Web and HTTP

Q

What is a Protocol ?

Protocol

- **Synonymous of Etiquette**

a code of behavior that delineates expectations for social behavior according to contemporary conventional norms within a society, social class, or group.

Communications protocol, a set of rules and regulations that determine how data is transmitted in telecommunications and computer networking

Protocol

- *A **protocol** defines:*

1. *the format and*
2. *the order of messages exchanged between two or more communicating entities, as well as*
3. *the actions taken on the transmission and/or receipt of a message or other event.*

What is a Port?

Port

A port is an **endpoint of communication in an operating system**.

A **process** associates its input or output channels, via an Internet **socket**, with a transport protocol, a port number, and an IP address.

socket: {protocol, local address, local port, remote address, remote port}

This process is known as binding.

HTTP on port 80
- HTTP with SSL (HTTPS) on port 443
- FTP on port 21
- SMTP on port 25
- POP on port 110
- SSH on port 22

Mistranslated into Italian as “Porta” (door

PID	PORT	IP	Protocol
84	21	193.205.196.130	FTP
78	80	193.205.196.130	HTTP
321	8080	193.205.196.130	HTTP
541	25	193.205.196.130	SMTP

Q

What are URI, URL, URN ?

URI, URL, URN

- A **web resource**, or simply **resource**, is any identifiable thing, whether digital, physical, or abstract.

- A **Uniform Resource Identifier (URI)** is a compact sequence of characters that identifies an abstract or physical resource. (RFC 3986)
- A **Uniform Resource Locator (URL**) refers to the subset of URI that identify resources via a ***representation of their primary access mechanism*** (e.g., their network "location")
- **Uniform Resource Name (URN)** refers to the subset of URI that are required to **remain globally unique and persistent even when the resource ceases to exist** or becomes unavailable.

  It is intended to serve as persistent, location-independent, resource identifier (RFC 2141).

RFC (Request for comment)

Memos in the RFC document series contain technical and organizational notes about the Internet.

RFCs cover many aspects of computer networking, including protocols, procedures, programs, and concepts, as well as meeting notes, opinions, and sometimes humour.

https://ietf.org/standards/rfcs/

RFC

[RFC Home] [TEXT|PDF|HTML] [Tracker] [IPR]

PROPOSED STANDARD

Internet Engineering Task Force (IETF)
Request for Comments: 8141
Obsoletes: 2141, 3406
Category: Standards Track
ISSN: 2070-1721

P. Saint-Andre
Filament
J. Klensin
April 2017

Uniform Resource Names (URNs)

Abstract

A Uniform Resource Name (URN) is a Uniform Resource Identifier (URI) that is assigned under the "urn" URI scheme and a particular URN namespace, with the intent that the URN will be a persistent, location-independent resource identifier. With regard to URN syntax, this document defines the canonical syntax for URNs (in a way that is consistent with URI syntax), specifies methods for determining URN-equivalence, and discusses URI conformance. With regard to URN namespaces, this document specifies a method for defining a URN namespace and associating it with a namespace identifier, and it describes procedures for registering namespace identifiers with the Internet Assigned Numbers Authority (IANA). This document obsoletes both RFCs 2141 and 3406.

URI, URL, URN

- Both URL and URN are URI.

- A URN identifies a resource
- A URL provides a method for finding it.

- A URN may be compared to a person's name,
- A URL may be compared to their street address.

- A URN can be associated to many URLs

URN + URL example

- The ISBN system (namespace) uniquely identifies books.
- urn:isbn:0-486-27557-4 cites unambiguously a *specific edition* of Shakespeare's play Romeo and Juliet.
- A typical URL for this book might look like the file path file:///home/username/RomeoAndJuliet.pdf

URI Schemes

http:

https:

ftp:

mailto:<address>[?<header1>=<value1>[&<header2>=<value2>]]

geo:<lat>,<lon>

fax:<phone number>

file:[//host]/path

bitcoin:<address>[?[amount=<size>]…

skype:<username|phonenumber>…

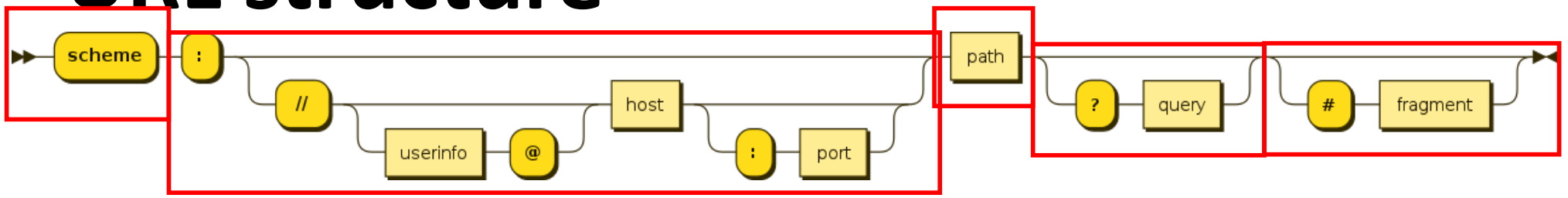
https://en.wikipedia.org/wiki/List_of_URI_schemes

