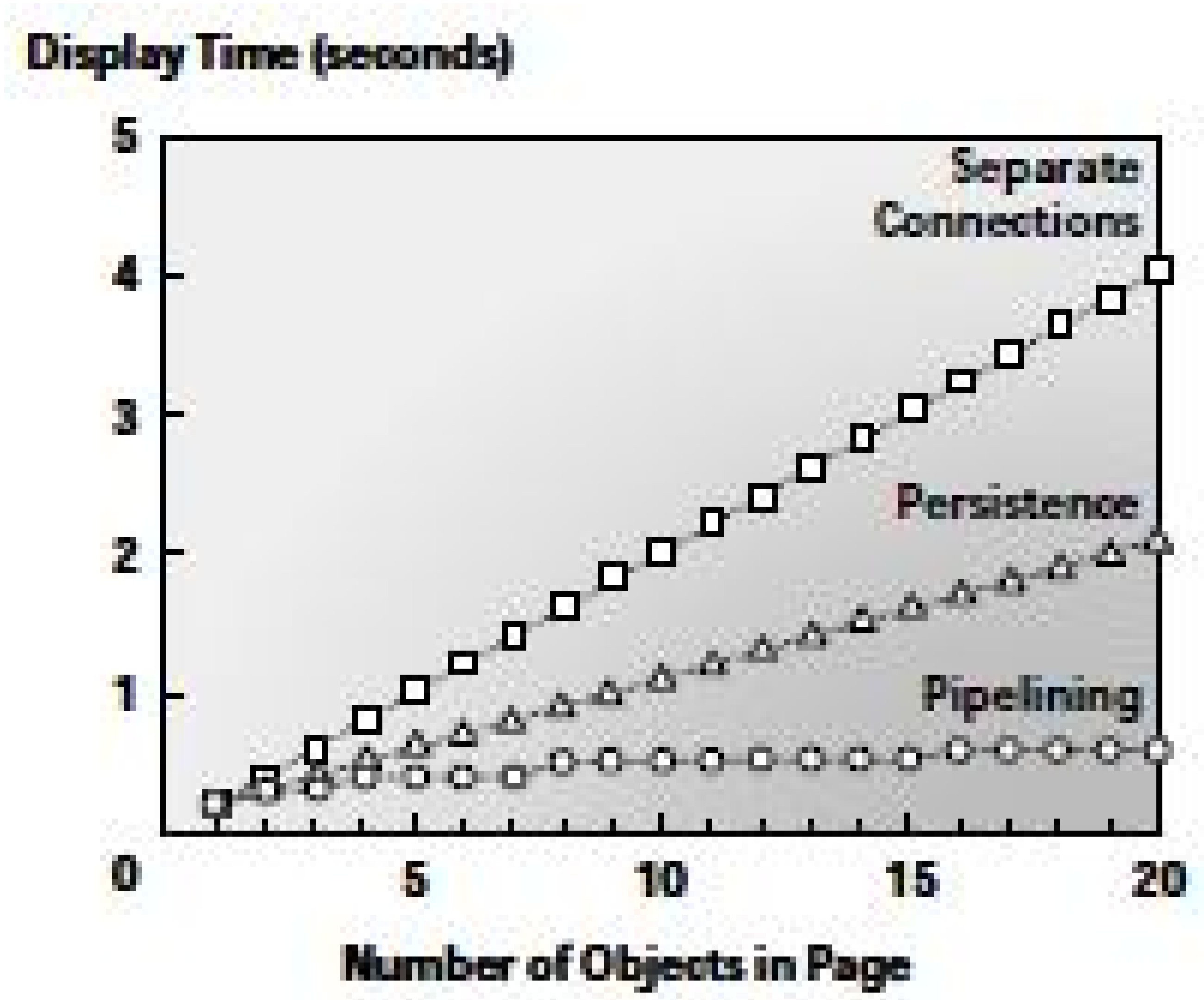
Pipelining

- Client does not have to wait for a response to one request before issuing a new request on the connection. It can follow the first request immediately with a second request.



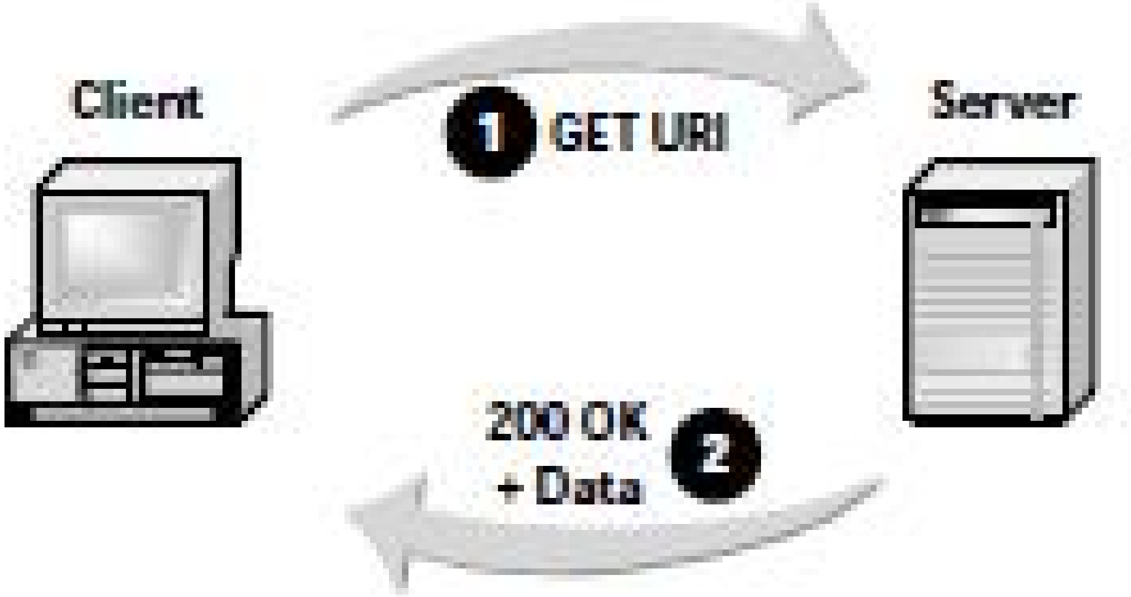


Impact of Persistence and Pipelining



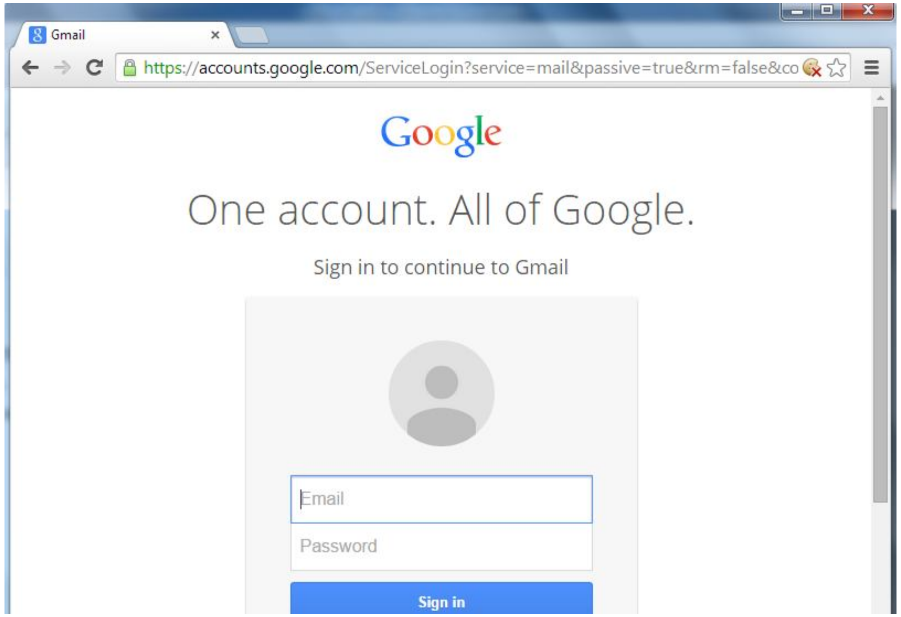
User Operations

- Web Page Retrieval – GET
  - A server responds to a GET request by returning the requested resource, often a new Web page. The new page is the data in the response.

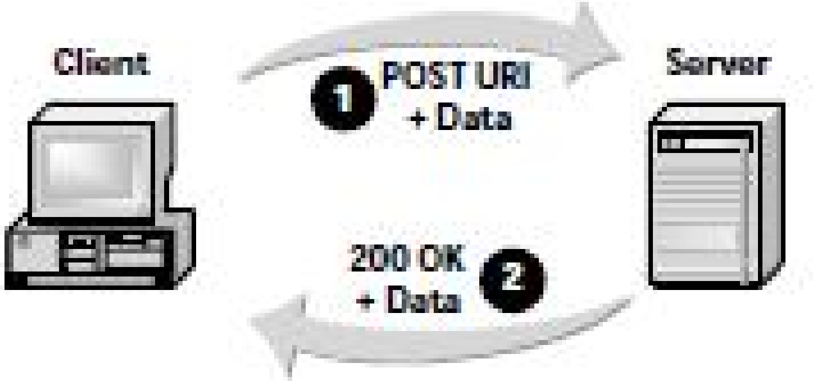


User Operations

- Web Forms – POST
  - A server responds to a POST request by returning new information such as search results. This information is carried as data in the response.

