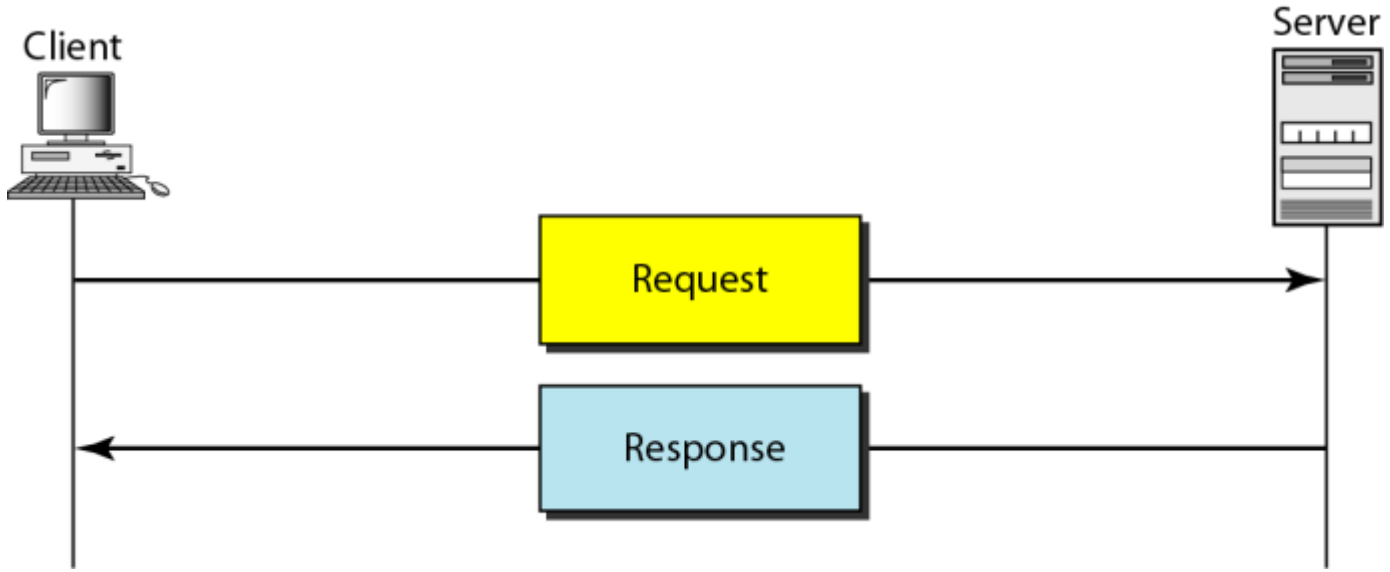
Chapter-4
Hypertext Transfer Protocol

HTTP

Web browsers interact with web servers with a simple application-level protocol called HTTP (Hypertext Transfer Protocol), which runs on top of TCP/IP network connections.

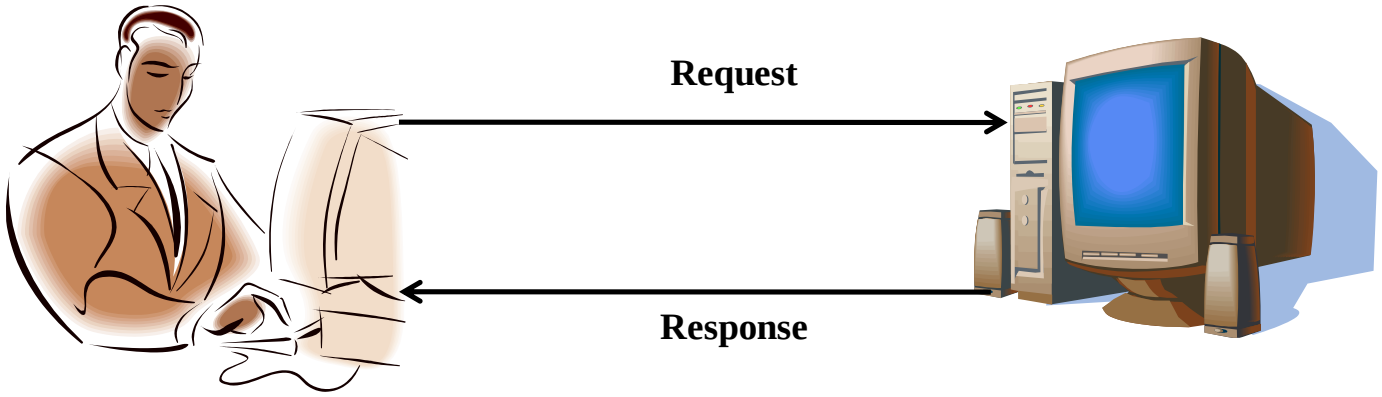
HTTP is a client-server protocol that defines how messages are formatted and transmitted, and what action web servers and browsers should take in response to various commands.

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



Characteristics of HTTP

- The **HTTP protocol uses the request/response paradigm**, that is, an HTTP client program sends an HTTP request message to an HTTP server, which returns an HTTP response message.



CLIENT PROGRAM
- **Running on end host**
- **Requests service**
- **E.g., Web browser**

SERVER PROGRAM
- **Running on end host**
- **Provides service**
- **E.g., Web server**

- **HTTP is a *pull protocol;*** the client *pulls i*nformation from the server (instead of server *pushing* information down to the client).

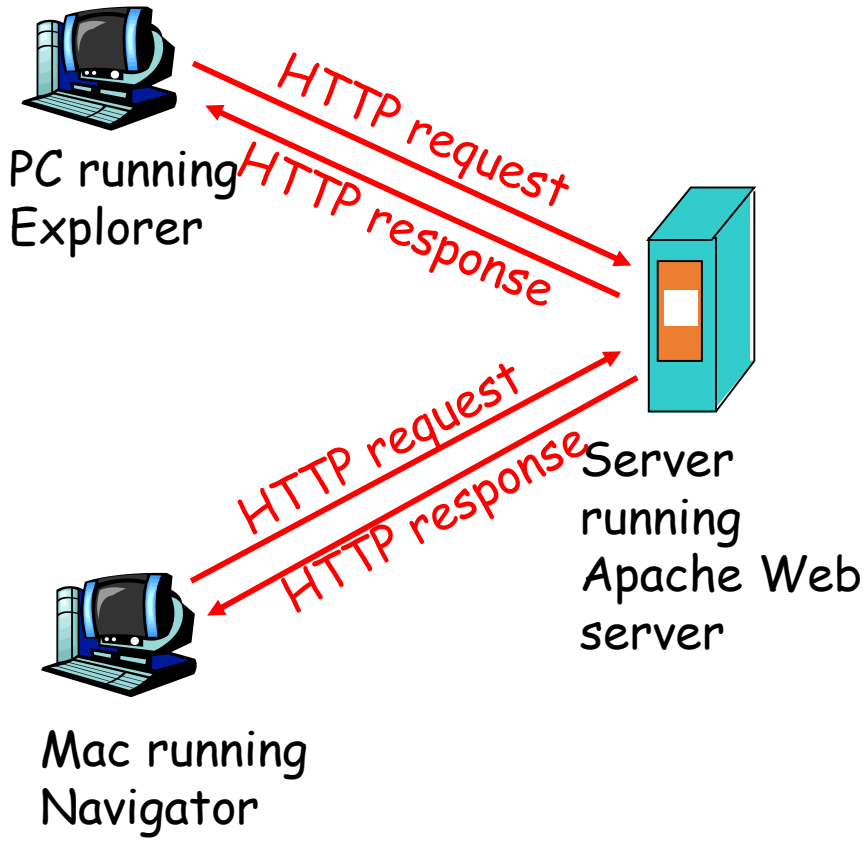
- **HTTP is a stateless protocol**, that is, each request-response exchange is treated independently. Clients and servers are not required to retain state.

- **HTTP is media independent:** Any type of data can be sent by HTTP as long as both the client and the server know how to handle the data content. It is required for the client as well as the server to specify the content type using appropriate MIME-type.

HTTP overview

HTTP: hypertext transfer protocol

- Web's application layer protocol
- client/server model
  - *client:* browser that requests, receives, "displays" Web objects
  - *server:* Web server sends objects in response to requests
- HTTP 1.0: RFC 1945
- HTTP 1.1: RFC 2068



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 version

HTTP uses a <major>.<minor> numbering scheme to indicate versions of the protocol. The version of an HTTP message is indicated by an HTTP-Version field in the first line.

Syntax:

HTTP-Version=“HTTP” “/” 1*DIGIT “.” 1*DIGIT

The initial version of HTTP was referred to as HTTP/0.9 which was a simple protocol for raw data transfer across the Internet. HTTP/1.0, as defined by RFC (Request for Comments), improved the protocol.

HTTP/1.1 was formally defined, and is currently an Internet Draft Standard [RFC-2616]. Essentially all operational browsers and servers support HTTP/1.1.

HTTP Connections

HTTP connection can either be persistent or non-persistent.

Non-persistent HTTP was used by HTTP/1.0.

The latest version HTTP/1.1 uses the persistent type of connection which is also known as kept-alive type connection as multiple messages or objects can be sent over a single TCP connection between client and server.

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 a single TCP connection between client and server.
- HTTP/1.1 uses persistent connections in default mode

Web and HTTP

First some jargon

- Web page consists of objects
- Object can be HTML file, JPEG image, Java applet, audio file,…
- Web page consists of base HTML-file which includes several referenced objects
- Each object is addressable by a URL
- Example URL:

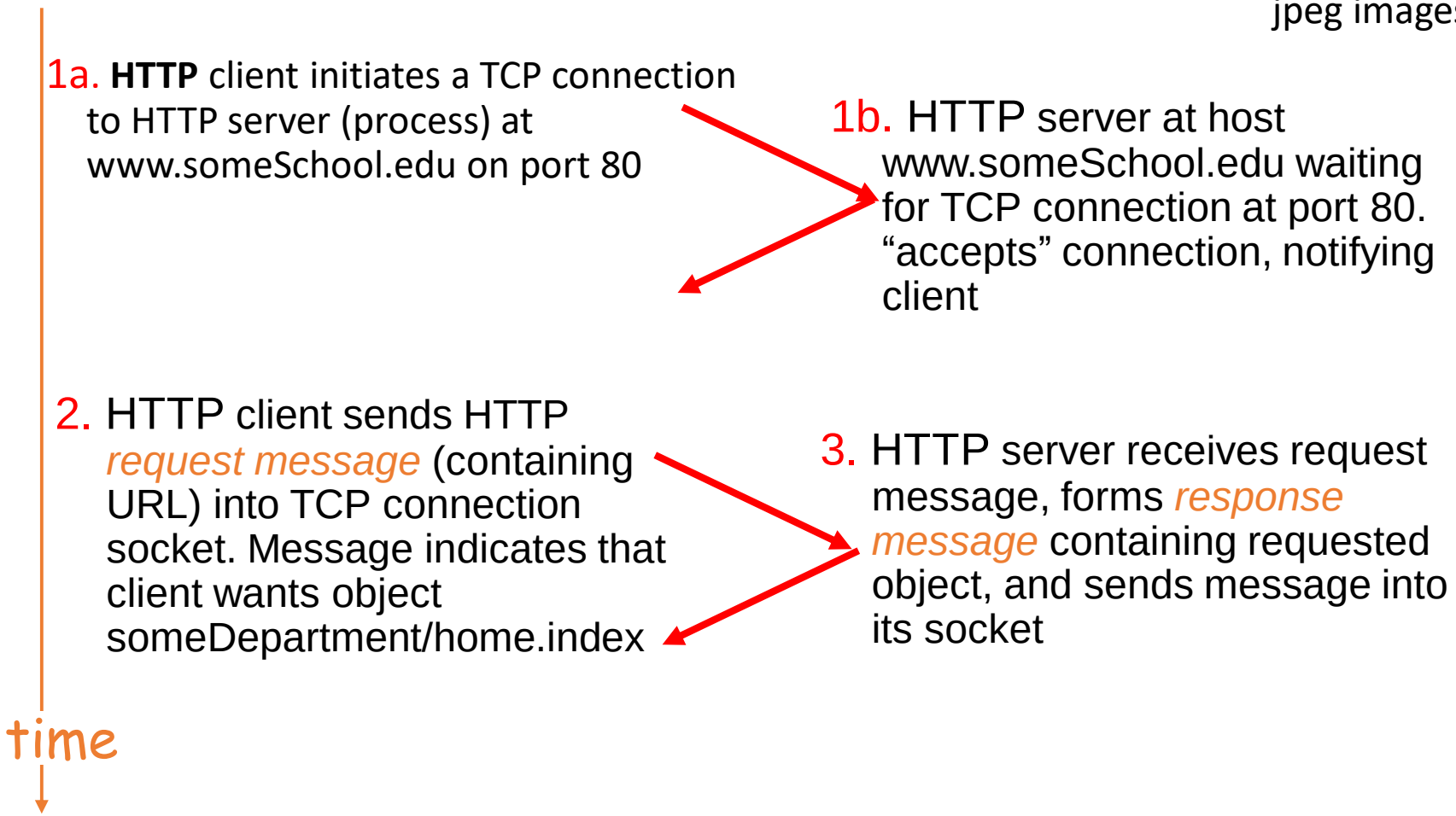
`www.someschool.edu/someDept/pic.gif`

host name | path name

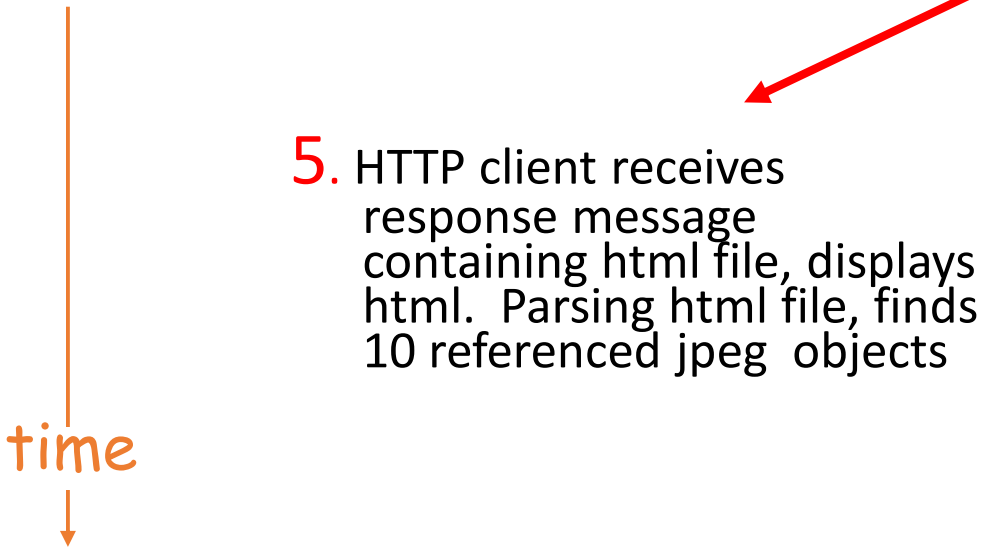
Nonpersistent HTTP

Suppose user enters URL
`www.someSchool.edu/someDepartment/home.index`

(contains text, references to 10 jpeg images)



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

HTTP/1.0 used a non-persistent connection in which at most one object is sent over a TCP connection.

Thus, for transmitting a file from one machine to other required two Round Trip Time (RTT). One RTT to initiate TCP connection as shown in the figure and second for HTTP request and first few bytes of HTTP response to return, and rest of the time is taken in transmitting the file.

```
total = 2RTT+transmit time
```

While using non-persistent HTTP, operating system has an extra overhead for maintaining each TCP connection, that's why browsers often open parallel TCP connections to fetch referenced objects.

Steps for connection of non-persistent HTTP:

- Client (Browser) initiates a TCP connection to www.anyCollege.edu (Server): Handshake
- Server at host www.anyCollege.edu accepts connection and acknowledges
- Client sends HTTP request for file /someDir/file.html
- Server receives message, finds and sends file in HTTP response.
- Client receives response. It terminates connection, parse object.
- Steps 1-5 are repeated for each embedded object.