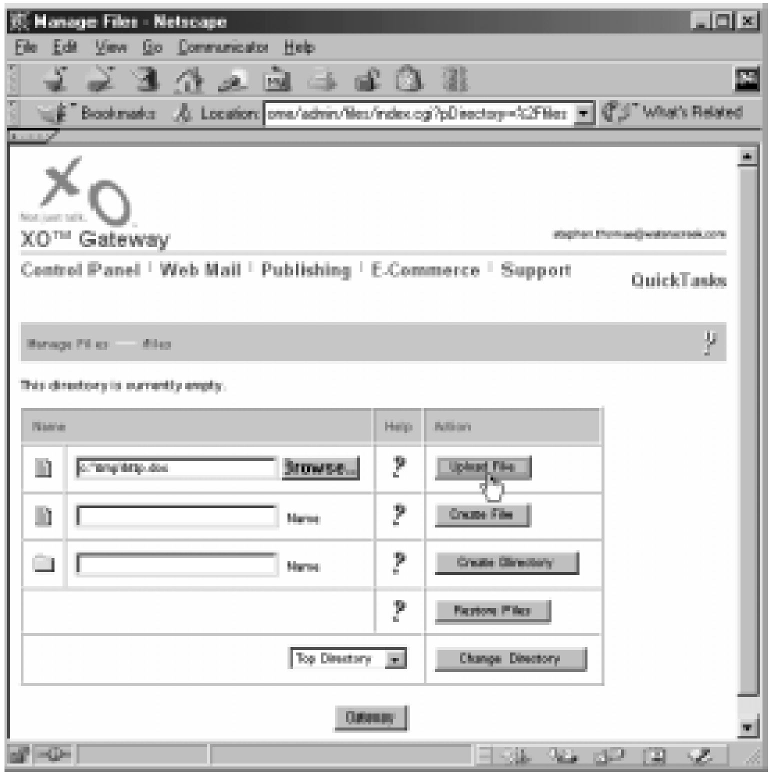


Gmail
https://accounts.google.com/ServiceLogin?service=mail&passive=true&rm=false&co
Google
One account. All of Google.
Sign in to continue to Gmail
Email
Password
Sign in



User Operations

- **File Upload – PUT**
- Clients can use the PUT request to send a new object to a server. The URI that's part of the request tells the server where to put the object.



Manage Files - Netscape

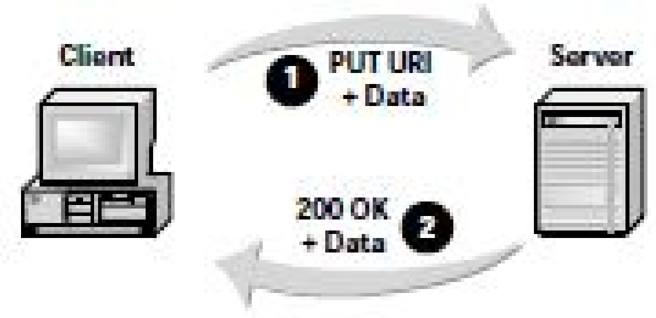
Location: oms/admin/files/index.cgi?pDirectory=%2Ffiles

XO™ Gateway

Manage Files — files

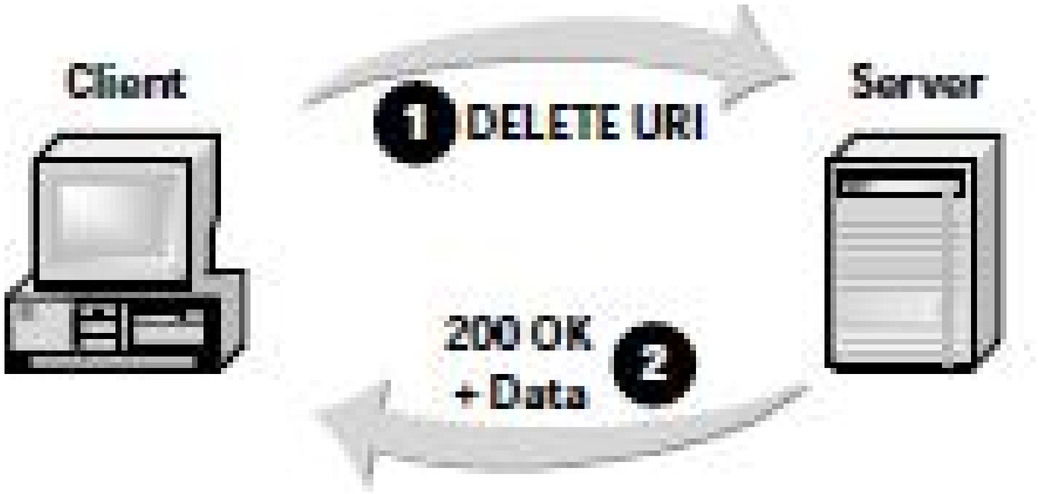
This directory is currently empty.

Name		Help	Action
	c:\tmp\ftp.doc	?	Upload File
	Name	?	Create File
	Name	?	Create Directory
		?	Restore Files
	Top Directory		Change Directory



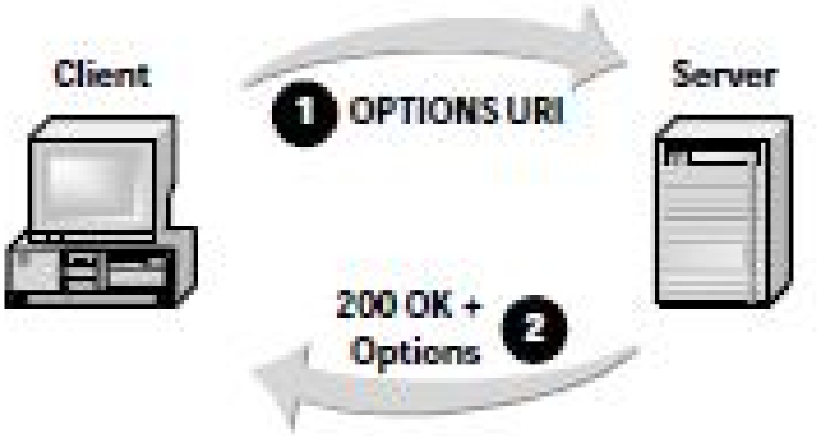
User Operations **File Deletion – DELETE**

- Client sends a **DELETE** message along with the *uri* of the object the server should remove. The server responds with a status code and, optionally, more data for the client.



User Operations Capabilities – OPTIONS

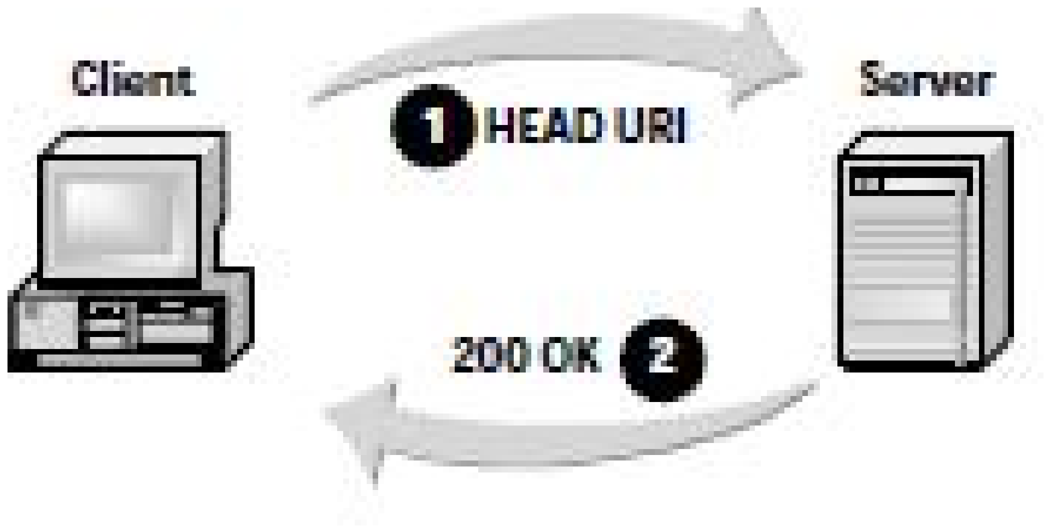
- Clients can use an OPTIONS message to discover what capabilities a server supports. If the client includes a uri, the server responds with the options relevant to that object.
- If the client sends an asterisk (*) as the uri, the server returns the general options that apply to all objects it maintains.
- A client might use it to determine the
  - http version or,
  - encoding methods (in the case of a specific uri).



User Operations

Status – HEAD

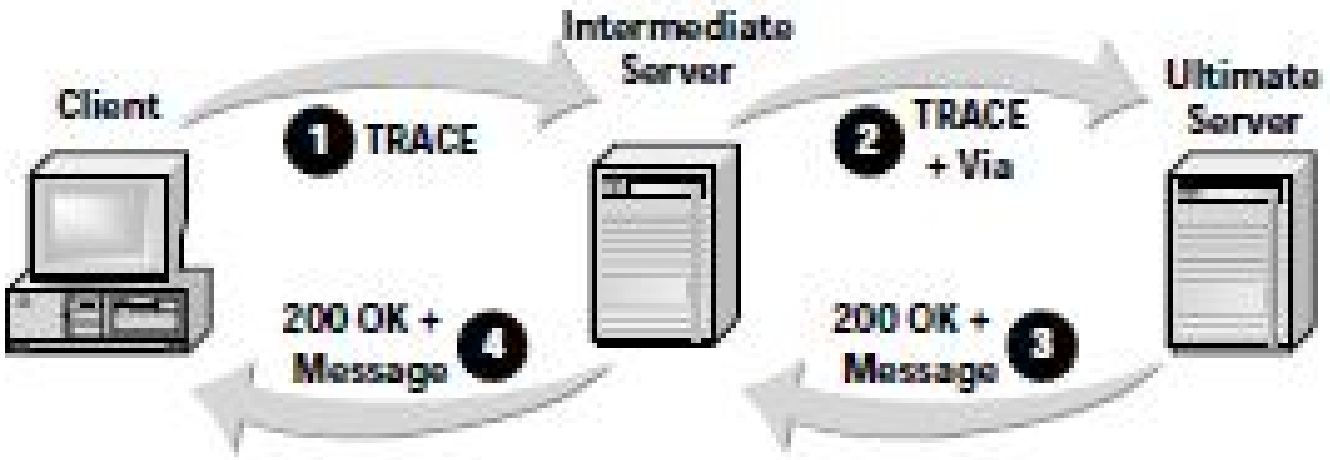
- The HEAD request mimics a GET operation, except that the server does not actually return the requested object, only HTTP headers.



User Operations

Path – TRACE

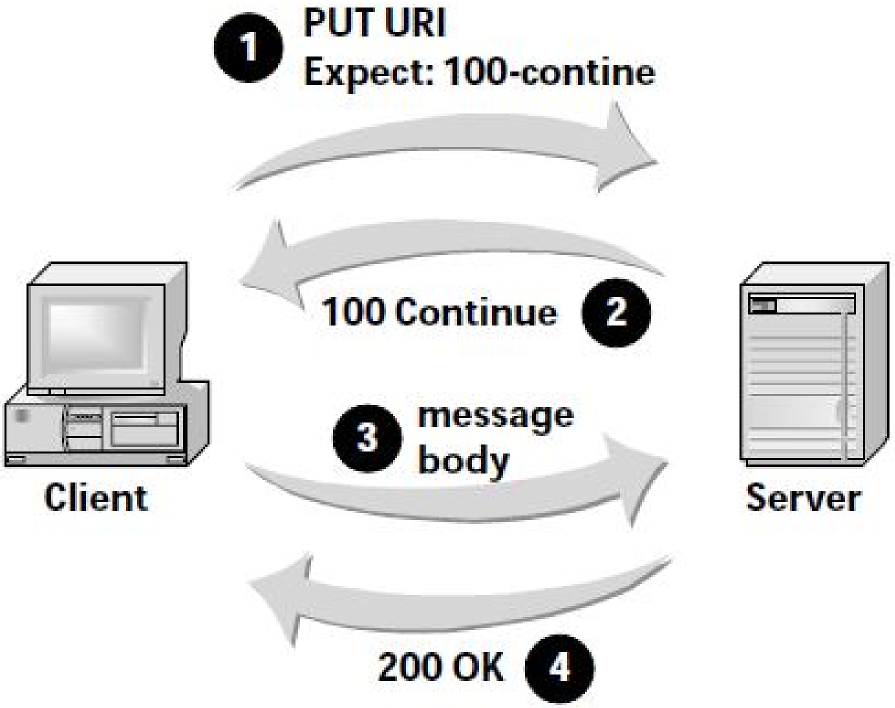
- The TRACE message gives clients a way to check the network path to a server.
- When a server receives a TRACE, it responds simply by copying the TRACE message itself into the data for the response.



100 Information

- Clients can ask a server to accept a request before they send the entire message body.

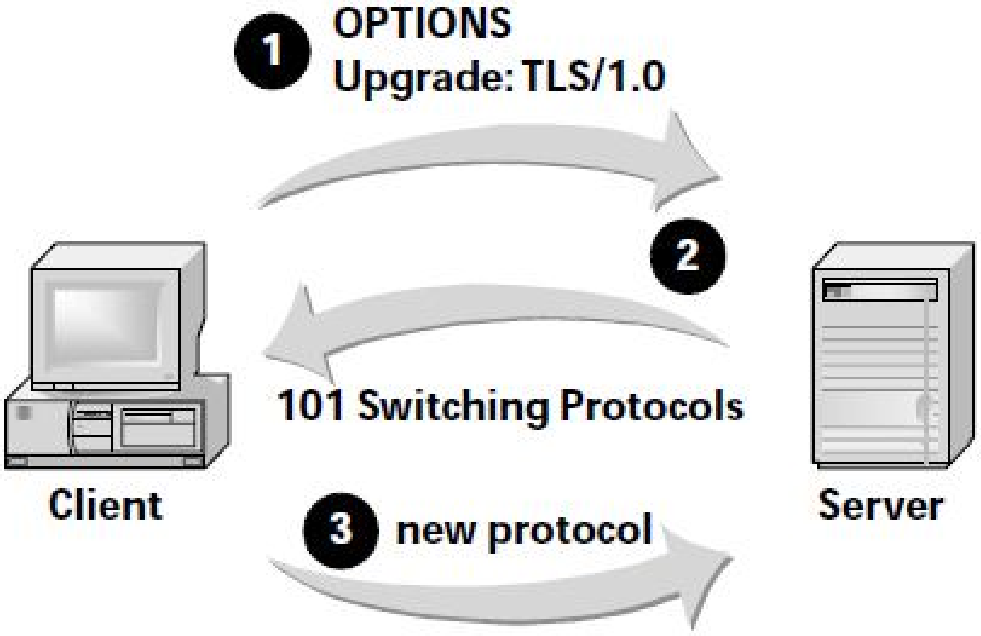
- The Expect header asks the server to signal its acceptance by returning a 100 status. Once the client receives a 100 status, it continues by sending the rest of the request.



101 Switching Protocol Information

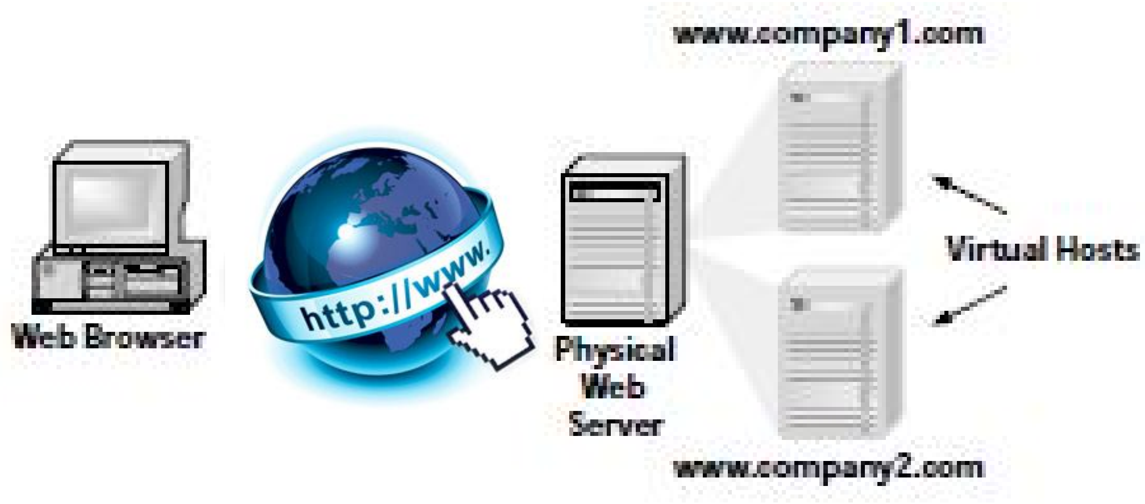
- Servers use the 101 Switching Protocols response to accept a client's request to upgrade protocols.

- The 101 status indicates that the sender is going to change protocols. The client should be using the new protocol as soon as it receives the 101 response.



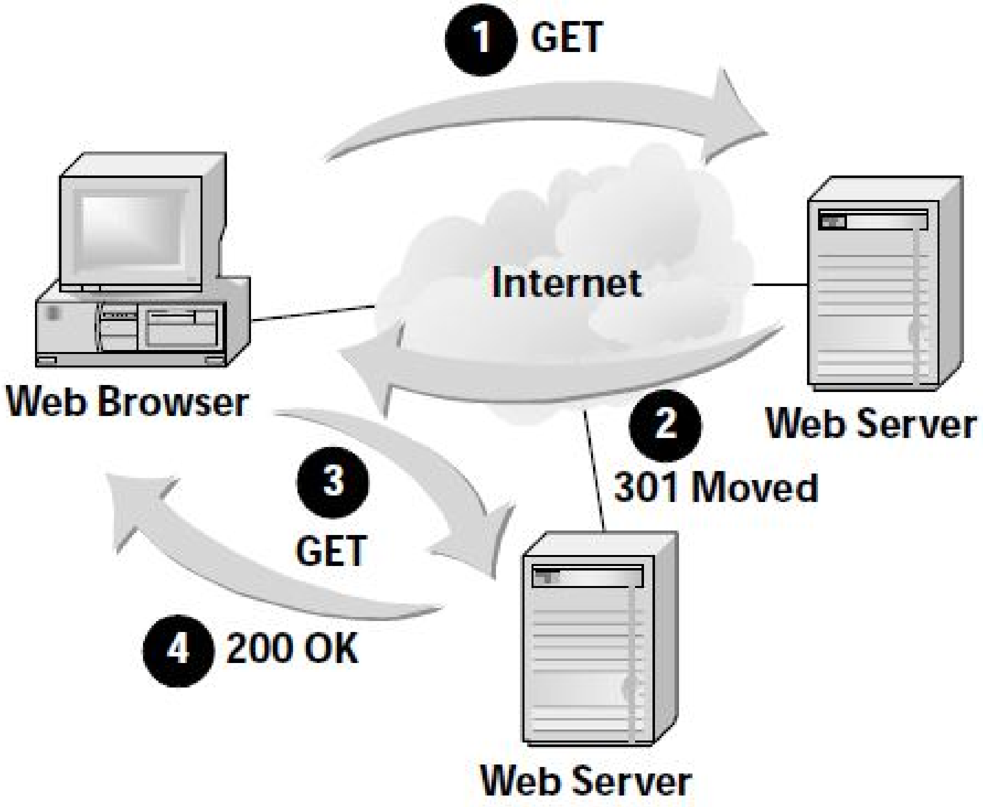
Virtual Hosts

- The designers of version 1.0 did not anticipate—Web hosting providers.
- 1.1 adds Virtual host support