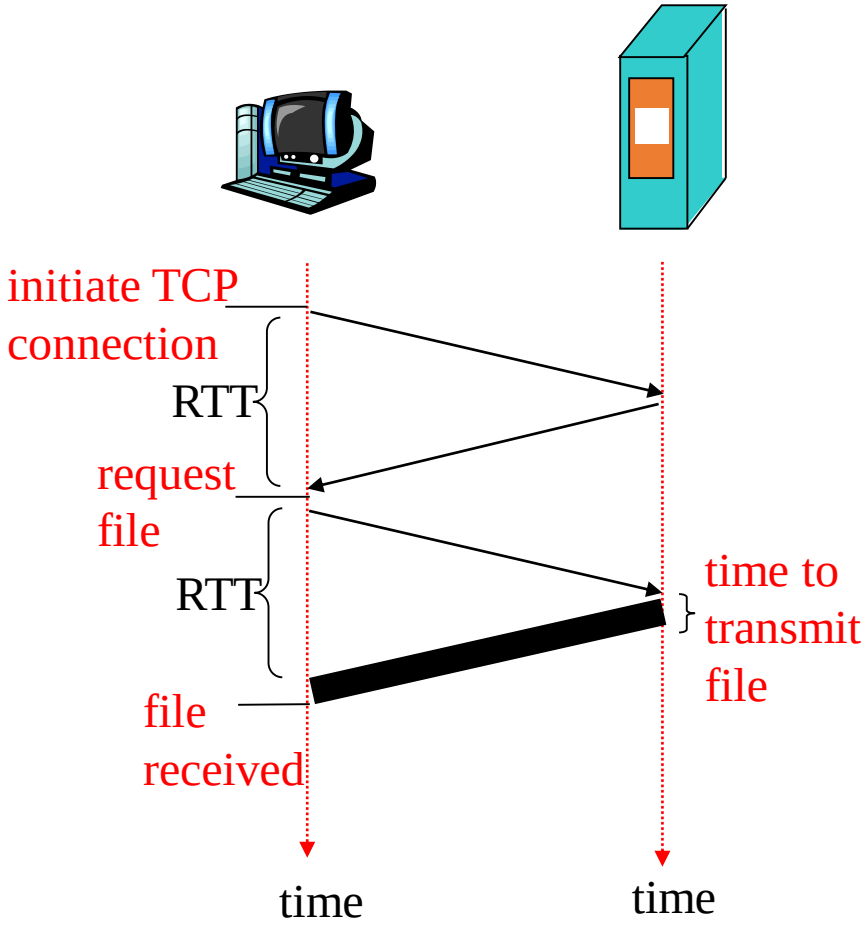
Response time modeling

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 must work and allocate host resources for each TCP connection
- but browsers often open parallel TCP connections to fetch referenced objects

Persistent HTTP

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

Several dimensions to help speed up:

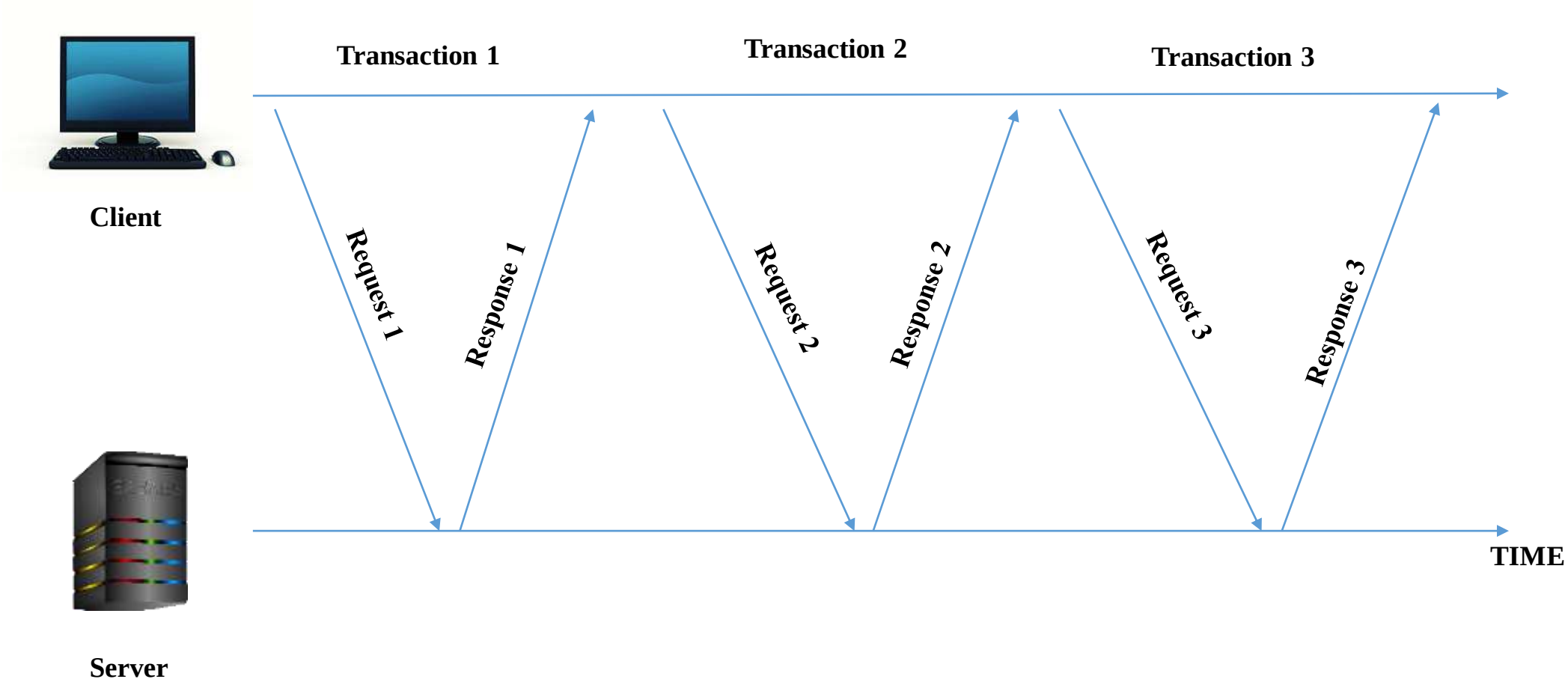
Persistent connections, pipelining, parallel connections

Persistent HTTP

To overcome the issues of HTTP/1.0, HTTP/1.1 came with persistent connections through which multiple objects can be sent over single TCP connection between client and server. As server leaves the connection open after sending response so subsequent HTTP messages between same client/server sent over open connection.

Persistent connection also overcome the problem of slow start as in non-persistent each object transfer suffers from slow start, and also overall number of RTTs required for persistent is much lesser than in non-persistent.

RTT in a Persistent HTTP



Steps for connection of persistent HTTP:

- Client (Browser) initiates a TCP connection to www.anyCollege.edu (Server): Handshake
- Server at host www.anyCollege.edu accepts connection and acknowledges.
- Client sends HTTP request for file /someDir/file.html
- Server receives request, finds and sends object in HTTP response.
- Client receives response. It terminates connection, parse object.
- Steps 3-5 are repeated for each embedded object.

Advantages of Persistent Connection

- Saves CPU time in routers and hosts.
- Reduction in network congestion & latency on subsequent requests.
- Can be used with or without pipelining.
- Overcomes the problem of slow start.

Persistent HTTP without pipeline

In persistent connections without pipelining, the client issues a new request only after the previous response has arrived whereas in with pipelining client sends the request as soon as it encounters a reference, i.e., multiple requests/responses.

Persistent HTTP with pipelining

- HTTP/1.1 supports ***pipelining***, i.e., **i**t allows clients to send multiple requests at once, without waiting for a reply.
- Servers can also send multiple replies without closing their socket. This results in fewer round trips and faster load times.
- This is particularly useful for satellite Internet connections and other connections with high latency as separate requests need not be made for each file.
- Since it is possible to fit several HTTP requests in the same TCP packet, HTTP pipelining allows fewer TCP packets to be sent over the network, reducing network load.
- HTTP pipelining requires both the client and the server to support it.

HTTP Communication

The communication model of HTTP involves an HTTP client (*Web browser)* on a client machine, and an HTTP server (*Web server)*. The basic HTTP communication model has four steps:

- ***Handshaking:*** Opening a TCP connection to the web server.

- ***Client Request:*** After a TCP connection is created, the HTTP client sends a request message formatted according to the rules of the HTTP standard—an *HTTP Request*. This message specifies the resource that the client wishes to retrieve, or includes information to be provided to the server.

HTTP Communication

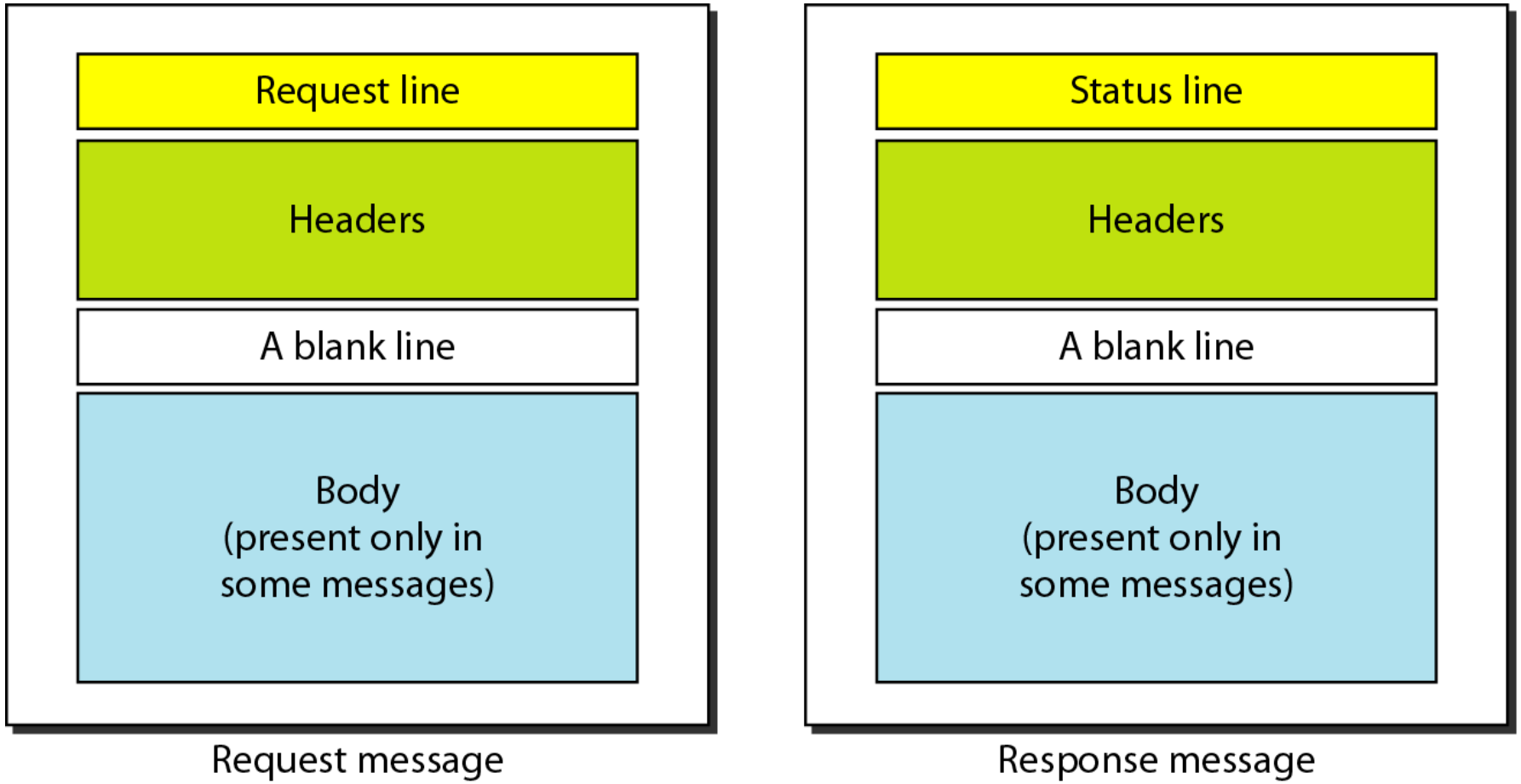
- ***Server Response:*** The server reads and interprets the request. It takes action relevant to the request and creates an HTTP Response message, which it sends back to the client. The response message indicates whether the request was successful, and may also contain the content of the resource that the client requested, if appropriate.

- ***Closing:*** Closing the connection (optional).

Handshaking

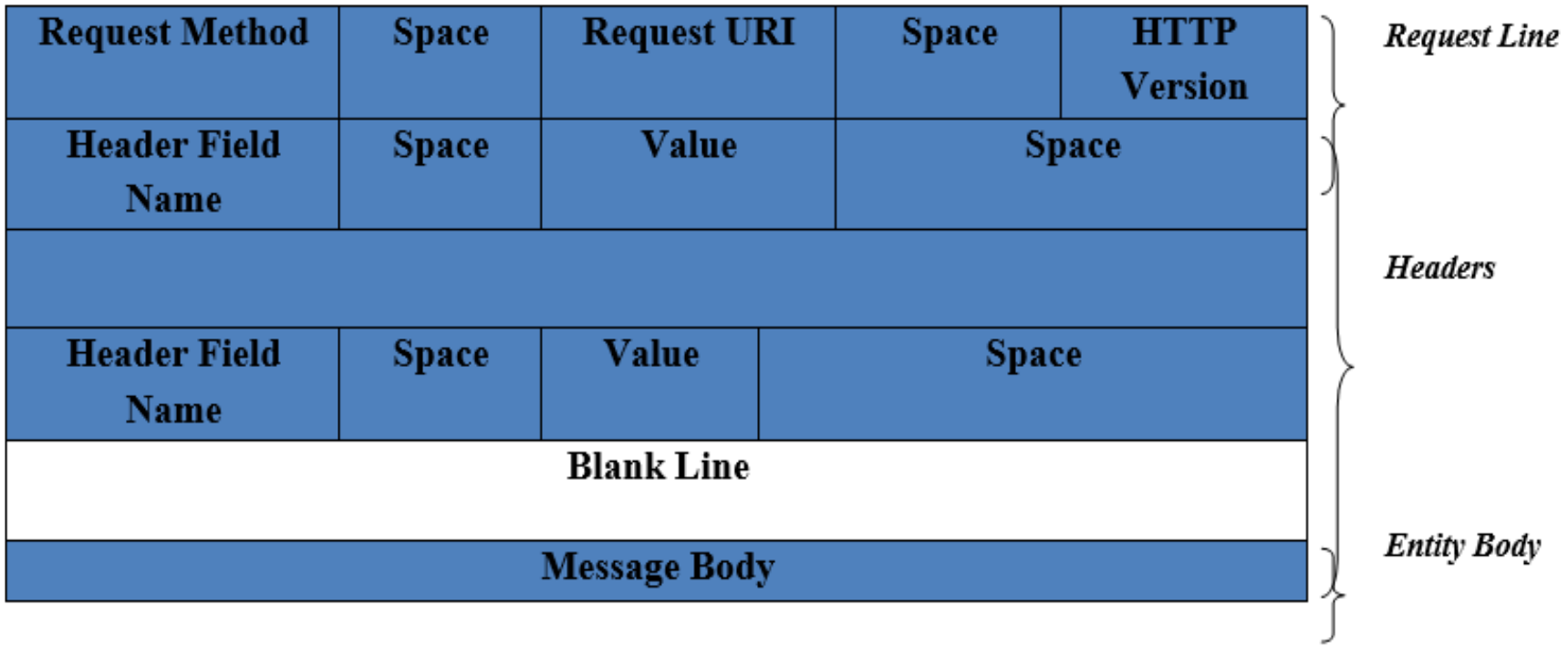
For opening a TCP connection, user on client side inputs the URL containing the name of the web server in the web browser. Then, web browser asks the DNS (Domain Name Server) for the IP address of the given URL. If the DNS fails to find the IP address of the URL, it shows the error on clients' screen: "Netscape (Browser) is unable to locate error". And if the DNS finds the IP address of the URL, then the client browser opens a TCP connection to port 80 (default port of HTTP, although one can specify another port number explicitly in the URL) of the machine whose IP address has been found.

Request and response messages (all in Plain Text)

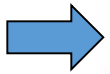


Request Message

After handshaking in the first step, the client (browser) requests an object (file) from the server. This is done with a human-readable message. Every HTTP request message has the same basic structure:



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

Request 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.

Method = GET | HEAD | POST | PUT| DELETE| TRACE | OPTIONS| CONNECT | COPY| MOVE

GET:

Request server to return the resource specified by the Request-URI as the body of a response. It is the most frequently used method in the web. It is specified when a client wants to retrieve a resource from the server. The URL in the request line identifies the resource. If it is 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. The message body is empty for the GET method. This method may also be used to send information (possibly small) to the server for processing without using an HTML form. The information is sent by appending it to the URL using name-value pair.

HEAD:

Requests server to return the same HTTP header fields that would be returned if a GET method were used, but not return the message body that would be returned to a GET (this provides information about a resource without the communication overhead of transmitting the body of the response, which may be quite large).It is useful to inspect the characteristics of the resource (possibly large) without actually downloading it. This effectively saves the bandwidth.

POST:

Request server to pass the body of this request message on as data to be processed by the resource specified by the Request-URI. The actual information is included in the body part of the request message instead of appending it to the URL as done in the GET method. The commonest form of the POST method is to submit an HTML form to the server. Since the information is included in the body, large chunks of data such as an entire file can be sent to the server. Consequently, the length of the request can be unlimited. It is more versatile but slower than using the GET method.

POST:

Request server to pass the body of this request message on as data to be processed by the resource specified by the Request-URI. The actual information is included in the body part of the request message instead of appending it to the URL as done in the GET method. The commonest form of the POST method is to submit an HTML form to the server. Since the information is included in the body, large chunks of data such as an entire file can be sent to the server. Consequently, the length of the request can be unlimited. It is more versatile but slower than using the GET method.

PUT:

Request server to store the body of this message on the server and assign the specified Request-URI to the data stored so that future GET request messages containing this Request-URI will receive this data in their response messages. It is used to upload a new resource or replace an existing document. The actual document is specified in the body part. It is a vulnerable and is not permitted by most of the web servers. However, developers can customize their web servers to accept this method.

DELETE:

Request server to respond to future HTTP request messages that contain the specified Request-URI with a response indicating that there is no resource associated with this Request-URI.

TRACE:

Request server to return a copy of the complete HTTP request message, including start line, header fields, and body, received by the server. Used primarily for test purposes by instructing the web servers to echo the request back to the client.

OPTIONS:

Request server to return a list of HTTP methods that may be used to access the resource specified by the Request-URI. It is usually used to check whether a server is functioning properly before performing other tasks.

CONNECT:

It is used to convert a request connection into the transparent TCP/IP tunnel. It is usually done to facilitate secured socket layer (SSL) encrypted communication e.g. HTTPS, through an unencrypted HTTP proxy server.

COPY:

The HTTP protocol may be used to copy a file from one location to another. The method COPY is used for this purpose. The URL specified in the request line specifies the location of the source file. The location of the target file is specified in the entity header. This method is also vulnerable.

MOVE:

It is similar to the COPY method except that it deletes the source file.

Request-URI and HTTP Version

Request-URI: The second part of the start line is known as the Request-URI. It identifies the resource upon which to apply the request.

HTTP version: HTTP/1.1 version is used as it is the latest available version.

Headers

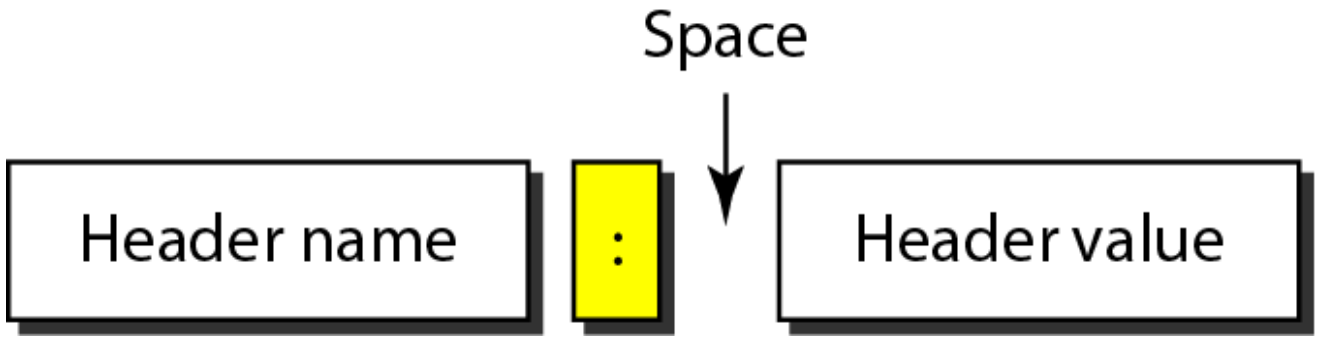
- HTTP headers are a form of message metadata.
- Use of headers makes it possible to establish and maintain sessions, set caching policies, control authentication, and implement business logic.
- They collectively specify the characteristics of the resource requested and the data that are provided.
- For example, a client may want to accept the audio files only in some specified format.
- The HTTP protocol specification makes a clear distinction between general headers, request headers, response headers, and entity headers.

The headers are separated by an empty line from the request and response body.