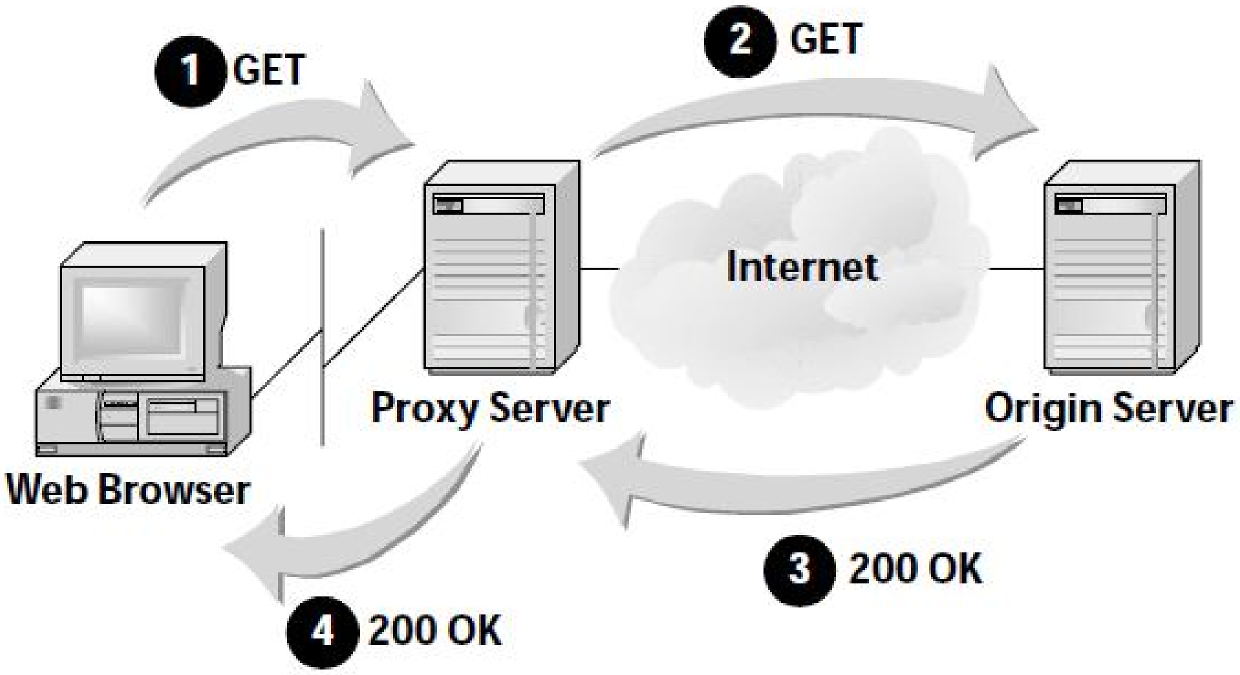
Redirection

- Redirection offers a way to support a single site to use multiple servers. Redirection lets a server redirect a client to another uri for an object.



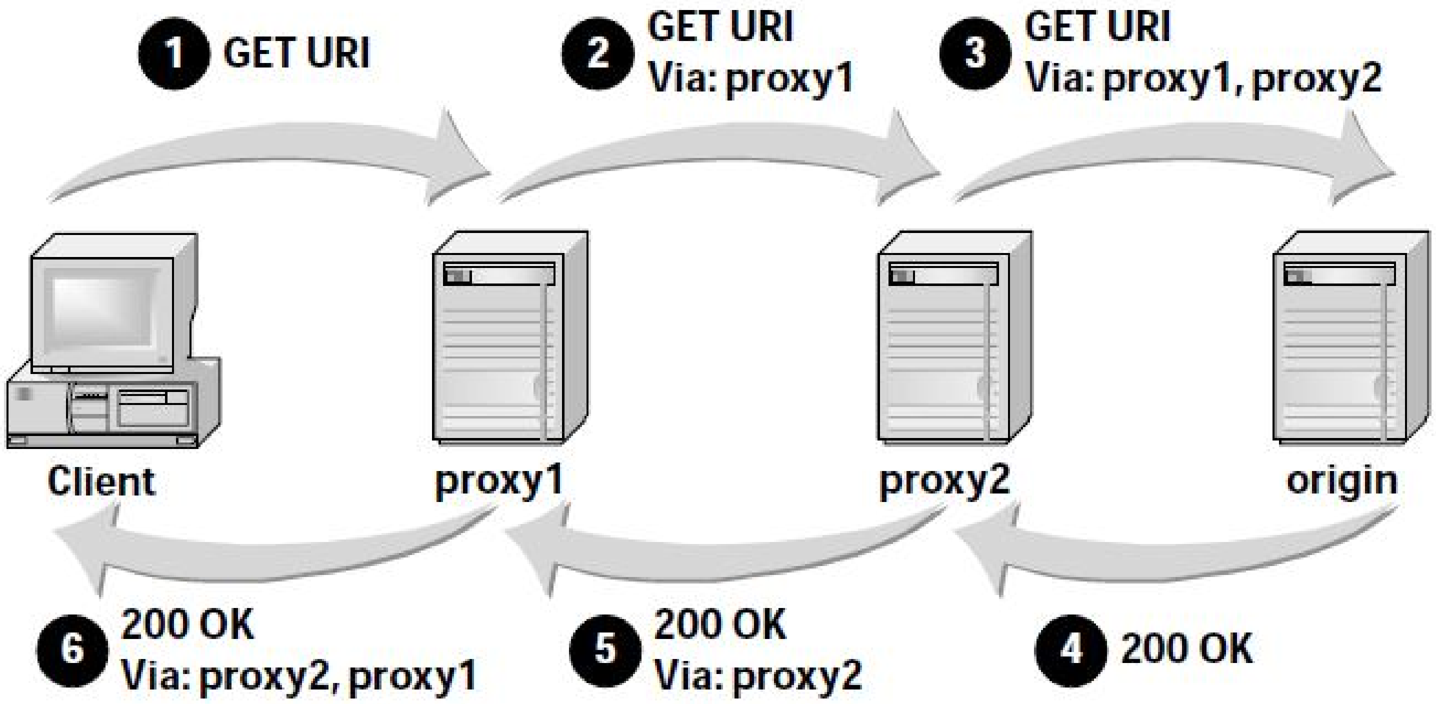
Proxies

- The server that the client first contacts relays the request to a new server and then relays the second server's response back to the client.
- Enforcing policy for an organization to filter that Internet access



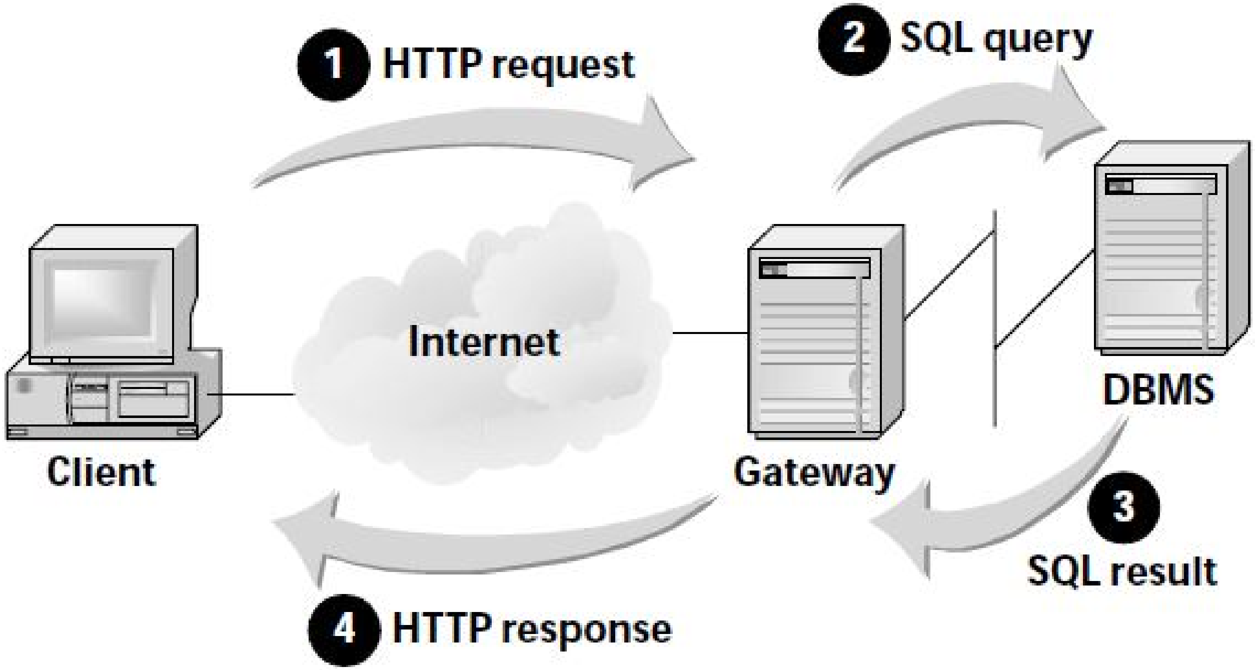
Proxy Chain

- As each server adds its own identity to the Via header in the request, which captures the path taken by the request through the server chain. The response follows the same process, with each proxy inserting its identity in the *Via* header.
- Proxy servers perform several important functions for http communications. The most common is in support of caching



Gateways

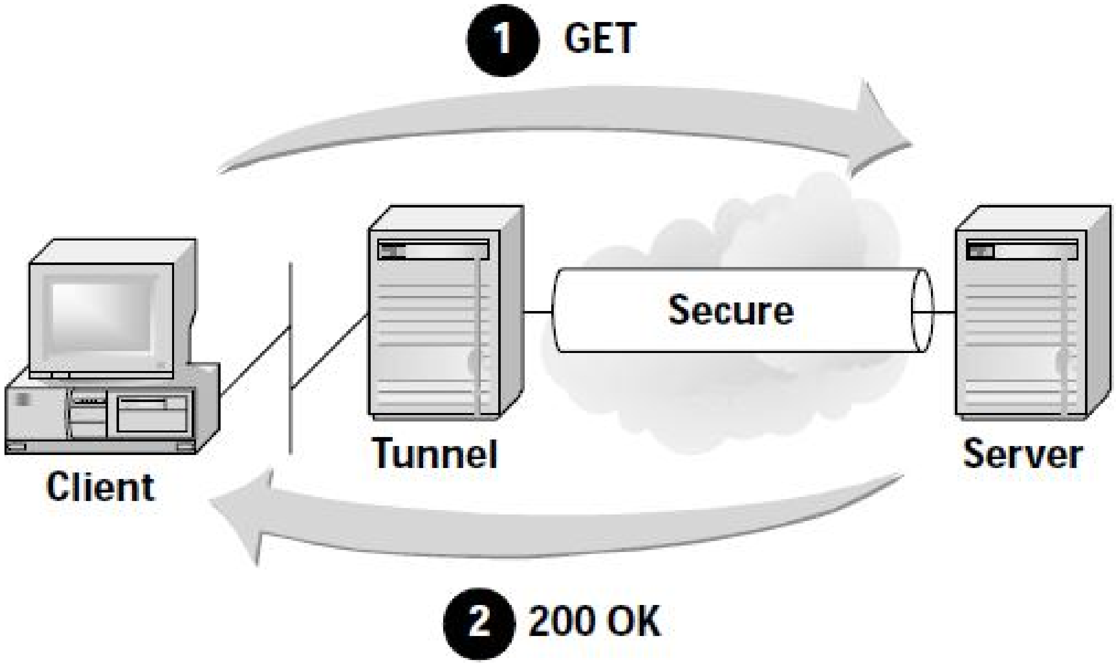
- Gateways act as an endpoint to a server chain, but they still rely on other servers to provide all or part of the requested object.
- Gateways may use a protocol other than http to access the object.



Tunnels

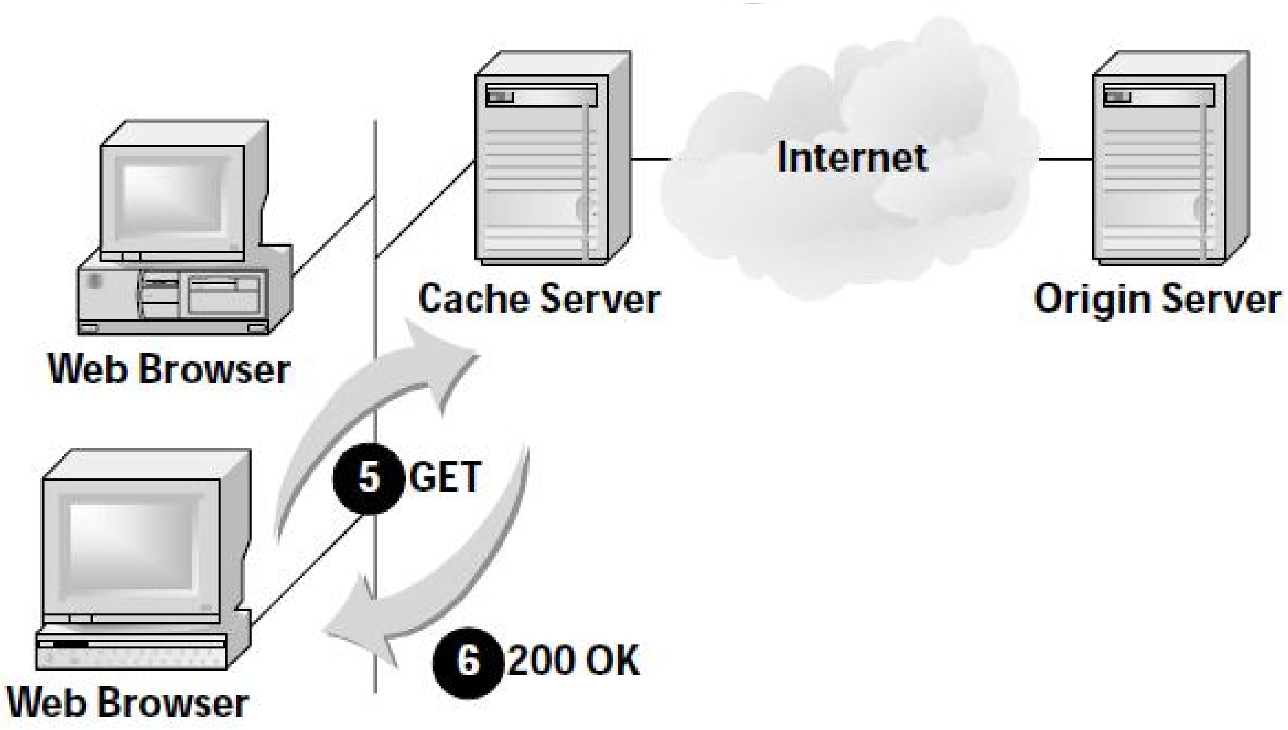
- A tunnel allows a client to communicate directly with a distant server. Tunnel creates a secure path for the client’s request and the server’s response.

- Tunnels are relatively transparent to the original client



Cache Servers

- Cache servers are proxy servers that relay requests and responses. In addition, they keep a local copy of any responses they receive.

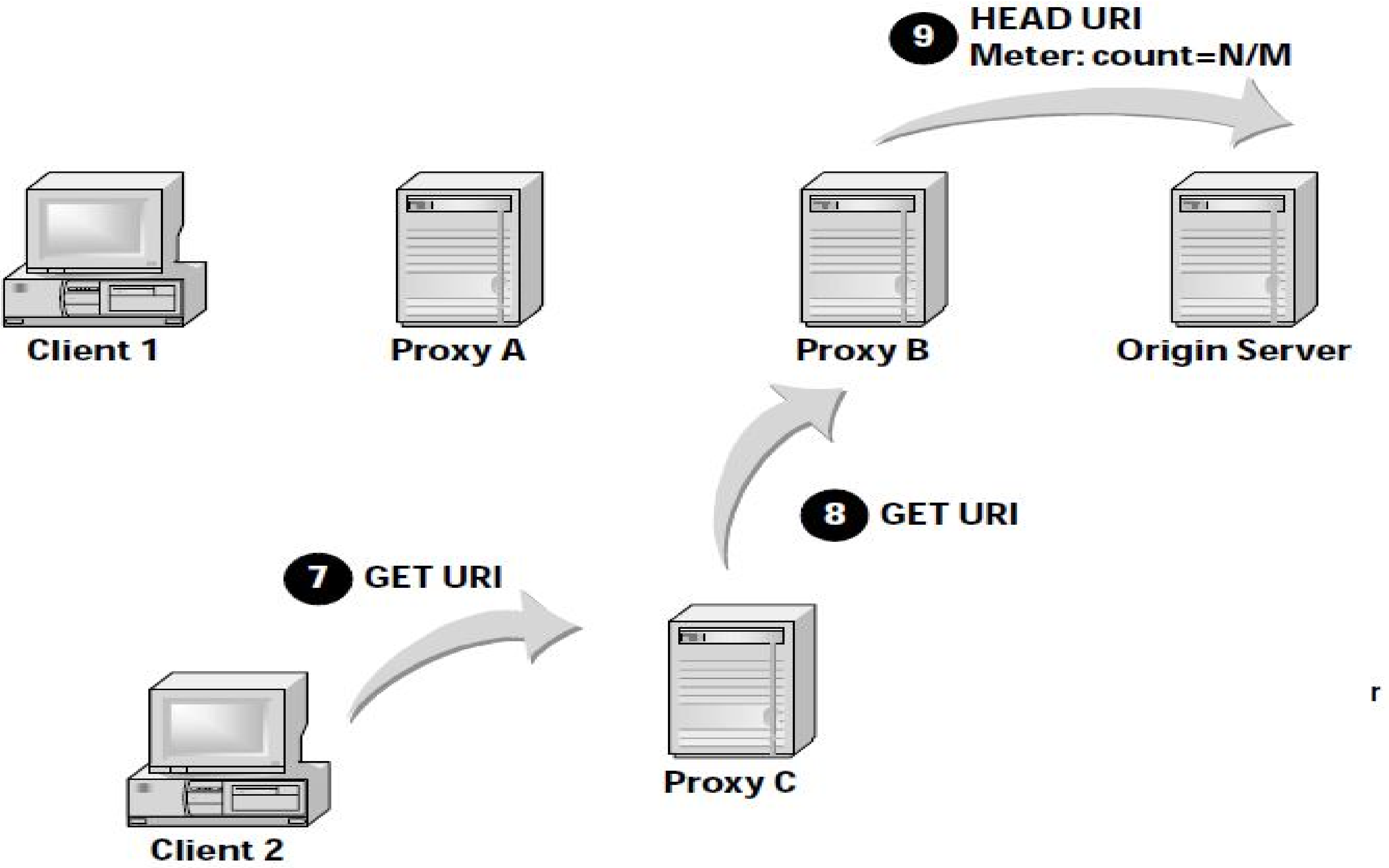


Advantages

- Reduces the load on origin servers
- Improves Performance

- Disadvantages
  - Cache deliver pages remains transparent to origin server
  - Site derives revenue from advertising