Format of request header:

General Header
Request Header
Entity Header

Header format



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 27.4 *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

A header consists of a single line or multiple lines. Each line is a single header of the following form:

Header-name : Header-value

The header name is not case-sensitive; but the header value may be. If a header line starts with a space, it is considered to be a part of the previous header line. This happens if a long header value is broken into multiple lines for easy reading. So, the following headers are equivalent.

Date: Mon, 12 Sept 2016 16:09:05 GMT

Date: Mon, 12 Sept 2016
16:09:05 GMT

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.

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.

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 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

Figure 27.16 *Example 27.1*

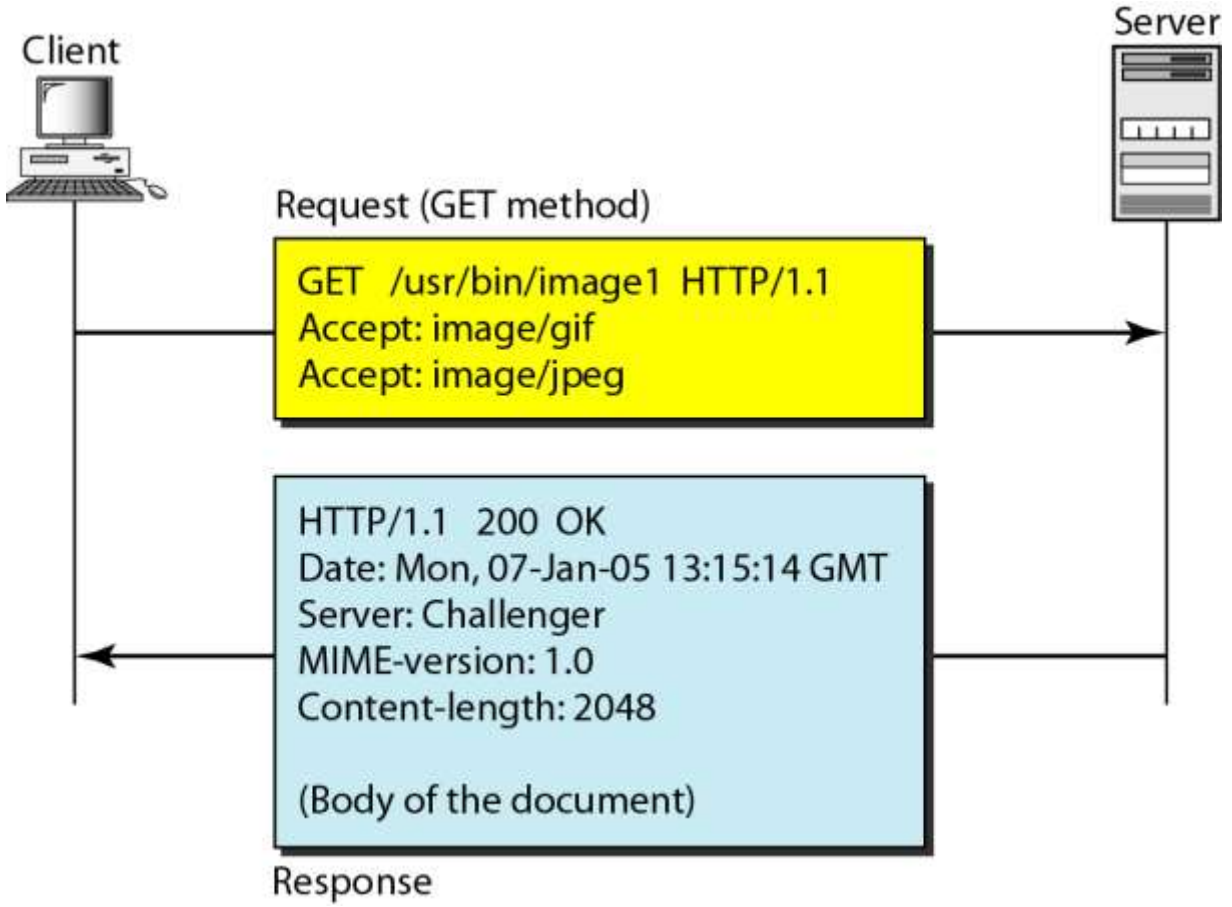
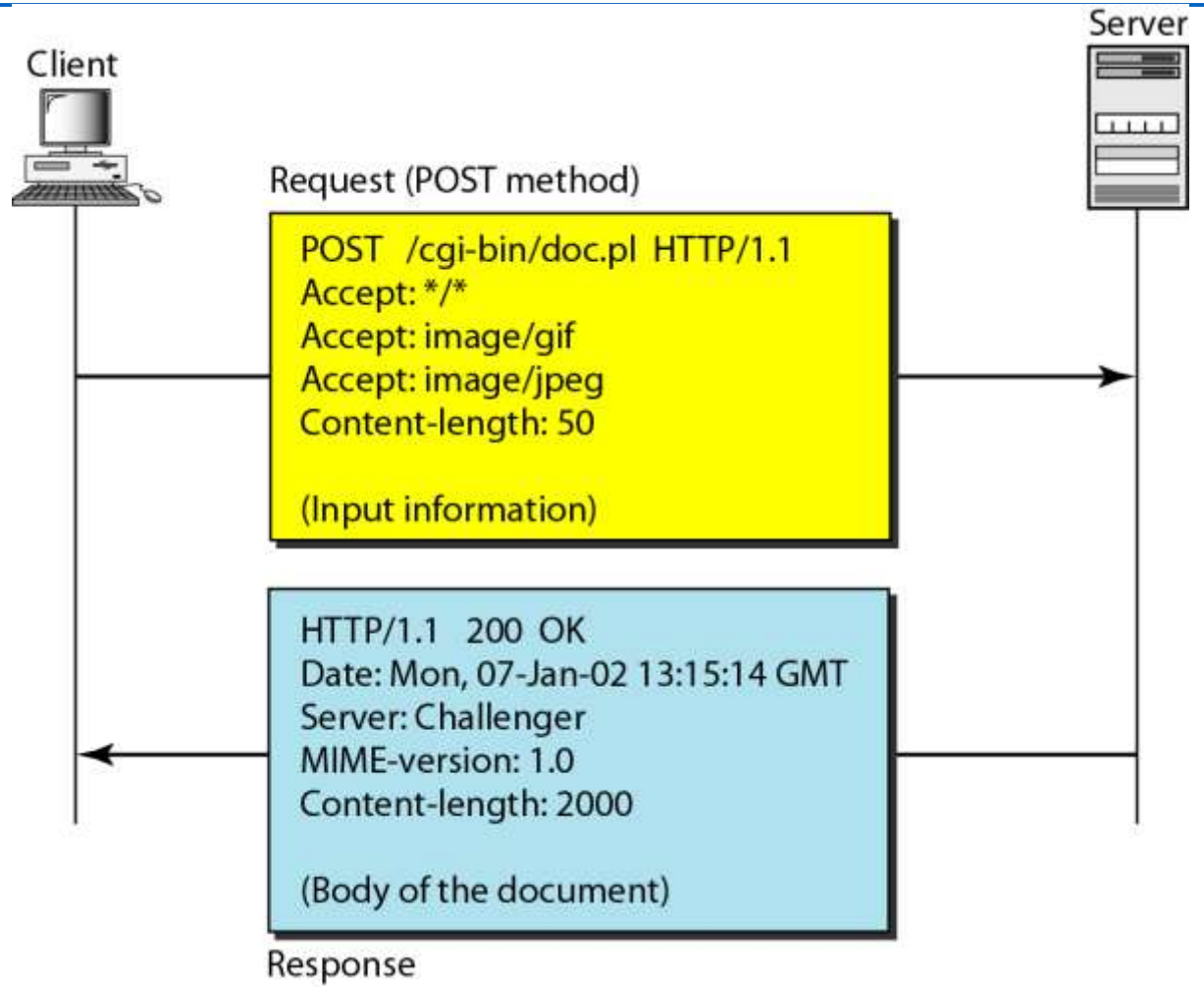


Figure 27.17 *Example 27.2 (client sends data to server)*



Example 27.3 (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
```

General Headers

- General headers do not describe the body of the message.
- They provide information about the messages instead of what content they carry.
- They are primarily used to specify how messages should be processed and handled.

Some of the General Headers

- ***Date: Mon, 12 Sept 2016 16:09:05 GMT***

  This header specifies the time and date of the creation of the message.

- ***Connection: Close***

  This header indicates whether or not the client or server that generated the message intends to keep the connection open.

- ***Warning: Danger, This site may be hacked!***

  This header stores text for human consumption, something that would be useful when tracing a problem.

- ***Cache-Control: no-cache***

  This header shows whether the caching should be used.

Request Header

It allows the client to pass additional information about themselves and about the request such as the data format that the client expects.

- ***User-Agent: Mozilla/4.75***

  Identifies the software (e.g. a web browser) responsible for making the request.

- ***Host: www.netsurf.com***

  This header was introduced to support virtual hosting, a feature that allows a web server to service more than one domain.

- ***Referer: http://wwwdtu.ac.in/~akshi/index.html***

  This header provides the server with context information about the request. If the request came about because a user clicked on a link found on a web page, this header contains the URL of that referring page.

- ***Accept: text/plain***

  This header specifies the format of the media that the client can accept.

Entity Header

It contains the information about the messages body or the target messages in case of the request messages have no body.

Content-Type: mime-type/mime-subtype

This header specifies the MIME type of the message body's content.

Content-Length: 546

This optional header provides the length of the message body. Although it is optional, it is useful for clients such as web browsers that wish to impart information about the progress of a request. Where this header is omitted, the browser can only display the amount of data downloaded. But when the header is included, the browser can display the amount of data as a percentage of the total size of the message body.

Last-Modified: Sun, 11 Sept 2016 13:28:31 GMT

This header provides the last modification date of the content that is transmitted in the body of the message. It is critical for the proper functioning of caching mechanisms.

Allow: GET, HEAD, POST

This header specifies the list of the valid methods that can be applied on a URL.

Message Body

- Optional part for HTTP.
- If available, then use to carry the entity-body associated with the request.
- If entity body is associated, then usually Content-Type and Content-Length headers lines specify the nature of the body associated.
- Carries the actual HTTP request data (including form data and uploaded, etc.).

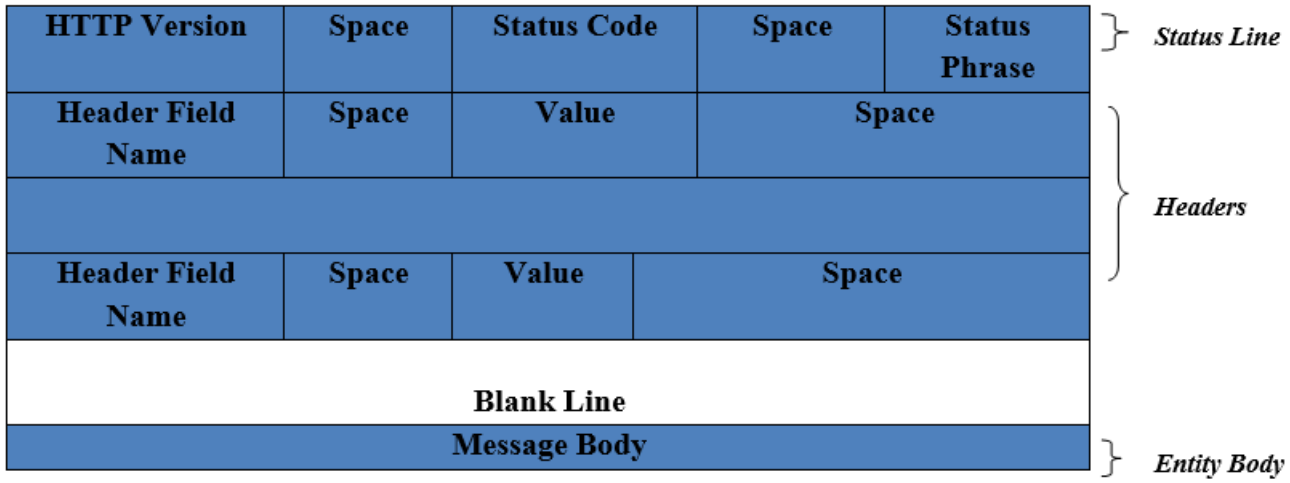
A sample HTTP Request message with no message body

GET/index.html HTTP/1.1	**Request Line**	**HTTP Request**
Date: Fri, 16 Sep 2016 22:08:32 GMT **Connection: Close**	**General headers**	
User-Agent: Mozilla/5.0 **Host: www.drakshikumar.com** **Accept: text/html, text/plain** **Accept-Language: en-US** **Content- Length: 35**	**Request header**	
BLANK LINE		
	Message Body	

Response Message

Server responds to the HTTP request of the client by providing the requested document (object, entity), e.g. it transfers an HTML file. Before sending the real data, a status code is sent (e.g. 202, “Accepted”), as well as information about the document and the server.

A HTTP response message consists of a status line, header fields and the body of the



Status Line

Status line consists of three parts: HTTP version, Status code, and Status phrase.

HTTP version		**Status Code**		**Status phrase**

- **HTTP Version:** This field specifies the version of the HTTP protocol being used by the server. The current version is HTTP/1.1.
- **Status Code:** It is a three-digit code that indicates the status of the response.
- **Status Phrase:** It is also known as Reason-phrase, intended to give a short textual description of status code.

Headers

Headers in HTTP response message are similar to the one in request message except for one thing, in place of request header in the headers it contains a response header. The general header and entity header has same purpose and structure as request message.

Response headers help the server to pass additional information about the response that cannot be inferred from the status code alone, like the information about the server and the data being sent.

Example-

Location: http://www.mywebsite.com/relocatedPage.html

This header specifies a URL towards which the client should redirect its original request. It always accompanies the '301' and '302' status codes that direct clients to try a new location.

Message Body

Similar to HTTP request messages, message body in HTTP response message is also optional. Message body carries the actual HTTP response data from the server (including files, images, etc.) to the client.

A sample HTTP Response Message

HTTP/1.1 200 OK	**Status Line**	**HTTP Response**
Date: Fri, 16 Sep 2016 22:08:32 GMT **Connection: Close**	**General headers**	
Server: Apache/2.4.7 **Accept-Ranges: bytes**	**Response header**	
Content-Type: text/html **Content- Length: 95** **Last-Modified: Mon, 5 Sept 2016 10:14:24 GMT**	**Entity header**	
BLANK LINE		
<html> <head> <title> Welcome to my Homepage </title> </head> <body> Coming Soon! </body> </html>	**Message Body**	

Hypertext Transfer Protocol Secure (HTTPS)

HTTPS is a protocol for secure communication over the Internet. It was developed by Netscape. It is not a protocol but it is just the result of combination of the HTTP and SSL/TLS(Secure Socket Layer/ Transport Layer Security) protocol. It is also called secure HTTP, as it sends and receives everything in the encrypted form, adding the element of safety.

HTTPS is often used to protect highly confidential online transactions like online banking and online shopping order forms. The use of HTTPS protects against eavesdropping and man-in-the-middle attacks. While using HTTP servers and clients still speak exactly the same HTTP to each other, but over a secure SSL connection that encrypts and decrypts their requests and responses.

SSL Layer

The SSL Layer has 2 main purposes:

- Verifying that you are talking directly to the server that you think you are talking to.
- Ensuring that only the server can read what you send it and only you can read what it sends back.

HTTP State Retention: Cookies

Cookies are an application based solution to provide state retention over a stateless protocol. They are small pieces of information that are sent in response from the web server to the client. Cookies are the simplest technique used for storing client state. A cookie is also known as HTTP cookie, Web cookie or Browser cookie.

Cookies are not software; they cannot be programmed, cannot carry viruses and cannot install malware on the host computer.

Cookies are stored on client's computer. They have a lifespan and are destroyed by the client browser at the end of that lifespan.

A cookie is a pair of name and value, as in (name, value). A web application can generate multiple cookies, set their life spans in terms of how many milliseconds each of them should be alive, and send them back to a web browser as part of an HTTP response.

If cookies are allowed, a web browser will save all cookies on its hosting computer, along with their originating URLs and life spans.

When an HTTP request is sent from a web browser of the same type on the same computer to a web site, all live cookies originated e as part of the HT tored in cookies i.e

Creating Cookies

When receiving an HTTP request, a server can send a Set-Cookie header with the response. The cookie is usually stored by the browser and, afterwards, the cookie value is sent along with every request made to the same server as the content of a Cookie HTTP header. Additionally, an expiration delay can be specified as well as restrictions to a specific domain and path, limiting how long and to which site the cookies is sent to.

A simple cookie can be set like this:

Set-Cookie: <cookie-name>=<cookie-value>

Advantages of Cookie

- **Persistence:** One of the most powerful aspects of cookies is their persistence. When a cookie is set on the client's browser, it can persist for days, months or even years. This makes it easy to save user preferences and visit information and to keep this information available every time the user returns to your site. Moreover, as cookies are stored on the client's hard disk so if the server crashes they are still available.

- **Transparent:** Cookies work transparently without the user being aware that information needs to be stored.

- They lighten the load on the server's memory.

HTTP Cache

Caching is the term for storing reusable responses in order to make subsequent requests faster. The caching of web pages is an important technique to improve the Quality of Service (QoS) of the web servers.

Caching can reduce network latency experienced by clients. For example, web pages can be loaded more quickly in the browser.

Caching can also conserve bandwidth on the network, thus increasing the scalability of the network with the help of an HTTP proxy cache (a.k.a. Web Cache).Caching also increases the availability of web pages.Web resources that are cached remain accessible even if the source of the resources or an intermediate network link goes down.

Content can be cached at many different points throughout the delivery chain:

- **Browser cache**: Web browsers themselves maintain a small cache. Typically, the browser sets a policy that dictates the most important items to cache. This may be user-specific content or content deemed expensive to download and likely to be requested again.

- **Intermediary caching proxies (Web proxy)**: Any server in between the client and your infrastructure can cache certain content as desired. These caches may be maintained by ISPs or other independent parties.

- **Reverse Cache**: Your server infrastructure can implement its own cache for backend services. This way, content can be served from the point-of-contact instead of hitting backend servers on each request.

Web Proxy

A ***Web proxy*** is the most commonly used caching service. A proxy is an application program or a computer system that behaves like an intermediary between servers and clients looking for services from these servers. To access a resource, a client sends the request to proxy instead of the original web server. The proxy, in turn, searches for the resource in its cache. If the resource is found, it is deliverable to the client. Otherwise, it contacts the specified server and gets the resource, puts that resource in its local cache, and finally returns the resource to the client. The subsequent requests for the same resource are served by the copy in its local cache thereby avoiding remote web server access. This way proxy speeds up the access to the resource

Advantages of using a cache

- **Decreased network costs**: Content can be cached at various points in the network path between the content consumer and content origin. When the content is cached closer to the consumer, requests will not cause much additional network activity beyond the cache.
- **Improved responsiveness**: Caching enables content to be retrieved faster because an entire network round trip is not necessary. Caches maintained close to the user, like the browser cache, can make this retrieval nearly instantaneous.
- **Increased performance on the same hardware**: For the server where the content originated, more performance can be squeezed from the same hardware by allowing aggressive caching. The content owner can leverage the powerful servers along the delivery path to take the brunt of certain content loads.
- **Availability of content during network interruptions**: With certain policies, caching can be used to serve content to end users even when it may be unavailable for short periods of time from the origin servers.

Cache Consistency

Cache consistency mechanisms ensure that cached copies of web pages are eventually updated to reflect changes in the original web pages.

There are basically, two cache consistency mechanisms currently in use for HTTP proxies:

- Pull Method
- Push Method

Pull Method

In this mechanism, each web page cached is assigned a time-to-serve field, which indicates the time of storing the web page in the cache. An expiry time of one or two days is also maintained, it determines how long a web page should remain valid in the cache. If the time is expired, a fresh copy is obtained when user requests for the page.

Push Method

In this mechanism, the web server is assigned the responsibility of making all cached copies consistent with the server copy. Whenever a web page is modified, the web server sends an update request to every web proxy that cached this web page. The advantage of this is that the cached pages are modified only when the original copies are modified, but this also increases the load of the web server.

HTTP
Hyper text Transfer Protocol

Hyper Text Transfer Protocol (HTTP)

- HTTP stands for hypertext transfer protocol and is used to transfer data across the Web.
- It is a critical protocol for web developers to understand and because of it widespread use it is also used in transferring data and commands in IOT applications.
- The first version of the protocol had only one method, **namely GET**, which would request a page from a server.
- The response from the server was always an HTML page

- There have been several versions of HTTP starting with the original 0.9 version.

- The current version is 1.1 and was last revised in 2014.

- HTTP uses a <major>.<minor> numbering scheme to indicate versions of the protocol. The version of an HTTP message is indicated by an HTTP-Version field in the first line. Here is the general syntax of specifying HTTP version number:

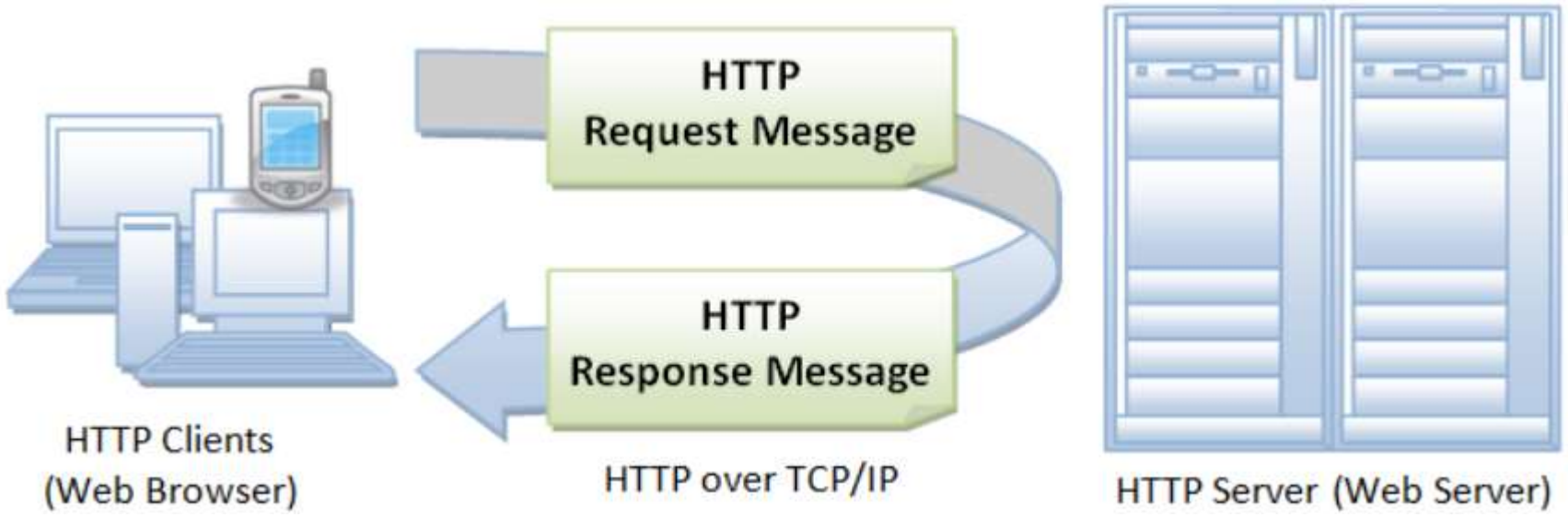
- HTTP-Version = "HTTP" "/" 1*DIGIT "." 1*DIGIT

  Example

HTTP/1.0 or HTTP/1.1

How it works

- Like most of the Internet protocols http it is a command and response text based protocol using a client server communications model.



- HTTP is an **asymmetric request-response client-server** protocol. An HTTP client sends a request message to an HTTP server. The server, in turn, returns a response message. In other words, HTTP is a pull protocol, the client pulls information from the server (instead of server pushes information down to the client).

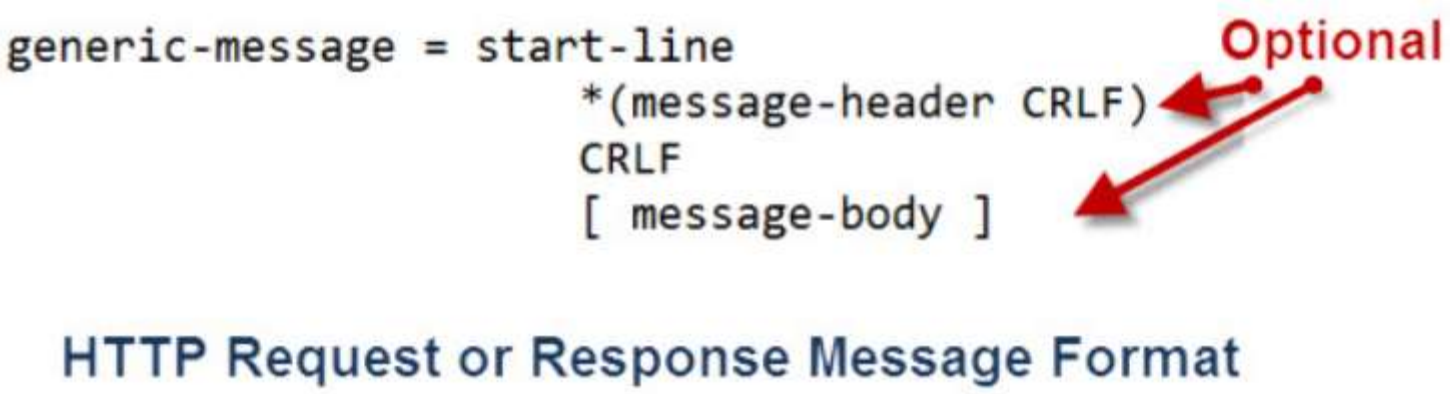
- The HTTP protocol is also a stateless protocol meaning that the server isn't required to store session information, and each request is independent of the other.

- This means:

  - All requests originate at the client (your browser)
  - The server responds to a request.
  - The requests(commands) and responses are in readable text.
  - The requests are independent of each other and the server doesn't need to track the requests.

Request and Response Structure

- Request and response message structures are the same and shown



HTTP Request or Response Message Format

- A request consists of:

A command or request + optional headers + optional body content

- A response consists of:

A status code + optional headers + optional body content

- A simple **CRLF (Carriage Return and Line feed)** combination is used to **delimit the parts**, and a **single** blank line (CRLF) indicates **end of the headers**.

- If the request or response contains a message body then this is indicated in the header.

- The presence of a message body in a request is signalled by a **Content-Length or Transfer-Encoding header field**. Request message framing is independent of method semantics, even if the method does not define any use for a message body.

HTTP Requests

- We saw the general request response format earlier now we will cover the request message in more detail.

- **The start line is mandatory and is structured** as follows:

Method + Resource Path + protocol version

- Example if we try to access the web page testpage.htm on www.testsite5.com

- The start line of the request would be

GET/ testpage.htm HTTP/1.1

- Where

  GET is the method

  /testpage.htm is the relative path to the resource.

  HTTP/1.1 is the protocol version we are using.

- **A relative path doesn’t include the domain name.**
- **The web browser uses the URL that we enter to create the relative URI of the resource.**

- **URL (uniform resource Locator) is used for web pages. It is an example of a URI (uniform resource indicator).**

- The actual http request is not shown by the browser, and is only visible using special tools like **http header live**.

HTTP vs URL

- Most people are familiar with entering a url into a web browser. Usually looking like this.

URL Example

- The url can also includes the port which is normally hidden by the browser, but you can manually include it as shown below:

URL Example with PORT

- This tells the web browser the address of the resource to locate and the protocol to use to retrieve that resource **(http)**.

- **http** is the transfer protocol that transfer the resource (web page,image,video etc) from the server to the client.

HTTP Responses and Response Codes

- Each request has a response. The Response consists of a
  - **STATUS code And Description**
  - **1 or more optional headers**
  - **Optional Body message can be many lines including binary data**

- Response Status codes are split into 5 groups each group has a meaning and a three digit code.
  - **1xx – Informational**
  - **2xx – Successful**
  - **3xx -Multiple Choice**
  - **4xx– Client Error**
  - **5xx -Server Error**

- For example a successful page request will return a 200 response code and an unsuccessful a 400 response code.

➢ 1xx – Informational

100 Control
101 Switching Protocols

➢ 2xx – Successful

200 Ok
201 Created
202 Accepted
203 Non-Authoritative Information
204 No Content
205 Reset Content
206 Partial Content

➢ 3xx -Multiple Choice

301 Moved Permanently
302 Found
303 See other
304 Not Modified
305 Use Proxy
306 Unused
307 Temporary Redirect

➢ 4xx– Client Error

400	Bad Request
401	Unauthorized
402	Payment Required
403	Forbidden
404	Page Not Found
405	Method Not allowed
406	Not acceptable
407	Proxy Authentication required
408	Request Time out
409	Conflict
410	Gone
411	Length Required
412	Precondition failed
413	Request Entity too large
414	Request URO too long
415	Unsupported Media Type
416	Requested Range not Satisfiable
417	Expectation failed

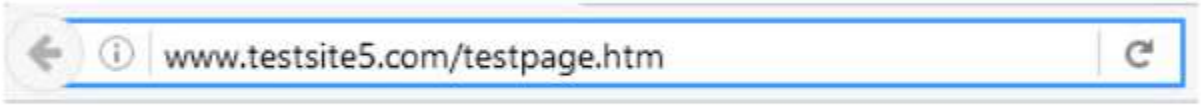
➢ 5xx -Server Error

500	Internal Server Error
501	Not Implemented
502	Bad gateway
503	Service Unavailable
504	Gateway Timeout
505	HTTP version Not Supported

Request Response Example

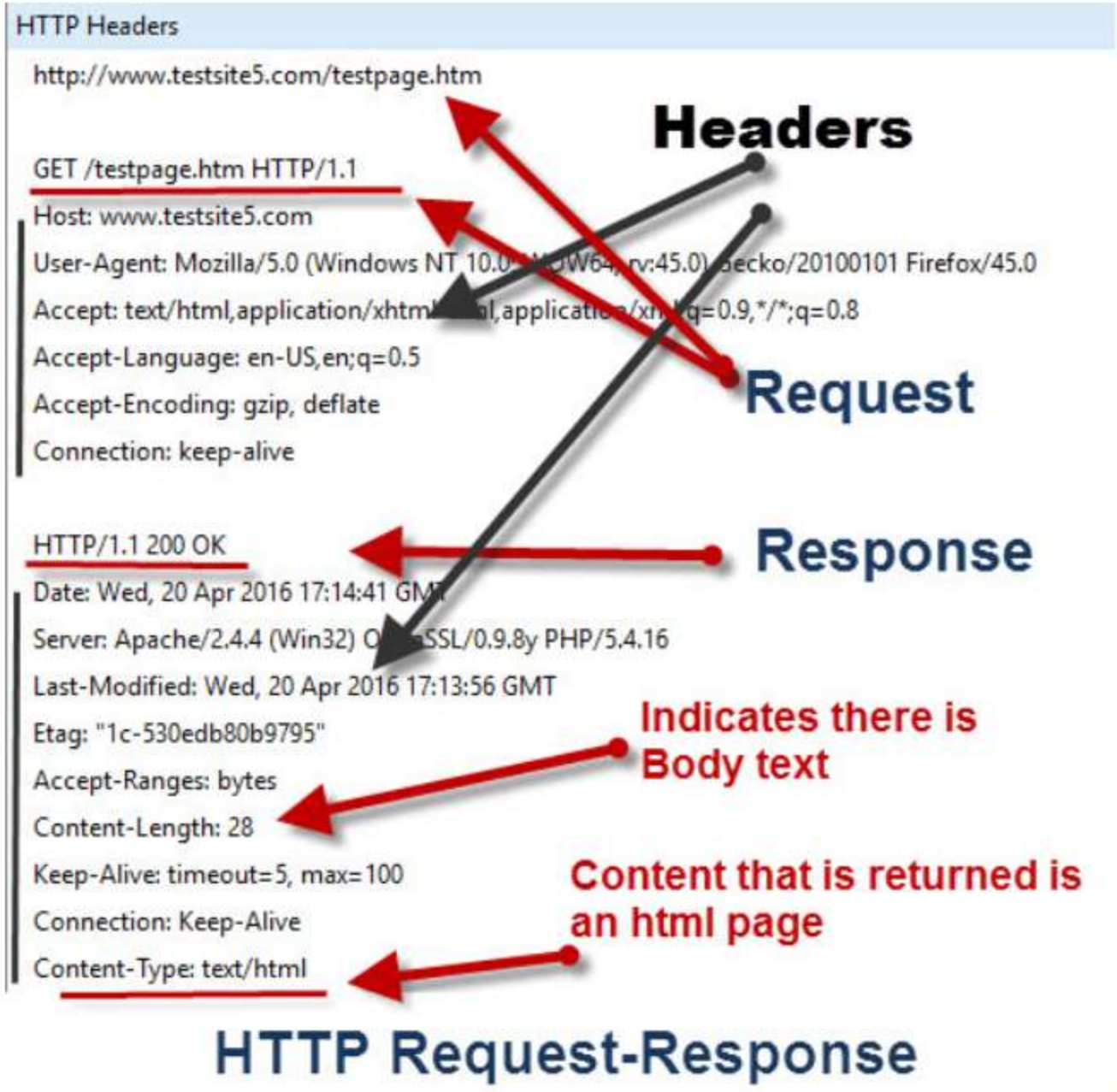
- We are going to examine the requests and response when we access a simple web page (testpage.htm)

Here is what I enter in the browser address bar:



and this is the response that the browser displays:

and here is a screen shot of the **http request-response** that happens behind the scenes.



- **Notice** the request headers are automatically inserted by the browser, and so are the response headers are inserted by the web server.

- There is no body content in the request. The body content in the reply is a web page, and is shown in the browser, and not by the **live headers tool**.

Request Types

- So far we haven’t mentioned request types, but we have seen the GET request type in our examples.
- The GET request type or method is used for requesting a resource from a web server.
- GET is most commonly used request type and was the only request type in the Original HTTP specification.

Request Types, Methods or Verbs

- The HTTP protocol now support 8 request types, also called methods or verbs in the documentation, they are:

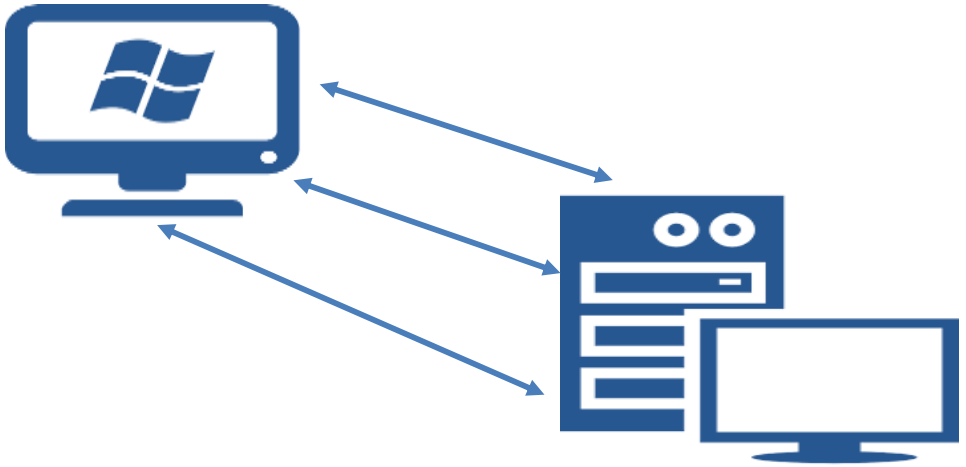
  GET – Requesting resource from server
  POST – Submitting a resource to a server (e.g. file uploads)
  PUT - As POST but replaces a resource
  DELETE- Delete a resource from a server
  HEAD – As GET but only return headers and not content
  OPTIONS - Get the options for the resource
  PATCH - Apply modifications to a resource
  TRACE - Performs message loop-back

- On the Internet today the GET (getting web pages) and POST (submitting web forms)methods are the ones most commonly used.

- The other methods are used when working with Web and IOT APIs specifically put, delete and head.

NON-PERSISTENT HTTP
&
PERSISTENT HTTP CONNECTION

- In Client-Server communication, Client making a series of requests to server, Server responding to each of the requests.
- Series of requests may be made back to back or periodically at regular time interval.
- So, Application developer need to make an important decision;
  - ✓ Should each request/response pair be sent over a separate TCP connection.
  - ✓ **OR** should all of the requests and corresponding responses be sent over same TCP connection?



Non-persistent HTTP

- A non-persistent connection is closed after the server sends the requested object to the client.
- The connection is used exactly for one request and one response.
- For downloading multiple objects it required multiple connections.
- Non-persistent connections are the default mode for HTTP/1.0.
- Example:
- Transferring a webpage from server to client, webpage consists of a base HTML file and 10 JPEG images.
- Total 11 object are reside on server.

Non-persistent HTTP – Cont.…

www.someSchool.edu/someDepartment/home.index

URL:

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

4. HTTP server closes TCP connection.

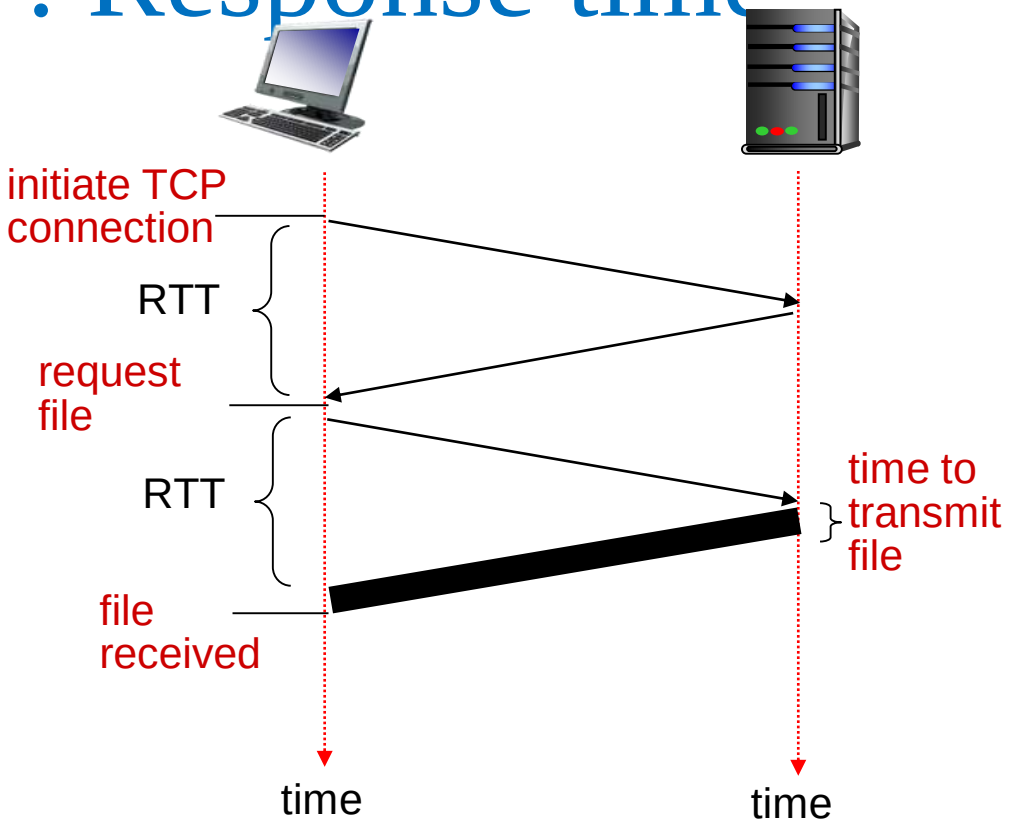
5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects

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

Time

Non-persistent HTTP: Response time

- RTT(round-trip time): A time for a small packet to travel from client to server and vice versa.
- HTTP 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



Non-persistent HTTP response time = 2RTT + file transmission time

Persistent HTTP

- Server leaves the TCP connection open after sending responses.
- Subsequent HTTP messages between same client and server sent over open connection.
- The server closes the connection only when it is not used for a certain configurable amount of time.
- It requires as little as one round-trip time (RTT) for all the referenced objects.
- With persistent connections, the performance is improved by 20%.
- Persistent connections are the default mode for HTTP/1.1.

HTTP Message Format

- Two types:
  1. Request Message
  2. Response Message

1. HTTP Request Message

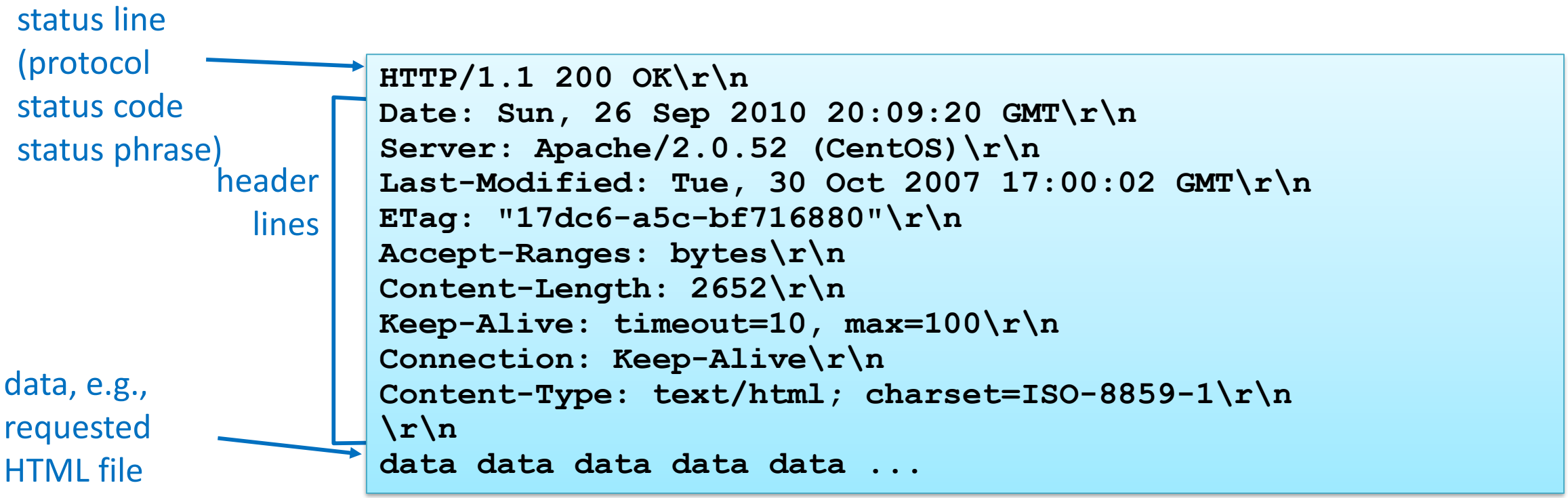
- It is in ASCII format which means that human-readable format.
- HTTP request message consist three part:
  - Request line
  - Header line
  - Carriage return

1. HTTP Request Message - Format

- The request line has three fields: Method field, URL field, and HTTP version field.
- The method field can take on several different values, including GET, POST, HEAD, PUT, and DELETE.
- In above message, browser is requesting the object */somedir/page.html* and version is self-explanatory; browser implements version HTTP/1.1.
- The header line Host: *www-net.cs.umass.edu* specifies the host on which the object resides.
- User agent indicate browser name and version.

2. HTTP Response Message

- HTTP response message consist of three part:
  1. Status line
  2. Header line
  3. Data (Entity body)



2. HTTP Response Message - Format

- The status line has three fields: protocol version field, status code and corresponding status message.
- In below example, the status line indicates that the server is using HTTP/1.1 and that everything is OK.

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n \r\n
data data data data data ...
```