Counting and Limiting Page Views

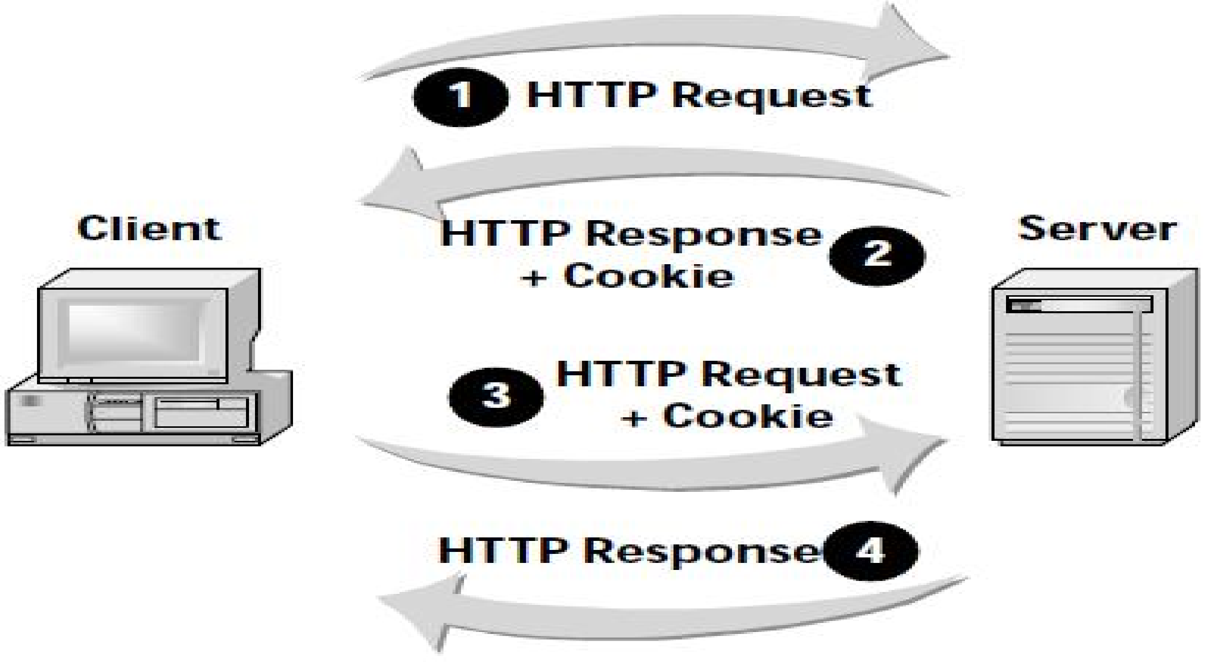


Cookies and State Maintenance

- http normally operates as if each client request is independent of all others.
- *Stateless*
  - Because maintaining state requires server resources (memory, processing power, etc.)
- *State full is also required*

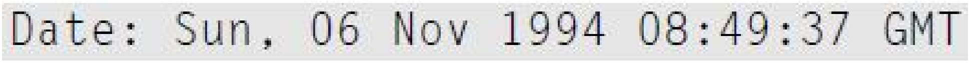
Cookies

- State maintenance requires one critical capability
- The mechanism that http defines for state maintenance is known as a *cookie.*
- *A server creates cookies to* track the state of a client, and it returns it to the client in its response. Once the client receives a cookie, it can include the cookie in subsequent requests to the same server



Date

The Date header indicates the time that the system sending a message originally generated that message.



```
Date: Sun, 06 Nov 1994 08:49:37 GMT
```

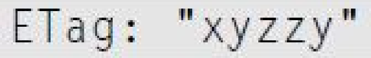
The Last-Modified header provides the time of the resource.

ETag

- Identify resources

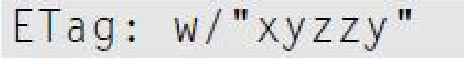
- Origin servers can assign an Etag

- Strong Etag and



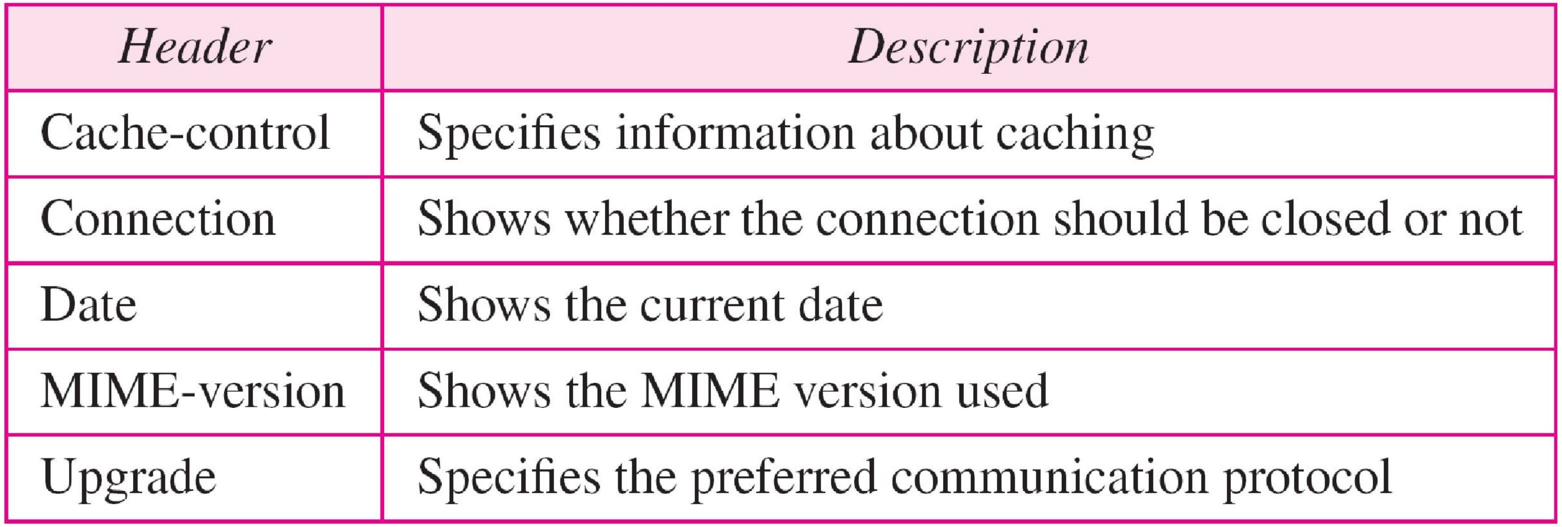
```
ETag: "xyzzy"
```

- Weak Etag



```
ETag: w/"xyzzy"
```

Table http.3 *General headers*



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

Table http.4 ***Request headers***

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

Table http.5 *Response headers*

Header	*Description*
Accept-range	Shows if server accepts the range requested by client
Age	Shows the age of the document
Public	Shows the supported list of methods
Retry-after	Specifies the date after which the server is available
Server	Shows the server name and version number

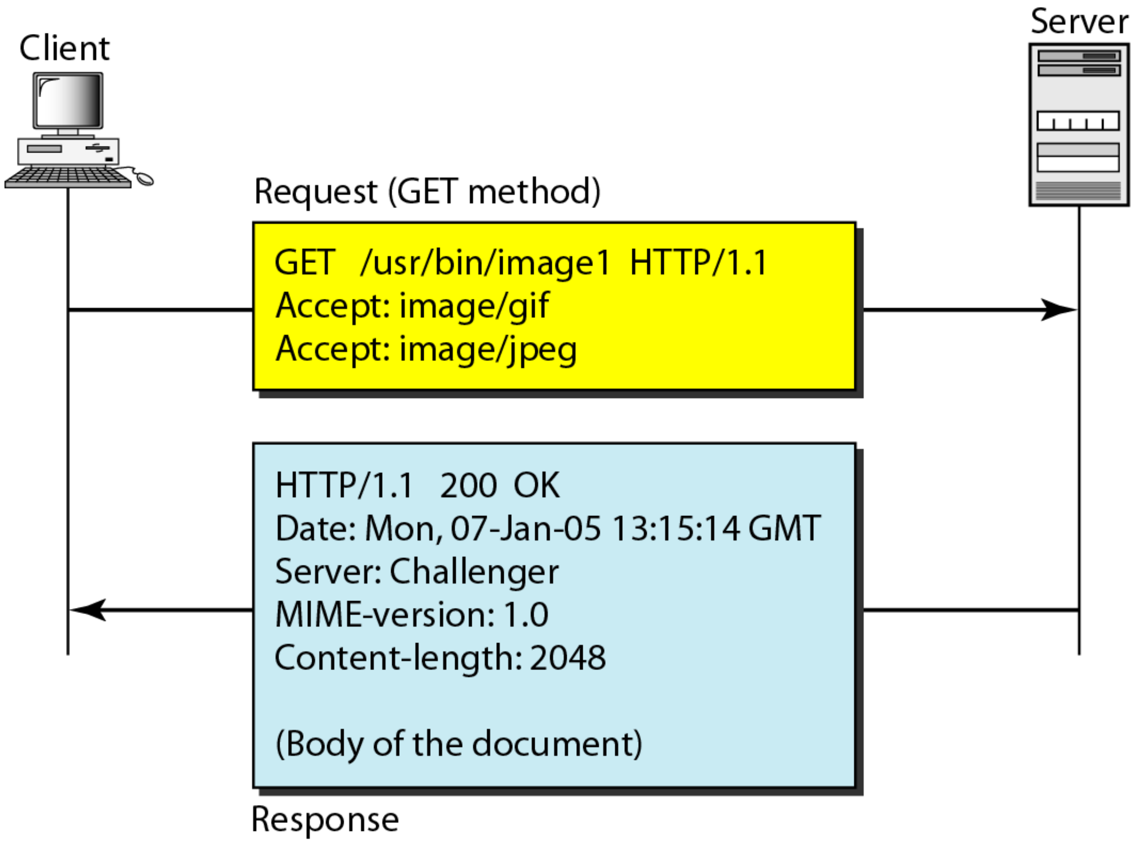
Table http.6 *Entity headers*

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

Example http.1

This example retrieves a document. We use the GET method to retrieve an image with the path /usr/bin/image1. The request line shows the method (GET), the URL, and the HTTP version (1.1). The header has two lines that show that the client can accept images in the GIF or JPEG format. The request does not have a body. The response message contains the status line and four lines of header. The header lines define the date, server, MIME version, and length of the document. The body of the document follows the header (see Figure http.16).