[illegible]

HTTP Response Status Codes

- A status code appears in 1st line in server-to-client response message.
- Some 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
    - Requested http version not support

HTTP Connections

- Non-Persistent
- Persistent
- Before starting with persistent and non-persistent HTTP connection lets know what is a RTT.
- **RTT-**> Time for a small packet to travel from client to server and back.

RTT= 2*propagation time

1. For an connection Persistent or Non-persistent it is sure that to initiate TCP connection one RTT is used.
2. One RTT is used for HTTP request and first few bytes to HTTP response to return.

- So in order to know total file transmission time

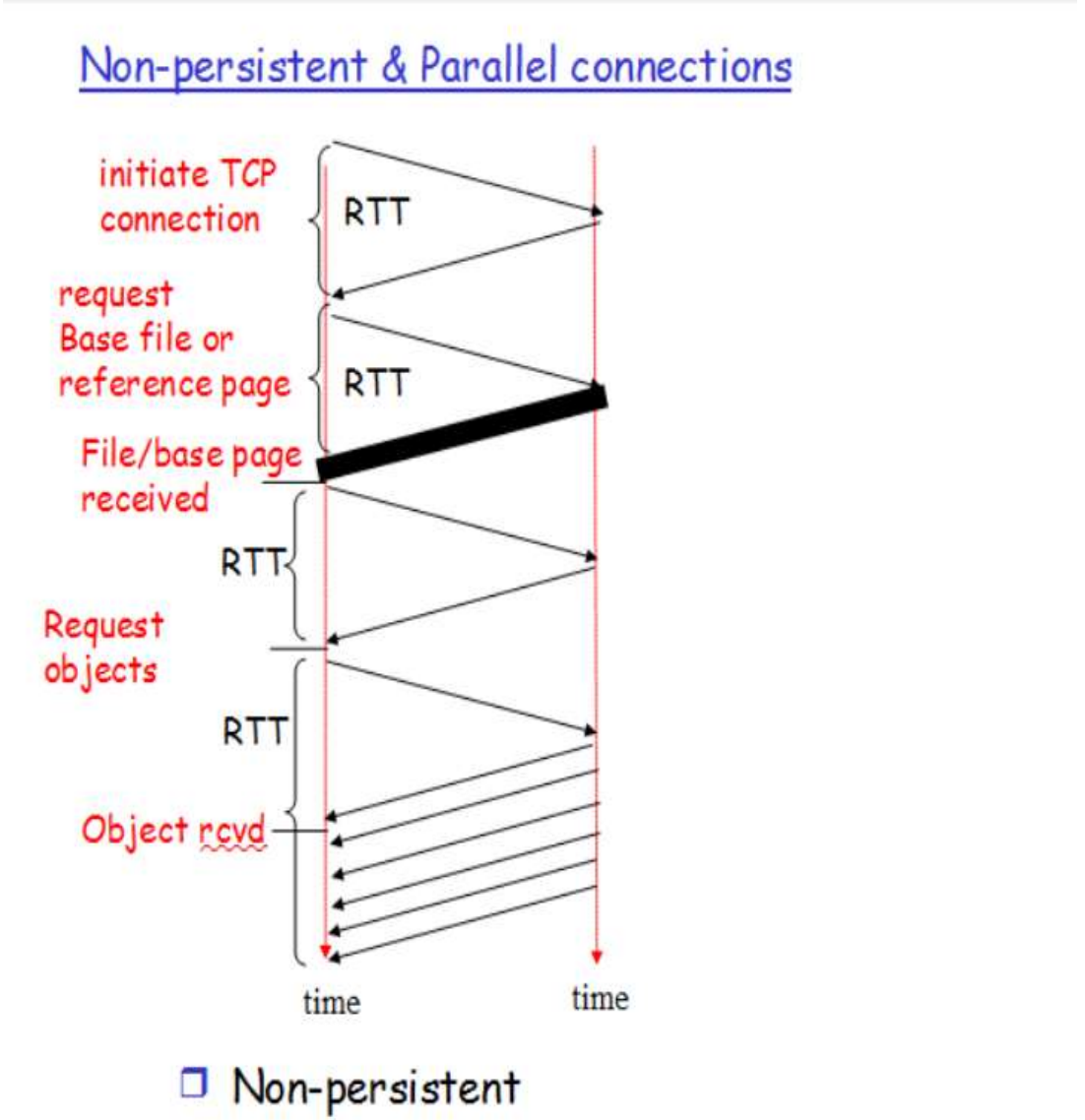
total = 2RTT+transmit time

Non-Persistent Connection

- Without parallel connection
- With parallel connection

- **Without parallel connection Non-Persistent**
  Each objection takes two RTT (assuming no window limit) one for TCP connection and other for HTTP image/text file.

With parallel connection Non-Persistent

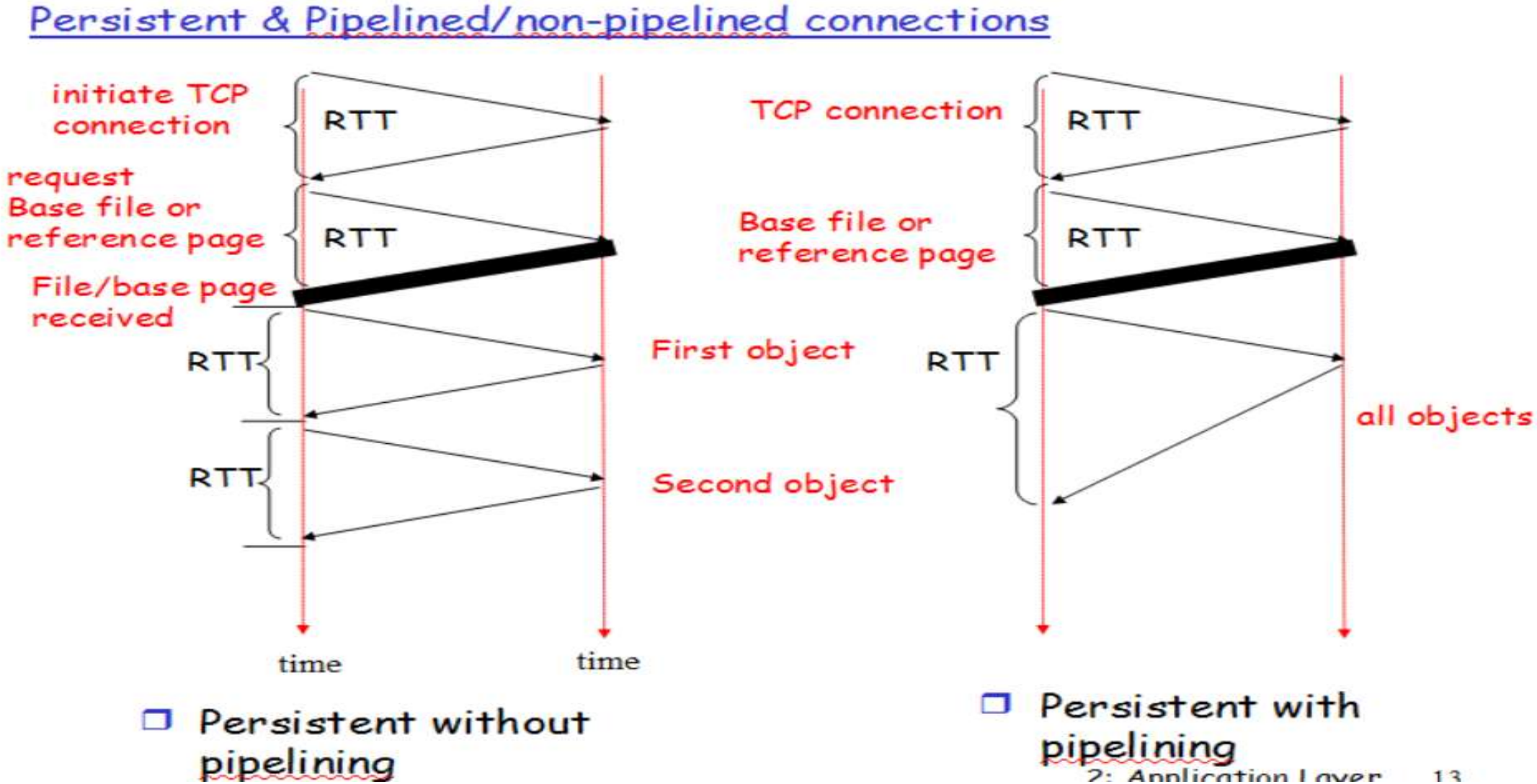


- Non-persistent

Persistent connection

Non-Pipelined

Pipelined



- In **Non-pipeline connection** we first establish connection which takes two RTT then we send all the objects images/text files which takes 1 RTT each (TCP for each object is not required).

- In **Pipelined connection** 2RTT for connection establishment and then 1RTT(assuming no window limit) for all the objects i.e. images/text.

HTTP Non-Persistent & Persistent Connection

- For question point of view you need to know that Non-persistent connection is known as *HTTP 1.0* and Persistent connection is known as *HTTP 1.1*.

  - **Non-Persistent Connection**: It requires connection setup again and again for each object to send.
  - **Persistent connection**: It does not require connection setup again and again. Multiple objects can use connection.

Questions :

Assume that you have base HTML file with 30 embedded images, images & base file are small enough to fit in one TCP segment. How many RTT are required to retrieve base file & images under-following condition :

(i) Non-Persistent connection without parallel connection
(ii) Non-persistent connection with 10 parallel connection
(iii) Persistent connection without pipe-lining
(iv) Persistent connection with pipe-lining

2RTT is the initial required connection one for TCP connection and one for HTML base file.

Total time = 2RTT + transmit time

(i) Non-Persistent connection with no parallel connection :

Here for each image 2 RTT are required one for TCP connection and one for image to send.
So transmit time for 30 images = 2*(30 RTT) = 60 RTT

Total time = 2 RTT+60 RTT = 62RTT

(ii) Non-persistent connection with 10 parallel connection :

Here 10 images can be send simultaneously.
So for 30 images it required -> 2*(30/10) = 6RTT

Total time = 2 RTT + 6 RTT = 8RTT

(iii) Persistent connection without pipelining :

Here TCP connection is not required again and again.
So for 30 images it requires -> 30 RTTs

Total time = 2 RTT + 30 RTT = 32RTT

(iv) Persistent connection with pipe-lining :

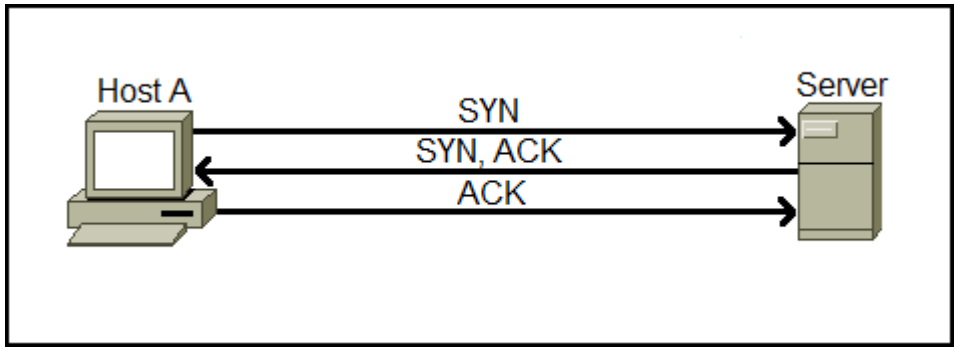
Since it is Persistent connection TCP connection is not required again and again.
Pipe-lining means in one packet only images which can fit, can be send.
In Pipe-lining connection we can send all images in 1RTT.