Figure http.16 *Example http.1*



```
GET   /usr/bin/image1  HTTP/1.1
Accept: image/gif
Accept: image/jpeg
```

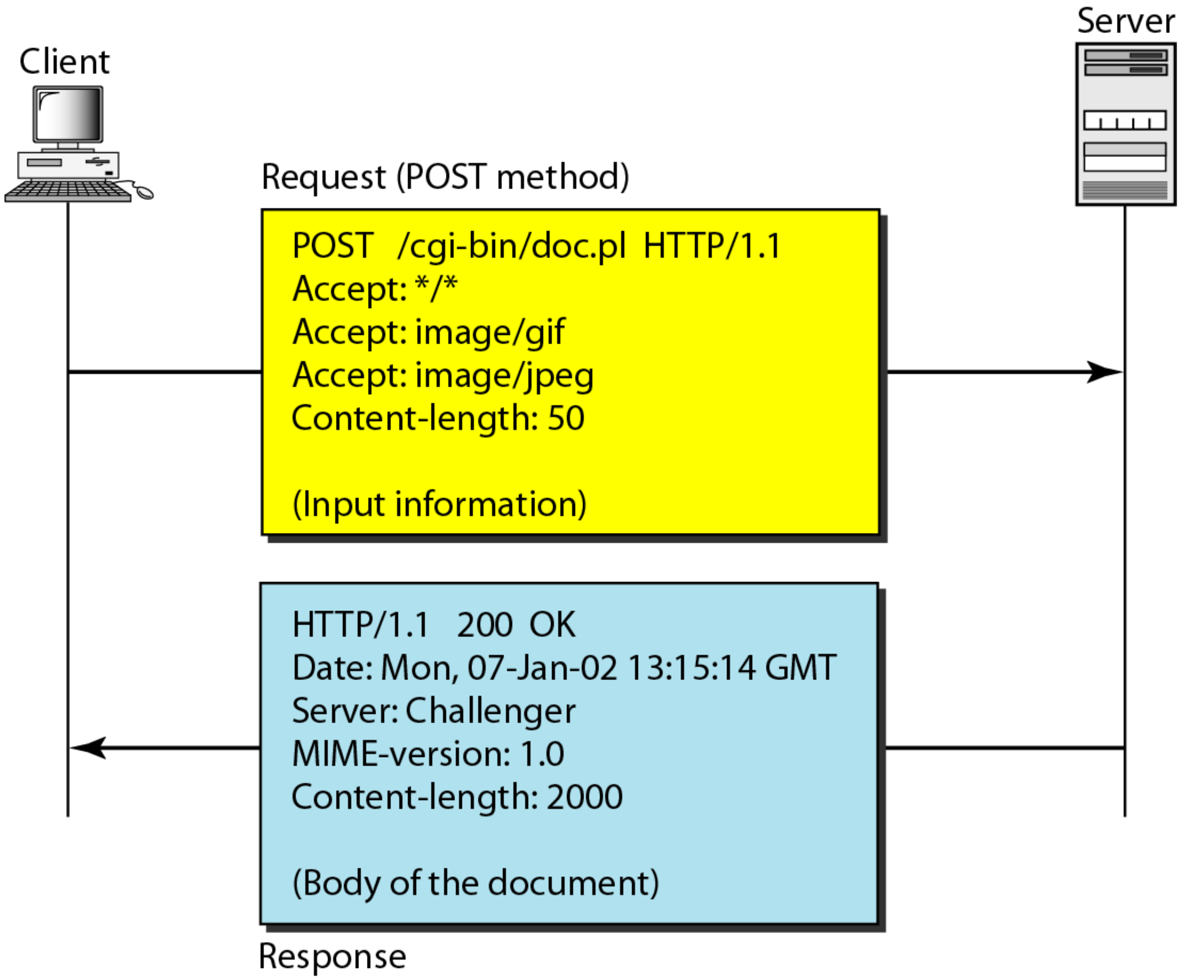
```
HTTP/1.1   200  OK
Date: Mon, 07-Jan-05 13:15:14 GMT
Server: Challenger
MIME-version: 1.0
Content-length: 2048

(Body of the document)
```

Example http.2

In this example, the client wants to send data to the server. We use the POST method. The request line shows the method (POST), URL, and HTTP version (1.1). There are four lines of headers. The request body contains the input information. The response message contains the status line and four lines of headers. The created document, which is a CGI document, is included as the body .

Figure http.17 *Example http.2*



```
POST   /cgi-bin/doc.pl  HTTP/1.1
Accept: */*
Accept: image/gif
Accept: image/jpeg
Content-length: 50

(Input information)
```

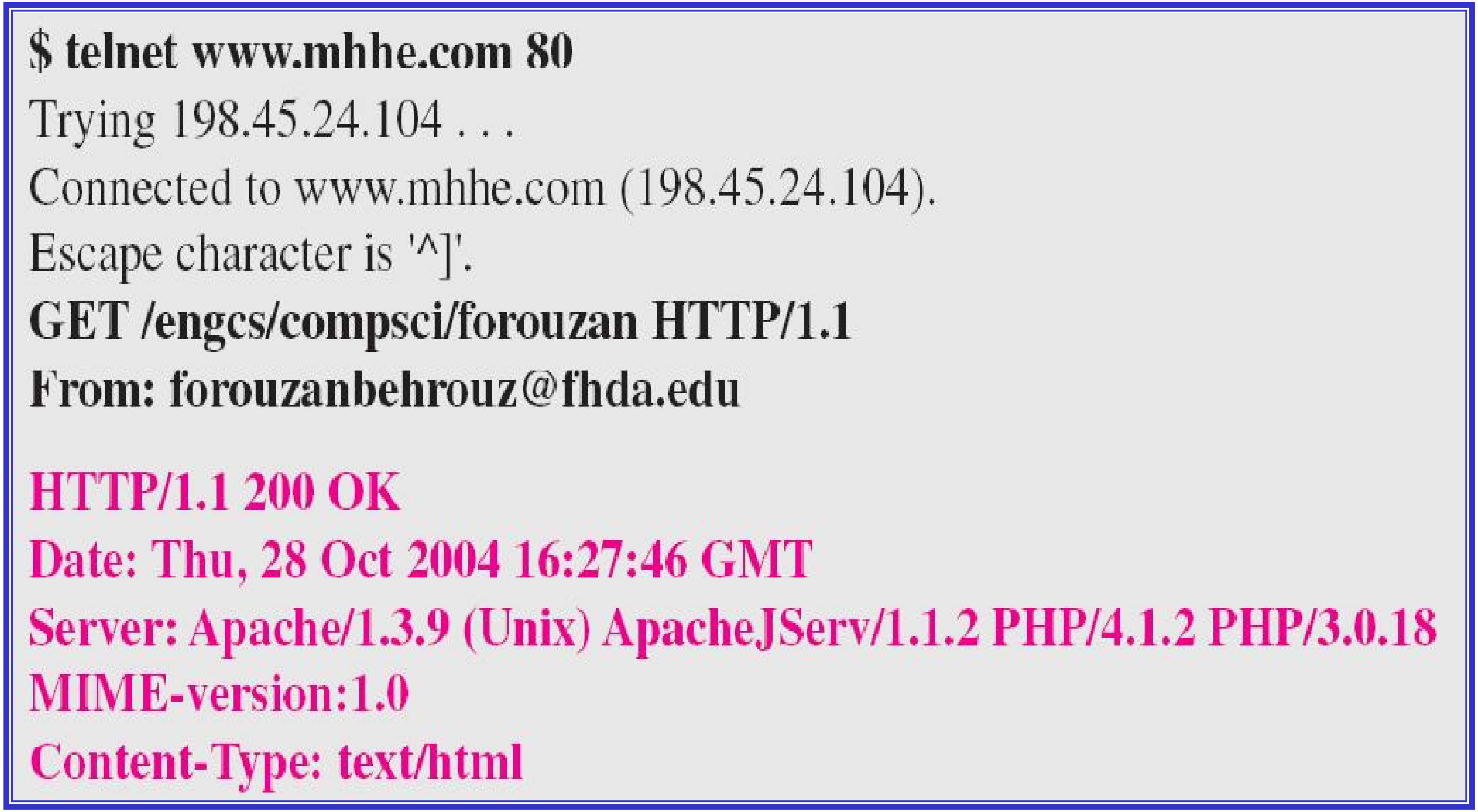
```
HTTP/1.1   200  OK
Date: Mon, 07-Jan-02 13:15:14 GMT
Server: Challenger
MIME-version: 1.0
Content-length: 2000

(Body of the document)
```

Example http.3

HTTP uses ASCII characters. A client can directly connect to a server using TELNET, which logs into port 80. We then type three lines. The first shows the request line (GET method), the second is the header (defining the host), the third is a blank, terminating the request. The server response is seven lines starting with the status line. The blank line at the end terminates the server response. The file of 14,230 lines is received after the blank line (not shown here). The last line is the output by the client.