Total time = 2 RTT + 1 RTT = 3RTT

HTTP handshake

TCP three-way handshake

- Since TCP is a connection-oriented protocol, a connection needs to be established before two devices can communicate. TCP uses a process called three-way handshake to negotiate the sequence and acknowledgment fields and start the session. Here is a graphical representation of the process:

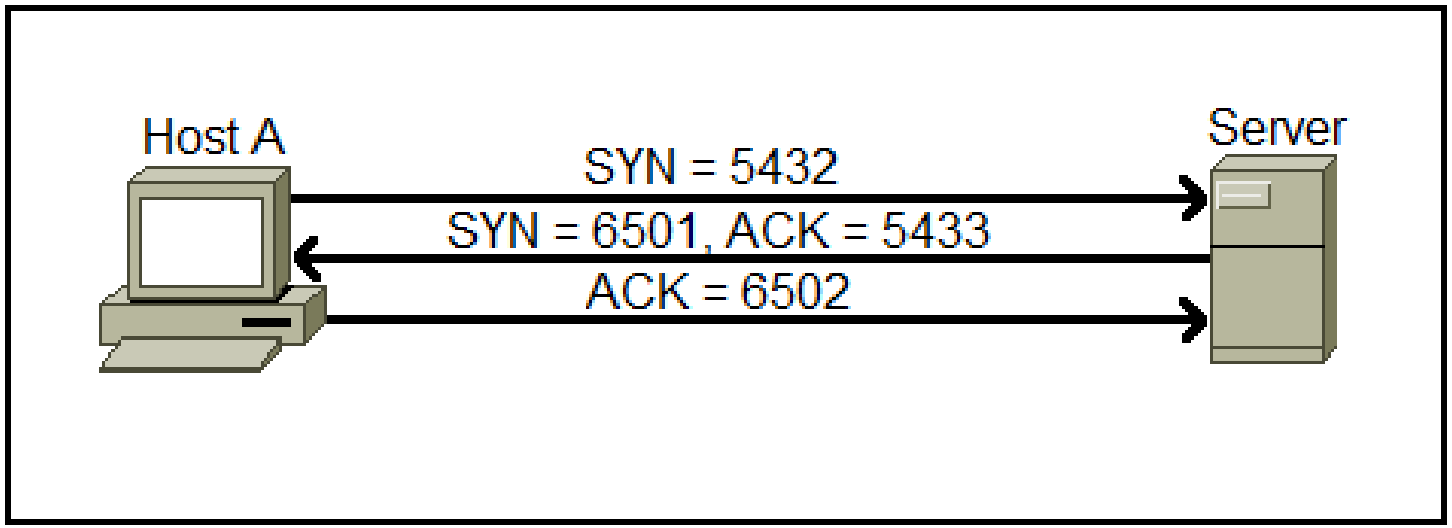


As the name implies, the three way handshake process consists of three steps:

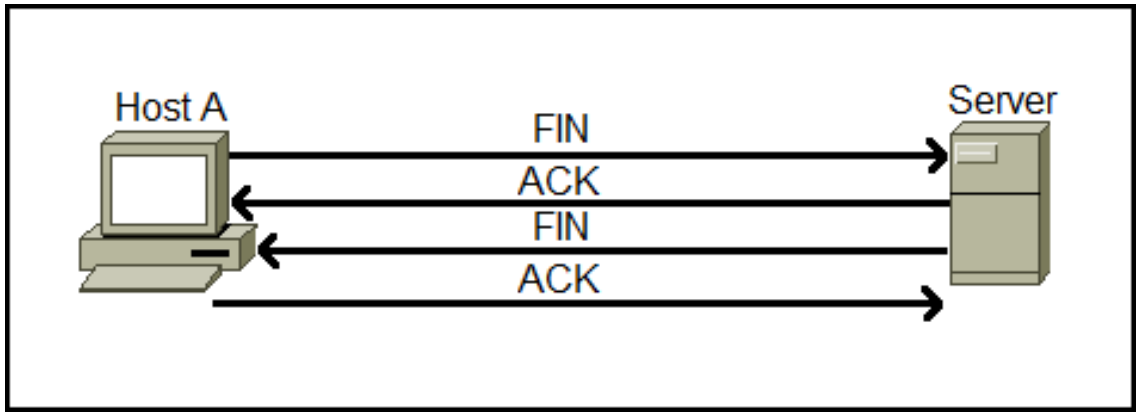
- Host A initiates the connection by sending the TCP SYN packet to the destination host. The packet contains the random sequence number (e.g. **5432**) which marks the beginning of the sequence numbers for data that the Host A will transmit.

- The Server receives the packet and responds with its own sequence number. The response also includes the acknowledgment number, which is Host A's sequence number incremented by 1 (in our case, that would be **5433**).

- Host A acknowledges the response of the Server by sending the acknowledgment number, which is the Server's sequence number incremented by 1.


Host A
Server
SYN = 5432
SYN = 6501, ACK = 5433
ACK = 6502

After the data transmission process is finished, TCP will terminate the connection between two endpoints. This four-step process is illustrated below:

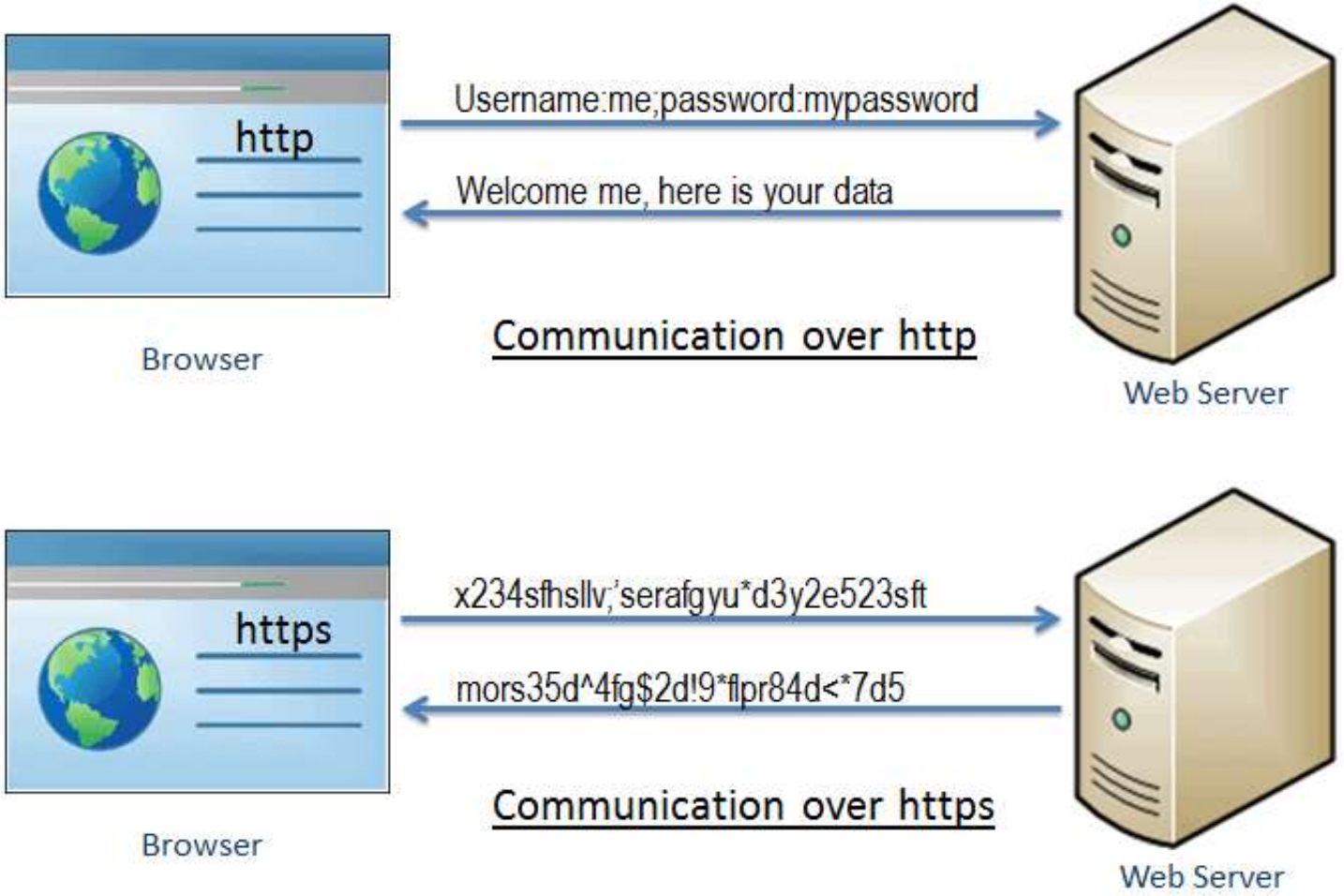


- The client application that wants to close the connection sends a TCP segment with the FIN (Finished) flag set to 1.
- The server receives the TCP segment and acknowledges it with the ACK segment.
- Server sends its own TCP segment with the FIN flag set to 1 to the client in order to terminate the connection.
- The client acknowledges the server's FIN segment and closes the connection.

HTTPS

HTTPS

- HTTPS stands for Hyper Text Transfer Protocol Secure. It is a protocol for securing the communication between two systems e.g. the browser and the web server.



The following figure illustrates the difference between communication over http and https

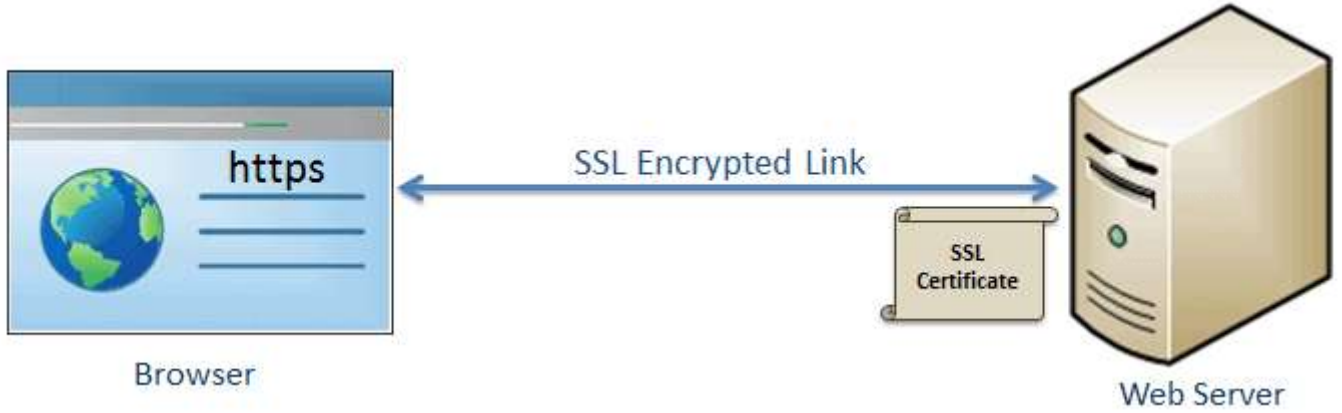
- As you can see in the above figure, http transfers data between the browser and the web server in the hypertext format, whereas https transfers data in the encrypted format.

- Thus, https prevents hackers from reading and modifying the data during the transfer between the browser and the web server. Even if hackers manage to intercept the communication, they will not be able to use it because the message is encrypted.

- HTTPS established an encrypted link between the browser and the web server using the Secure Socket Layer (SSL) or Transport Layer Security (TLS) protocols. TLS is the new version of SSL.

Secure Socket Layer (SSL)

- SSL is the standard security technology for establishing an encrypted link between the two systems. These can be browser to server, server to server or client to server. Basically, SSL ensures that the data transfer between the two systems remains encrypted and private.
- The https is essentially http over SSL. SSL establishes an encrypted link using an SSL certificate which is also known as a digital certificate.



http vs https

http	https
Transfers data in hypertext (structured text) format	Transfers data in encrypted format
Uses port 80 by default	Uses port 443 by default
Not secure	Secured using SSL technology
Starts with http://	Starts with https://

Advantage of https

- **Secure Communication:** https makes a secure connection by establishing an encrypted link between the browser and the server or any two systems.

- **Data Integrity:** https provides data integrity by encrypting the data and so, even if hackers manage to trap the data, they cannot read or modify it.

- **Privacy and Security:** https protects the privacy and security of website users by preventing hackers to passively listen to communication between the browser and the server

- **Faster Performance:** https increases the speed of data transfer compared to http by encrypting and reducing the size of the data.

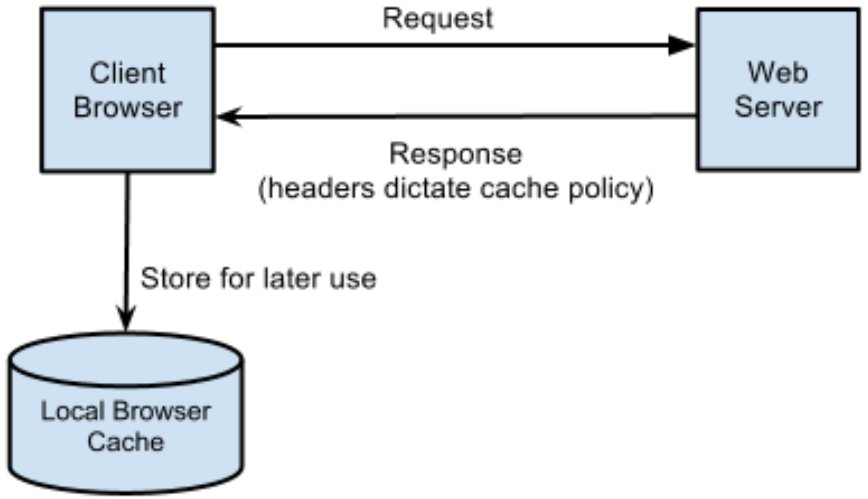
- **SEO:** Use of https increases SEO ranking. In Google Chrome, Google shows the **Not Secure** label in the browser if users' data is collected over http.

- **Future:** https represents the future of the web by making internet safe for users and website owners

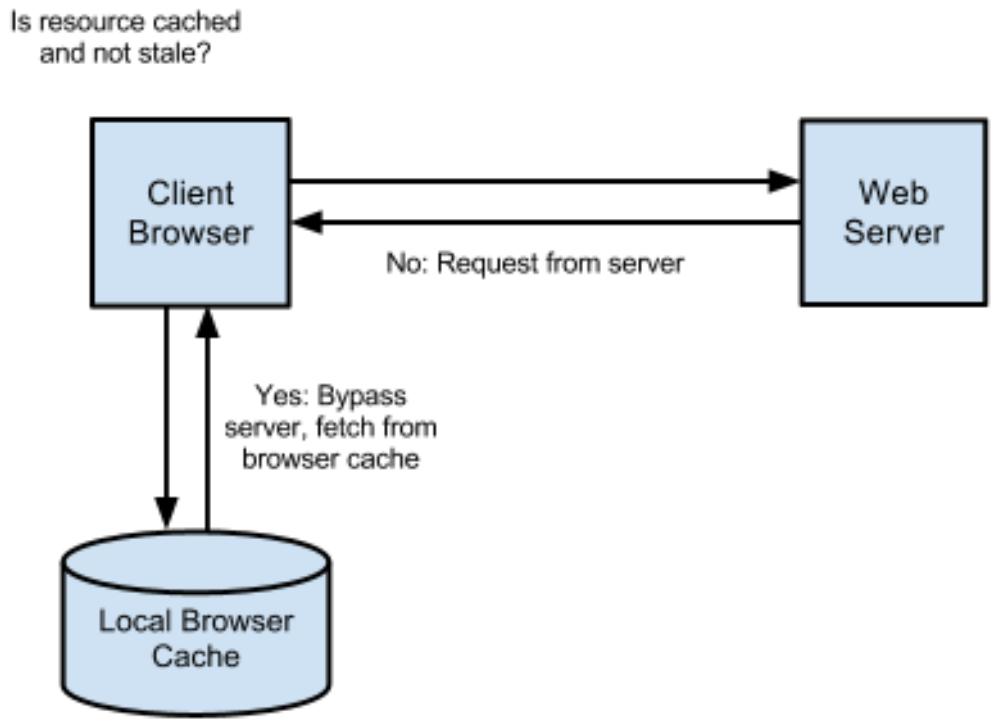
HTTP Cache

HTTP Cache

- HTTP caching occurs when the browser stores local copies of web resources for faster retrieval the next time the resource is required. As your application serves resources it can attach cache headers to the response specifying the desired cache behaviour.



- When an item is fully cached, the browser may choose to not contact the server at all and simply use its own cached copy:



HTTP cache headers

There are two primary cache headers

- Cache-Control and
- Expires

Cache-Control

- The Cache-Control header is the most important header to set as it effectively ‘switches on’ caching in the browser.
- With this header in place, and set with a value that enables caching, the browser will cache the file for as long as specified.
- Without this header the browser will re-request the file on each subsequent request.

Without the cache-control header set, no other caching headers will yield any results.

- **Public resources** can be cached not only by the end-user's browser but also by any intermediate proxies that may be serving many other users as well.

Cache-Control: public

- **Private resources** are bypassed by intermediate proxies and can only be cached by the end-client.

Cache-Control:private

- The value of the Cache-Control header is a composite one, indicating whether the resource is public or private while also indicating the maximum amount of time it can be cached before considered stale.
- The max-age value sets a timespan for how long to cache the resource (in seconds).

Cache-Control:public, max-age=31536000

- While the **Cache-Control** header turns on client-side caching and sets the max-age of a resource the **Expires** header is used to specify a specific point in time the resource is no longer valid.

Expires

- When accompanying the Cache-Control header, Expires simply sets a date from which the cached resource should no longer be considered valid. From this date forward the browser will request a fresh copy of the resource. Until then, the browsers local cached copy will be used:

If both Expires and max-age are set max-age will take precedence

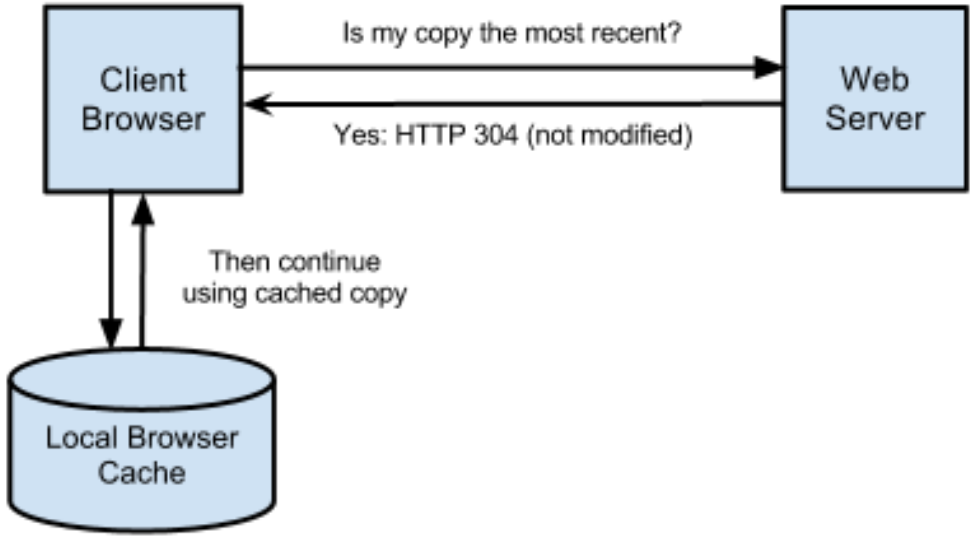
Cache-Control: public

Expires: Mon, 25 Jun 2012 21:31:12 GMT

- While Cache-Control and Expires tells the browser *when* to next retrieve the resource from the network a few additional headers specify *how* to retrieve the resource from the network. These types of requests are known as **conditional requests**.

Conditional Request

- Conditional requests are those where the browser can ask the server if it has an updated copy of the resource. The browser will send some information about the cached resource it holds and the server will determine whether updated content should be returned or the browser's copy is the most recent. In the case of the latter an HTTP status of 304 (not modified) is returned.



Time-based

- A time-based conditional request ensures that only if the requested resource has changed since the browser's copy was cached will the contents be transferred. If the cached copy is the most up-to-date then the server returns the 304 response code.
- To enable conditional requests the application specifies the last modified time of a resource via the **Last-Modified** response header.

Cache-Control:public, max-age=31536000

Last-Modified: Mon, 03 Jan 2011 17:45:57 GMT

- The next time the browser requests this resource it will only ask for the contents of the resource if they're unchanged since this date using the **If-Modified-Since** *request* header

If-Modified-Since: Mon, 03 Jan 2011 17:45:57 GMT

- If the resource hasn't changed since **Mon, 03 Jan 2011 17:45:57 GMT** the server will return with an empty body with the **304** response code.

Content based

- The **ETag** (or Entity Tag) works in a similar way to the **Last-Modified** header except its value is a digest of the resources contents (for instance, an MD5 hash). This allows the server to identify if the cached contents of the resource are different to the most recent version.

This tag is useful when for when the last modified date is difficult to determine.

Cache-Control:public, max-age=31536000

ETag: "15f0fff99ed5aae4edffdd6496d7131f"

- On subsequent browser requests the **If-None-Match** *request* header is sent with the ETag value of the last requested version of the resource.

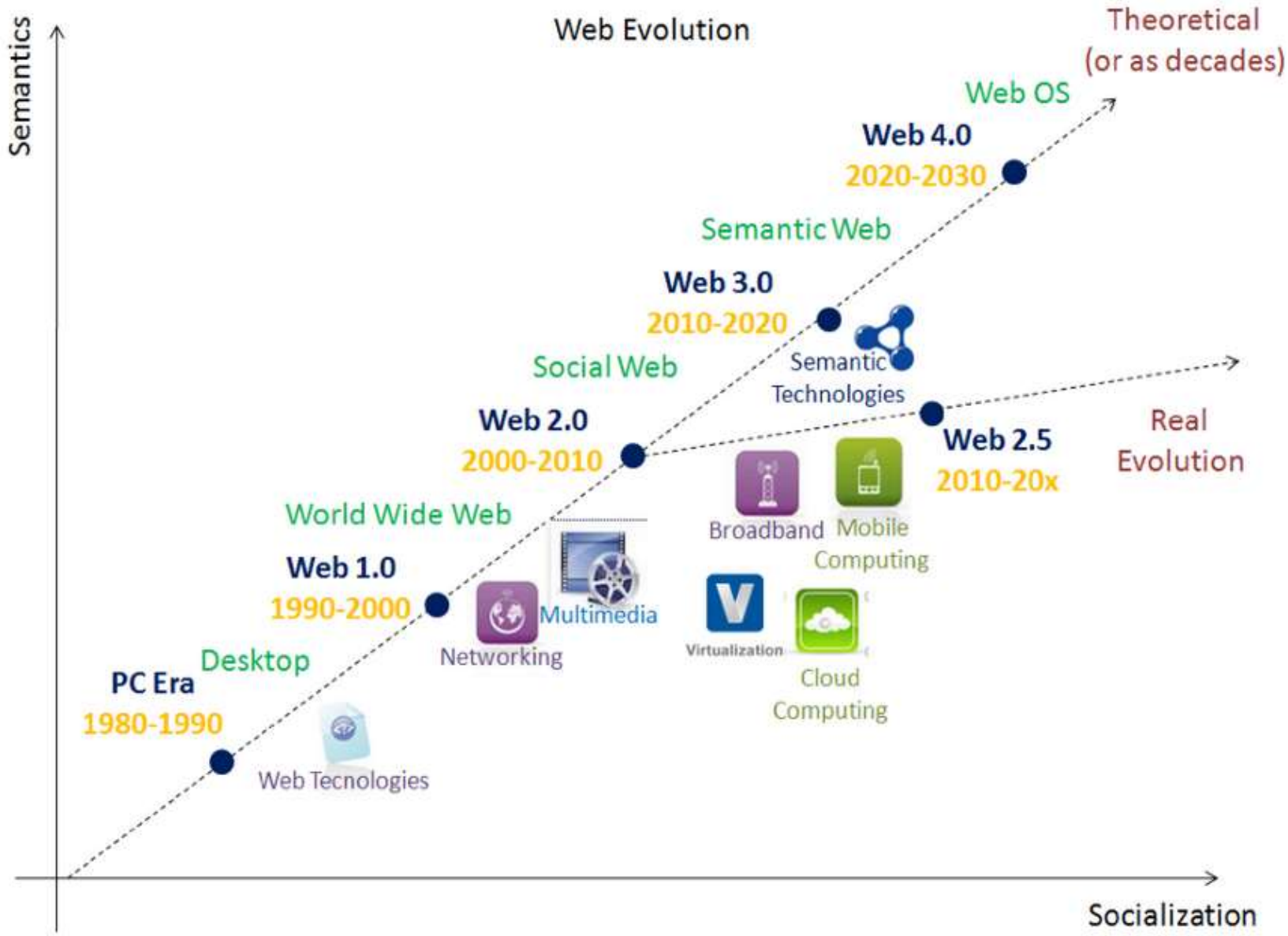
If-None-Match: "15f0fff99ed5aae4edffdd6496d7131f"

- As with the **If-Modified-Since** header, if the current version has the same ETag value, indicating its value is the same as the browser's cached copy, then an HTTP status of 304 is returned.

Evolution of Web

World Wide Web is the primary tool used by billions of people to share, read, and write information to interact with other people via internet.

World Wide Web made much progress since its advent, here we are sharing a brief idea of the evolution happened in web from web 1.0 to 2.0 and to web 3.0.


Web Evolution
Semantics
Socialization
Theoretical
(or as decades)
Web OS
Web 4.0
2020-2030
Semantic Web
Web 3.0
2010-2020
Social Web
Semantic
Technologies
Web 2.0
2000-2010
Real
Evolution
Web 2.5
2010-20x
World Wide Web
Broadband
Mobile
Computing
Web 1.0
1990-2000
Multimedia
Networking
Virtualization
Cloud
Computing
Desktop
PC Era
1980-1990
Web Tecnologies

WEB 1.0

- The first version of web **Web 1.0** also referred as Syntactic web or read only web is the era(1990–2000) where the role of a user is limited to reading information provided by the content producers.

- There is no option given for user or consumer to communicate back the information to the content producers.

- Example of Web 1.0 are static web sites and personal sites.

- In Web 1.0 advertisements on websites while surfing the internet is banned. Also, in Web 1.0, Ofoto is an online digital photography website, on which user could store, share, view and print digital pictures.

- Web 1.0 is a content delivery network (CDN) which enables to showcase the piece of information on the websites. It can be used as personal websites. It costs to user as per pages viewed. It has directories which enable user to retrieve a particular piece of information.

Four design essentials of a Web 1.0 site include:

- Static pages.
- Content is served from the server’s file-system.
- Pages built using Server Side Includes or Common Gateway Interface (CGI).
- Frames and Tables used to position and align the elements on a page.

WEB 2.0

- The **Web 2.0** also referred as Social Web or read-write web is the era(2000–2010 and continues even now) which facilitates interaction between web users and sites which intern allows users to communicate with other users.
- In this era every user can be a content producers and content is distributed and shared between sites.
- Some of the famous Web 2.0 applications are Facebook, Youtube, Flickr, Twitter etc.,
- The web technologies like HTML5, CSS3 and Javascript frameworks like ReactJs, AngularJs, VueJs etc., enables startups to innovate new ideas which enables users to contribute more in this Social Web.
- Web 2.0 is build around the users, producer just need build a way to enable and engage them.

Five major features of Web 2.0 – Free sorting of information, permits users to retrieve and classify the information collectively.

- Dynamic content that is responsive to user input.
- Information flows between site owner and site users by means of evaluation & online commenting.
- Developed APIs to allow self-usage, such as by a software application.
- Web access leads to concern different, from the traditional Internet user base to a wider variety of users.

Usage of Web 2.0 –

The social Web contains a number of online tools and platforms where people share their perspectives, opinions, thoughts and experiences. Web 2.0 applications tend to interact much more with the end user. As such, the end user is not only a user of the application but also a participant by these 8 tools mentioned below:

- Podcasting
- Blogging
- Tagging
- Curating with RSS
- Social bookmarking
- Social networking
- Social media
- Web content voting

WEB 3.0

The **Web 3.0** also referred as Semantic Web or read-write-execute is the era(2010 and above) which refers to the future of web. In this era computers can interpret information like humans via Artificial Intelligence and Machine Learning. Which help to intelligently generate and distribute useful content tailored to a particular need of a user.

- What if computers can understand meaning behind information
- What if they can learn “what we are interested in”
- Then they can help us find what we want
- It can recognise People, Place, Events, Companies, Product, Movies etc.,
- It can understand the relationship between things

Some of the examples of web 3.0 are Apple’s Siri, Googles Cloud API, Wolfram Alpha.

Below are 5 main features that can help us define Web 3.0

- **Semantic Web**

  The succeeding evolution of the Web involves the Semantic Web. The semantic web improves web technologies in demand to create, share and connect content through search and analysis based on the capability to comprehend the meaning of words, rather than on keywords or numbers.

- **Artificial Intelligence**

  Combining this capability with natural language processing, in Web 3.0, computers can distinguish information like humans in order to provide faster and more relevant results. They become more intelligent to fulfil the requirements of users.

- **3D Graphics**

  The three-dimensional design is being used widely in websites and services in Web 3.0. Museum guides, computer games, ecommerce, geospatial contexts, etc. are all examples that use 3D graphics.

- **Connectivity**

  With Web 3.0, information is more connected thanks to semantic metadata. As a result, the user experience evolves to another level of connectivity that leverages all the available information.

- **Ubiquity**

  Content is accessible by multiple applications, every device is connected to the web, the services can be used everywhere.

Difference between Web 1.0, Web 2.0 and Web 3.0

Web 1.0	Web 2.0	Web 3.0
Mostly Read-Only	Wildly Read-Write	Portable and Personal
Company Focus	Community Focus	Individual Focus
Home Pages	Blogs / Wikis	Live-streams / Waves
Owning Content	Sharing Content	Consolidating Content
Web Forms	Web Applications	Smart Applications
Directories	Tagging	User Behaviour
Page Views	Cost Per Click	User Engagement
Banner Advertising	Interactive Advertising	Behavioural Advertising
Britannica Online	Wikipedia	The Semantic Web
HTML/Portals	XML / RSS	RDF / RDFS / OWL

Chapter -6

Web IR: Information Retrieval on the Web

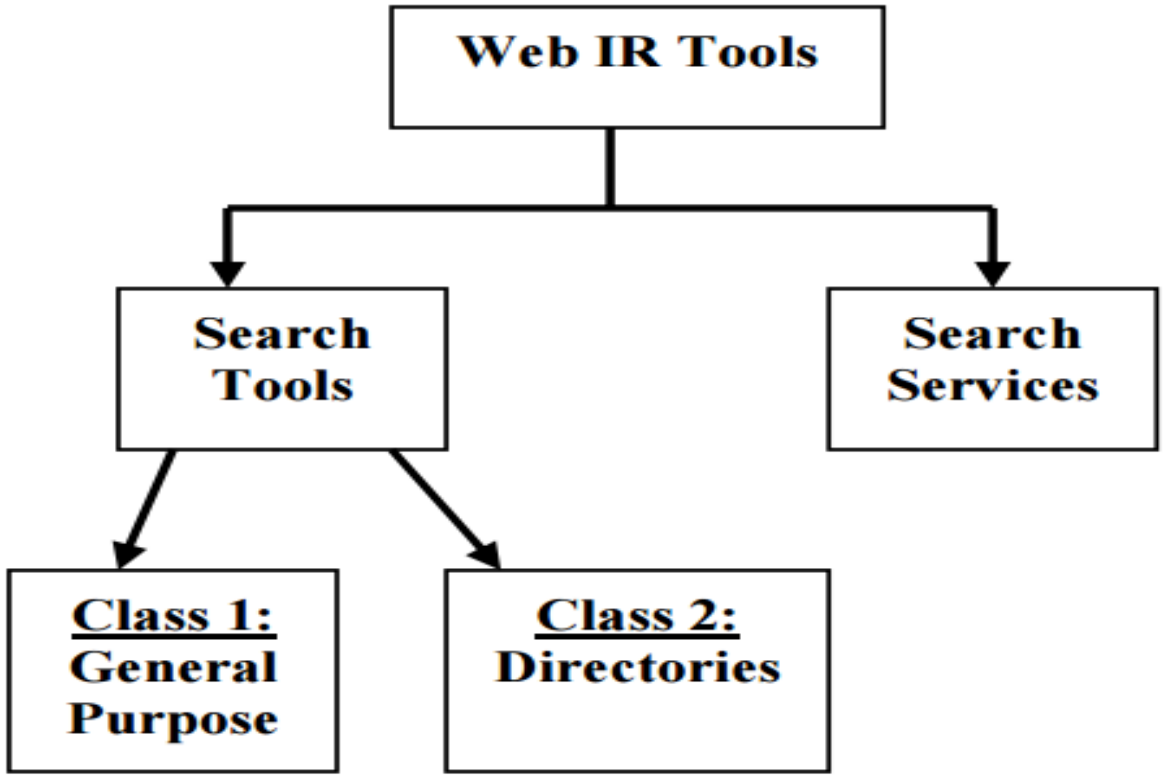
Web IR

Web IR can be defined as the application of theories and methodologies from IR to the World Wide Web. It is concerned with addressing the technological challenges facing Information Retrieval (IR) in the setting of WWW.

Web IR is different from classical IR for two kinds of reasons: concepts and technologies. The characteristics of Web make the task of retrieving information from it quite different from the Pre- Web (traditional) information retrieval.

Web IR Tools

Categories of Web IR tools:



Search Tools

The Search tools employ robots for indexing Web documents. They feature a user interface for specifying queries and browsing the results. At the heart of a search tool is the search engine, which is responsible for searching the index to retrieve documents relevant to a user query.

Search tools can be distinguished into two categories on the transparency of the index to the user.

- ✓Class1 search tools
- ✓Class2 search tools

Class1 search tools: General Purpose Search Engine: These tools completely hide the organization and content of the index from the user. Example: AltaVista: www.altavista.com; Excite: www.excite.com; Google: www.google.com; Lycos: www.lycos.com; Hotbot: www.hotbot.com

Class 2 search tools: Subject Directories: These feature a hierarchically organized subject catalog or directory of the Web, which is visible to users as they browse and search. Example: Yahoo!: www.search.yahoo.com; WWW Virtual Library: http://vlib.org/; Galaxy: https://use**galaxy**.org/

Search Services

The Search services provide users a layer of abstraction over several search tools and databases and aim at simplifying the Web search. Search services broadcast user queries to several search engines and various other information sources simultaneously. Then they merge the results submitted by these sources, check for duplicates, and present them to the user as an HTML page with clickable URLs.

Example: MetaCrawler: http://www.metacrawler.com/; Dogpile: http://www.dogpile.com/

Web IR Architecture (Search Engine Architecture)

The architecture of a search engine determined by two requirements – effectiveness (quality of results) and efficiency (response time and throughput) and has two parts namely: The indexing process and the query process.

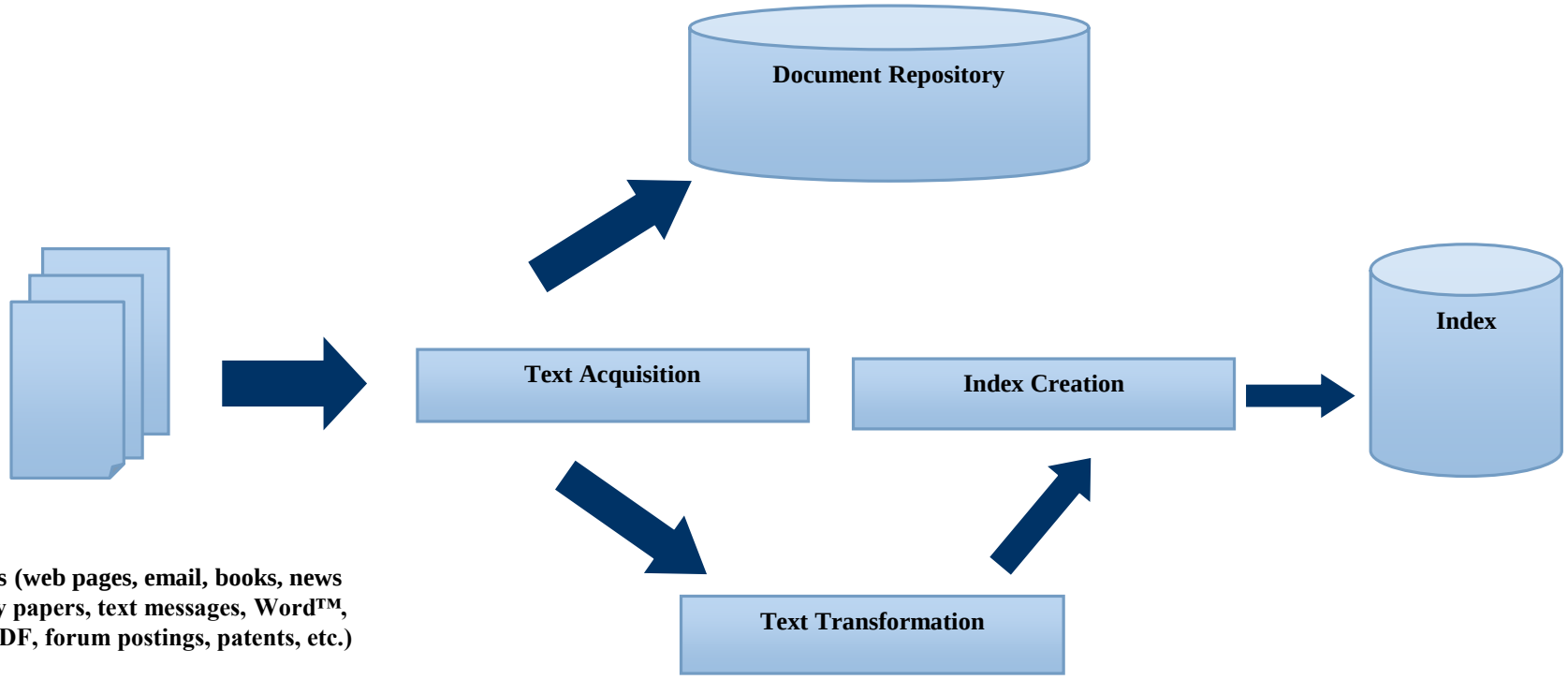
The Indexing Process

The Indexing process distills information contained within corpus documents into a format which is amenable to quick access by the query processor. Typically, this involves extracting document features by breaking down documents into their constituent terms, extracting statistics relating to term presence within the documents and corpus, and calculating any query-independent evidence. Once the indices are built, the system is ready to process queries.