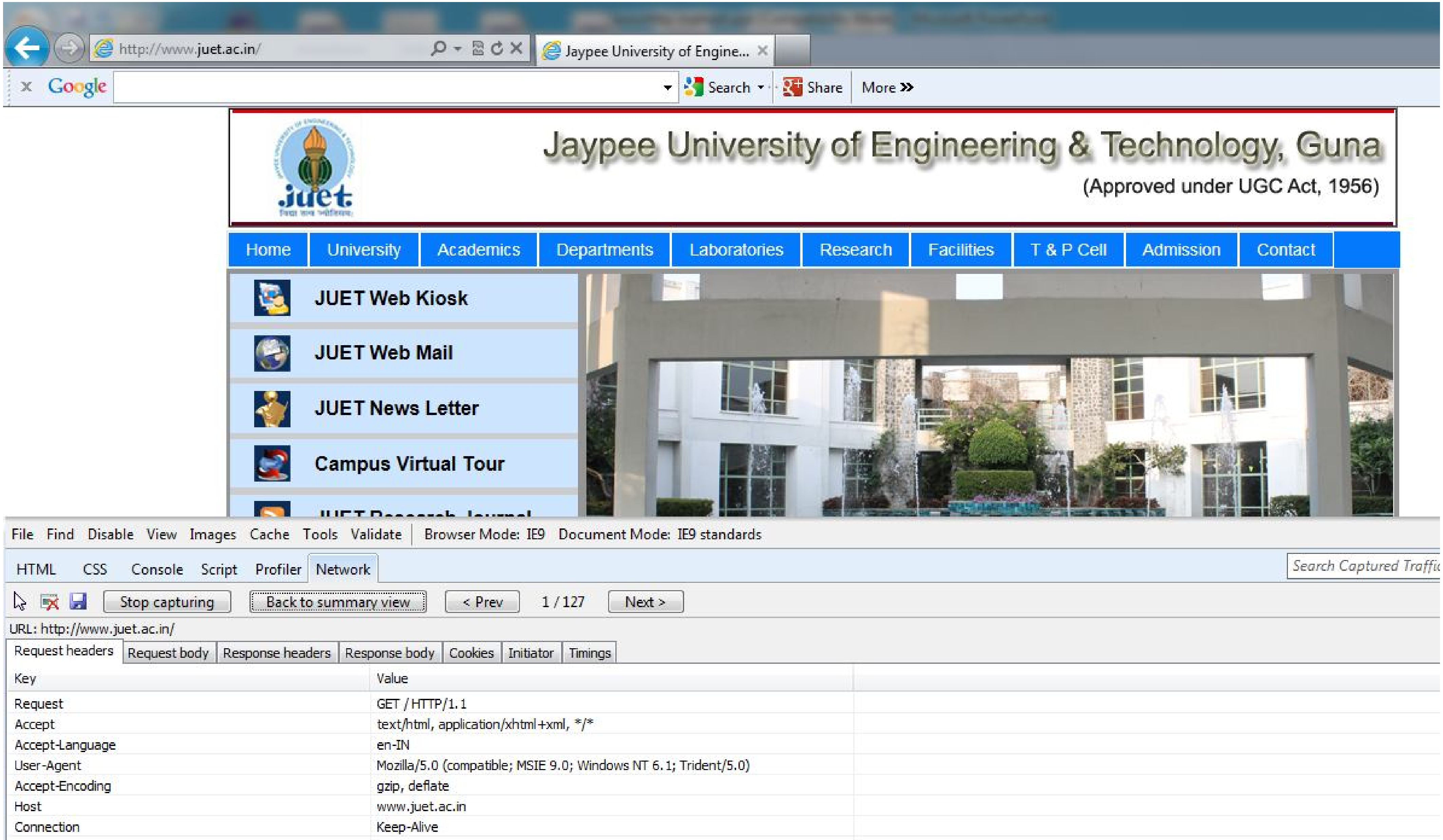
Example http.3 (continued)



```
$ telnet www.mhhe.com 80
Trying 198.45.24.104 . . .
Connected to www.mhhe.com (198.45.24.104).
Escape character is '^]'.
GET /engcs/compsci/forouzan HTTP/1.1
From: forouzanbehrouz@fhda.edu

HTTP/1.1 200 OK
Date: Thu, 28 Oct 2004 16:27:46 GMT
Server: Apache/1.3.9 (Unix) ApacheJServ/1.1.2 PHP/4.1.2 PHP/3.0.18
MIME-version:1.0
Content-Type: text/html
```

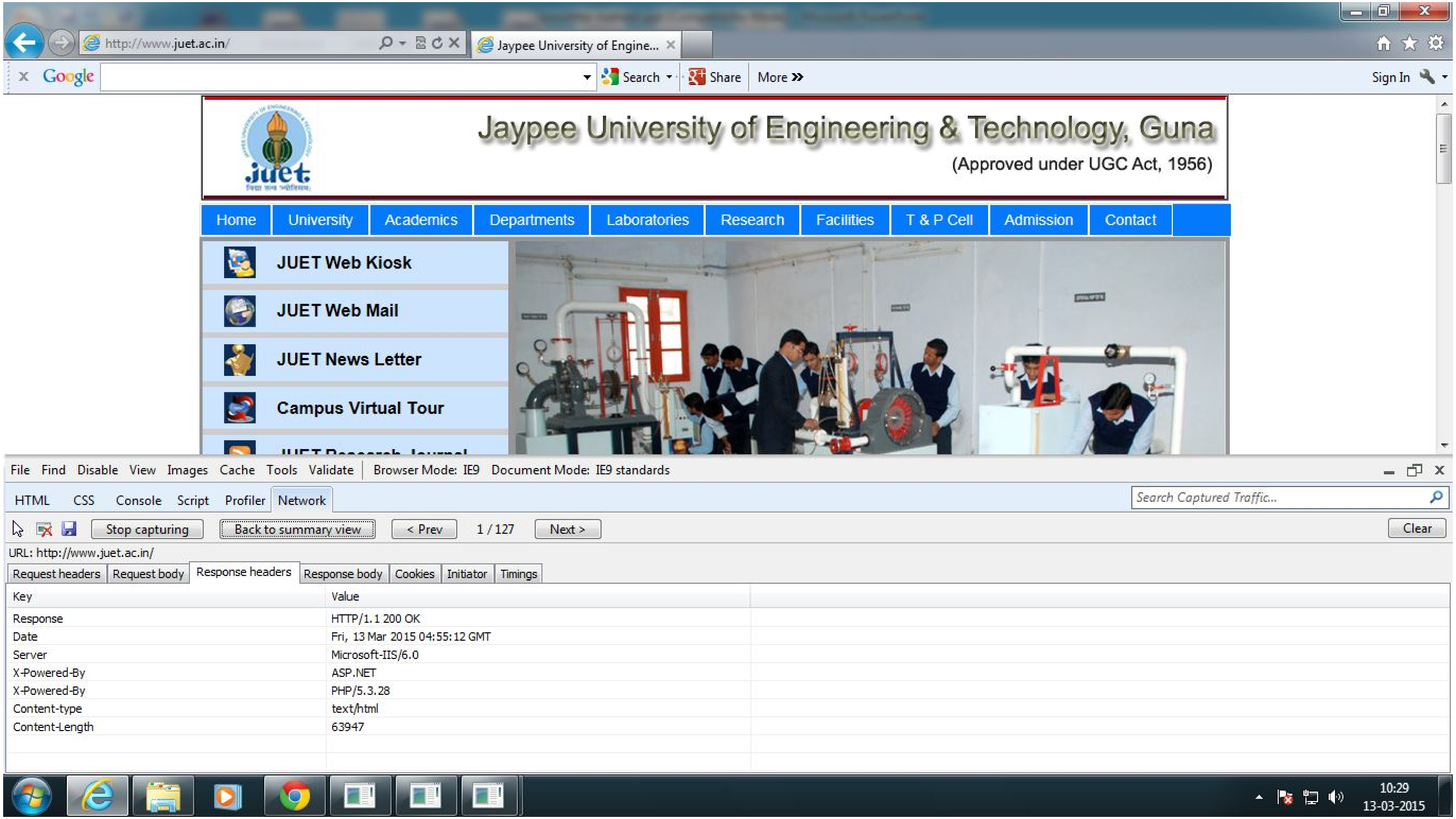
Request Headers



URL: http://www.juet.ac.in/

Key	Value
Request	GET / HTTP/1.1
Accept	text/html, application/xhtml+xml, */*
Accept-Language	en-IN
User-Agent	Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)
Accept-Encoding	gzip, deflate
Host	www.juet.ac.in
Connection	Keep-Alive

Response Headers



URL: http://www.juet.ac.in/

Key	Value
Response	HTTP/1.1 200 OK
Date	Fri, 13 Mar 2015 04:55:12 GMT
Server	Microsoft-IIS/6.0
X-Powered-By	ASP.NET
X-Powered-By	PHP/5.3.28
Content-type	text/html
Content-Length	63947