It consists of three steps:

- ✓Text Acquisition
- ✓Text Transformation
- ✓Index creation

The Indexing Process



Text Acquisition

A Crawler is usually responsible for gathering the documents and storing them in document repository. It identifies and acquires documents for search engine.

Web crawlers follow links to find documents which must efficiently find huge numbers of web pages (coverage) and keep them up-to-date (freshness).

The web crawler is implied to know that a word contained in headings, meta data and the first few sentences are likely to be more important in the context of the page, and that keywords in prime locations suggest that the page is really ‘about’ those keywords.

The text acquisition identifies and stores the documents for indexing by crawling the Web, then converting the gathered information into a consistent format & finally storing the findings in a document repository.

Text Transformation

Once the text is acquired, the next step is to transform the captured documents into index terms or features. This is a kind of preprocessing step which involves parsing, stop-word removal, stemming, link analysis & information extraction as the sub-steps.

Parser: It is the processing of the sequence of text tokens in the document to recognize structural elements. For e.g., titles, links, headings, etc.

Stopping: Commonly occurring words are unlikely to give useful information and may be removed from the vocabulary to speed processing. Stop-word removal or Stopping is a process that removes the common words like “and”, “or”, “the”, “in”.

Stemming: Stemming is the process of removing suffixes from words to get the common origin. It is a group words that are derived from a common stem. For e.g., “engineer”, “engineers”, “engineering”, “engineered”.

Link Analysis: It makes use of links and anchor text in web pages and identifies popularity and community information, for example, PageRank.

Information Extraction: This process identifies the classes of index terms that are important for some applications. For e.g., named entity recognizers identify classes such as people, locations, companies, dates, etc.

Classifier: It identifies the class-related metadata for documents i.e.: It assigns labels to documents, for example, topics, reading levels, sentiment, genre. The use of classifier depends on the application.

Index Creation

The document statistics such counts of index term occurrences, positions in the documents where the index terms occurred, counts of occurrences over groups of documents, lengths of documents in terms of the number of tokens and other features mostly used in ranking algorithms are gathered. Actual data depends on the retrieval model and the associated ranking method.

The index terms weights are then computed (for example, tf.idf weight which is a combination of term frequency in document and inverse document frequency in the collection).

Most indices use variants of inverted files.

An inverted file is a list of sorted words. Each word has pointers to the related pages. It is referred to as “Inverted” because documents are associated with words, rather than words with documents. A logical view of the text is indexed. Each pointer associates a short description about the page that the pointer points to.

This is the core of the indexing process which tends to convert document-term information to term-document for indexing but it is difficult to do that for very large numbers of documents. The format of inverted file is designed for fast query processing that must also handle updates.

Inverted index allows quick lookup of document ids with a particular word. The inverted index are built as follows, for each index term is associated with an *inverted list*

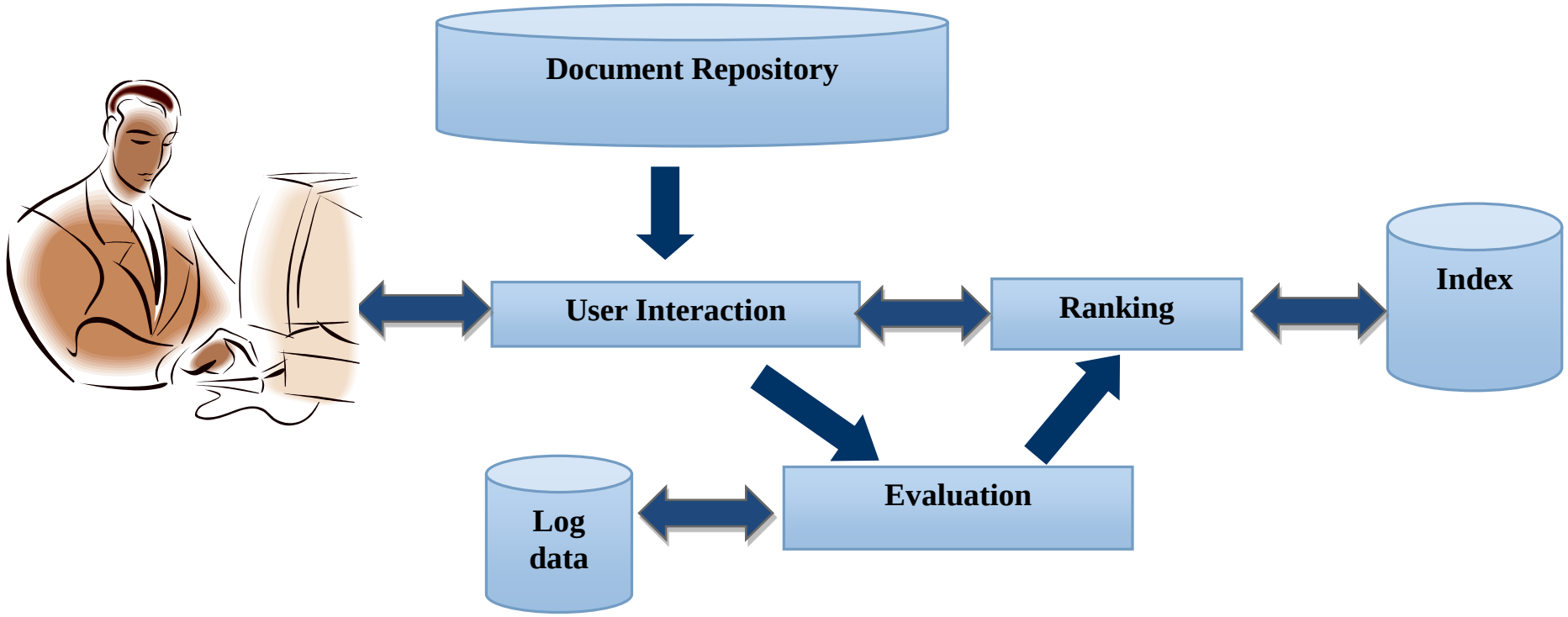
- Contains lists of documents, or lists of word occurrences in documents, and other information
- Each entry is called a *posting*
- The part of the posting that refers to a specific document or location is called a *pointer*
- Each document in the collection is given a unique number
- Lists are usually *document-ordered* (sorted by document number)

The Query Process

Query processing takes the user's query, and depending on the application, the context, and other inputs, builds a better query automatically, and submits the enhanced query to the search engine on the user's behalf and displays the ranked results.

Thus, a query process comprises of a user interaction module which supports creation and the refinement of query and displays the results and a ranking module which uses the query and indexes (generated during the indexing process) to generate a ranked list of documents.

The Query Process



User interacts with the system through an interface where he inputs the query.

Query transformation is then employed to improve the initial query, both before and after the initial search. It may use a variety of techniques, such as, the spell checker, query suggestion providing alternatives to original query.

The query expansion approaches attempt to expand the original search query by adding further, new or related terms. These additional terms are inserted to an existing query either by the user (Interactive query expansion, IQE), or by the retrieval system (Automatic query expansion, AQE) and intend to increase the accuracy of the search.

The fundamental challenge of a search engine is to rank pages that match the input query and returns an ordered list. The search engines rank individual web-pages of a website, not the entire site. There many variations of ranking algorithms and retrieval models.

Search engines use two different kinds of ranking factors: query-dependent factors and query-independent factors.

Query-dependent are all ranking factors that are specific to a given query. These include measures such as word documents frequency, the position of the query terms within the document or the inverted document frequency, which are all measures that are used in traditional Information Retrieval.

Query-independent factors are attached to the documents, regardless of a given query and consider measures such as an emphasis on anchor text, the language of the document in relation to the language of the query or the measuring of the “geographical distance between the user and the document”. They are used to determine the quality of a given document such that the search engines should provide the user with the highest possible quality and should omit low-quality documents.

The Web Search can be categorized into two phases, namely the Offline phase which includes the ‘Crawling’ & ‘Indexing’ Components; and the Online phase which includes the ‘Querying’ & ‘Ranking’ components of the Web IR system.

The results may be evaluated to monitor and measure the retrieval effectiveness and efficiency. This is generally done offline and involves logging of user queries and their interaction as it can be crucial to improve search effectiveness and efficiency. The query logs and click-through data is used for query suggestion, spell checking, query caching, ranking, advertising search, and other components.

A ranking analysis can be done to measure and tune the ranking effectiveness whereas a performance analysis is carried to measure and tune the system efficiency.

Web IR Performance Metrics

System evaluation in Web IR revolves around the notion of *relevant* and *not relevant* documents. This implies that with respect to a given query, a document is given a binary classification as either relevant or not relevant.

To measure information retrieval effectiveness, we need:

A test collection of documents

A benchmark suite of queries

A binary assessment of either relevant or not relevant for each query-document pair.

Confusion Matrix

In a binary decision problem, a classifier labels examples as either positive or negative. The decision made by the classifier can be represented in a structure known as a confusion matrix or contingency table.

The confusion matrix has four categories:

- ✓***True positives (TP):*** correctly labeled as positives.
- ✓***False positives (FP):*** negative examples incorrectly labeled as positive
- ✓***True negatives (TN):*** correspond to negatives correctly labeled as negative
- ✓***False negatives (FN):*** positive examples incorrectly labeled as negative

Confusion Matrix

	Not Retrieved **(Predicted NO)**	**Retrieved** **(Predicted YES)**
Irrelevant **(Actual NO)**	True Negatives (TN) (irrelevant& not retrieved)	False Positives (FP) (irrelevant & retrieved)
Relevant **(Actual YES)**	False Negatives (FN) (relevant & not retrieved)	True Positives (TP) (relevant & retrieved)

Precision and Recall

In an Information Retrieval scenario, the most common evaluation is retrieval effectiveness and the effect of indexing exhaustivity and term specificity on retrieval effectiveness can be explained by two widely accepted measures Precision & Recall.

Precision is defined as the number of relevant documents retrieved by a search divided by the total number of documents retrieved by that search,

$$\boldsymbol{P = \frac{Total\ number\ of\ relevant\ retrieved\ documents}{Total\ number\ of\ retrieved\ documents} \times 100}$$

That is, $\boldsymbol{P = \frac{TP}{(TP+FP)}}$

A perfect Precision score of 1.0 means that every result retrieved by a search was relevant

Recall is defined as the number of relevant documents retrieved by a search divided by the total number of existing relevant documents.

$$R = \frac{\textit{Total number of relevant retrieved documents}}{\textit{Total number of relevant documents}} \times 100$$

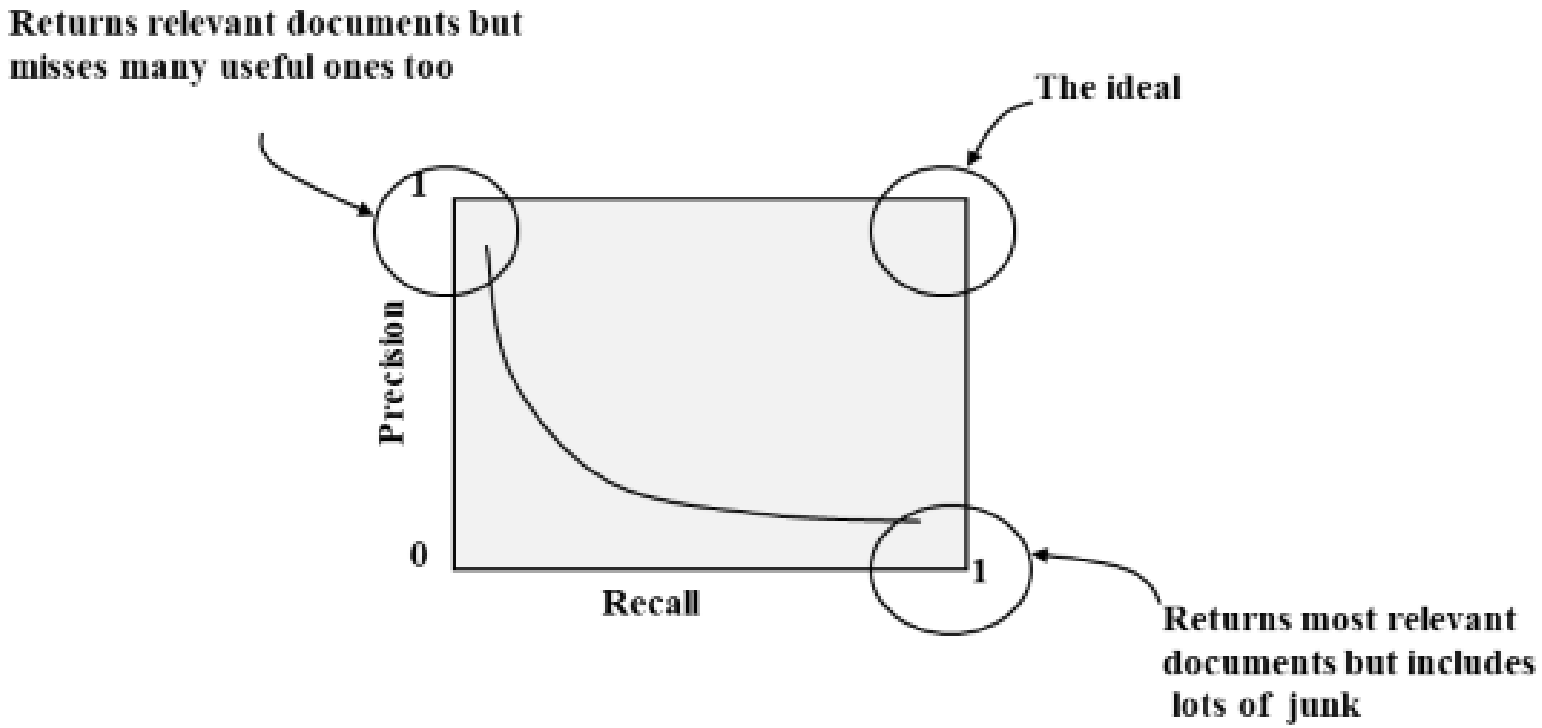
That is, $R = \frac{TP}{(TP+FN)}$

A perfect Recall score of 1.0 means that all relevant documents were retrieved by the search.

The Precision & Recall are inversely related. You can always get a recall of 1 (but very low precision) by retrieving all documents for all queries!

Recall is a non-decreasing function of the number of documents retrieved. On the other hand, precision usually decreases as the number of documents retrieved is increased.

The Recall-Precision Tradeoff Graph



Some other measures which evaluates the performance are as follows:

Accuracy is the proportion of the total number of predictions that were correct.

$$\boldsymbol{AC = \frac{TP + TN}{n}}$$

Misclassification Rate or Error-rate

$$\boldsymbol{Error = \frac{(FP + FN)}{n}}$$

or

$$\boldsymbol{Error = 1 - AC}$$

False Positive Rate (False Alarm Rate or Fall-out) is defined as the probability to find an irrelevant among the retrieved documents. It measures of how quickly precision drops as recall is increased.

$$\boldsymbol{Fallout = \frac{FP}{(TN + FP)}}$$

Specificity is a statistical measure of how well a binary classification test correctly identifies the negative cases

$$\boldsymbol{Specificity} = \frac{\boldsymbol{TN}}{(\boldsymbol{TN} + \boldsymbol{FP})} = \mathbf{1} - \boldsymbol{Fallout}$$

F-measure (in information retrieval): can be used as a single measure of performance. The F-measure is the harmonic mean of precision and recall. It is a weighted average of the true positive rate (recall) and precision.

$$\boldsymbol{F - measure} = \frac{\mathbf{2} * \boldsymbol{Precision} * \boldsymbol{Recall}}{\boldsymbol{Precision} + \boldsymbol{Recall}}$$

Web IR Models

The model of Web IR can be defined as a set of premises and an algorithm for ranking documents with regard to a user query.

A Web IR model is a quadruple [D, Q, F, R (q_i,d_j)] where D is a set of logical views of documents, Q is a set of user queries, F is a framework for modeling documents and queries, and R(q_i,d_j) is a ranking function which associates a numeric ranking to the query q_i and the document d_j.

The model is characterized by four parameters:

Representations for documents and queries, which define the model

Matching strategies for assessing the relevance of documents to a user query, which involves learning parameters from query.

Methods for ranking query output, and

Mechanisms for acquiring user-relevance feedback.

In general, retrieval models can be categorized into *"exact match"* or *"best match"* models.

An ***"exact match"*** model is where the query specifies precise retrieval criteria and every document either matches or fails to match the query. The result is an unordered set of documents with pure exact match. These models work well when we know exactly (or roughly) what the collection contains and what we're looking for, that is basically we are looking for precise documents using structured queries.

The ***"best match"*** or "rank-based" models the query describes "good" or "best" matching document and every document matches query to some degree. The result is ranked list of documents. These are significantly more effective than the exact match model and supports textual queries but suffer from problems related to natural language understanding. The efficiency is also less as compared to the exact match models owing to their incapability to reject documents early.

Standard Boolean Model

Standard Boolean model is based on Boolean logic and classical set theory where both the documents to be searched and the user's query are conceived as sets of terms.

Retrieval is based on whether or not the documents contain the query terms.

Query represented as a Boolean expression of terms in which terms are combined with the logical operators AND, OR, AND NOT, specifies precise relevance criteria and documents are retrieved iff they satisfy a Boolean expression.

The documents are returned in no particular order and thus are an unranked Boolean model where all terms are equally weighted.

Model:

- Retrieve documents iff they satisfy a Boolean expression
- Query specifies precise relevance criteria
- Documents are returned in no particular order (Unranked Boolean)

Operators:

- Logical operators: AND, OR, AND NOT (Unconstrained NOT not included)
- Distance operators: Proximity
- String matching operators: Wildcard (The wildcard operator search* matches “search”, “searching”, “searched”, etc. This was common before stemming algorithms were introduced)
- Field operators: Date, Author, Title

Algebraic Model

Documents are represented as vectors, matrices or tuples. These are transformed using algebraic operations to a one-dimensional similarity measure.

Implementations include the Vector Space Model and the Generalized Vector Space Model, (Enhanced) Topic-based Vector Space Model, Latent semantic indexing a.k.a. latent semantic analysis.

The strength of this model lies in its simplicity. Relevance feedback can be easily incorporated into it.

Vector Space Model (VSM): The VSM is an algebraic model used for Information Retrieval where the documents are represented through the words that they contain. It represent natural language document in a formal manner by the use of vectors in a multi-dimensional space. Any text object (documents, queries, sentences) can be represented by a term vector and similarity is determined by distance in a vector space.

Model:

- Each document is broken down into a word frequency table. The tables are called vectors and can be stored as arrays.
- A vocabulary is built from all the words in all documents in the system.
- Each document and user query is represented as a vector based against the vocabulary.
- Calculating similarity measure.
- Ranking the documents for relevance.