URI structure

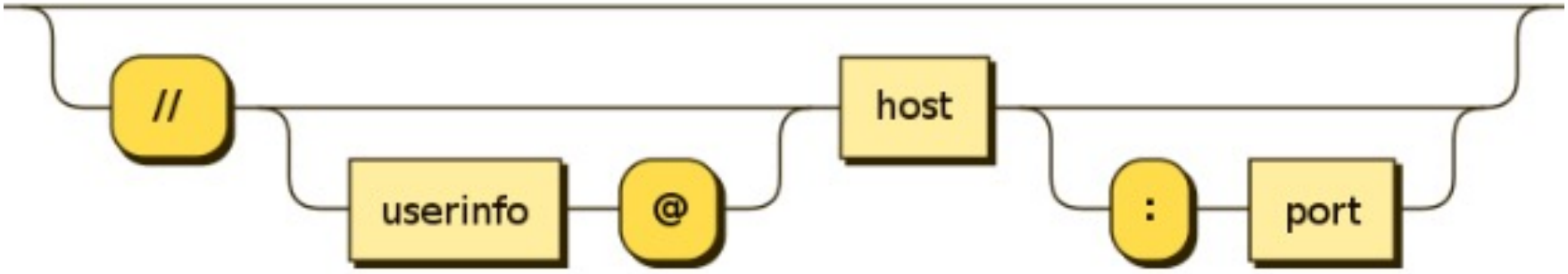
URI = scheme:[//authority]path[?query][#fragment]

authority = [userinfo@]host[:port]

URL structure



https://en.wikipedia.org/wiki/URL

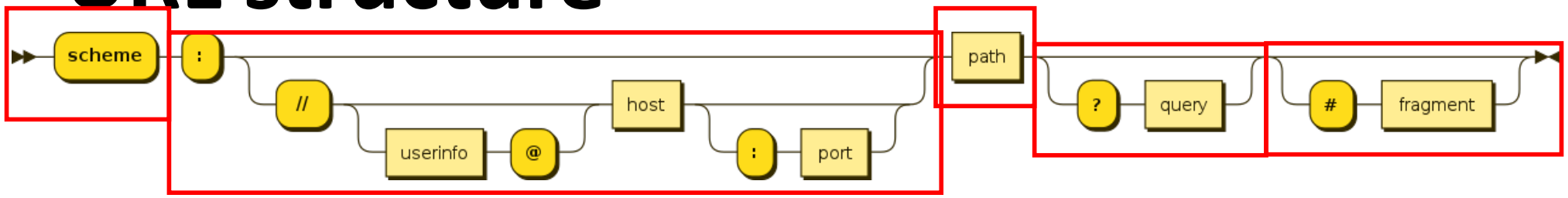


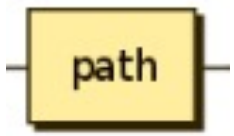
https://en.wikipedia.org/wiki/URL

Watch for phishing!

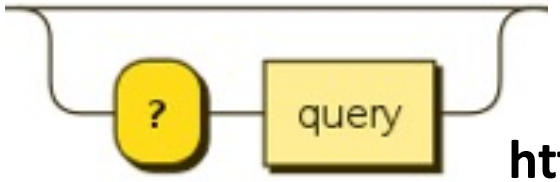
http://www.sitosicuro.it|search=hello@www.phishing.com/

URL structure

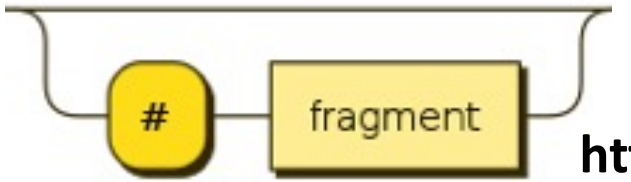




https://en.wikipedia.org/wiki/URL



https://en.wikipedia.org/w/index.php?title=URL&action=edit



https://en.wikipedia.org/wiki/URL#Protocol-relative_URLs

Q

What is MIME Type ?

MIME type

URIs give us info about name and location, but what about their content type?

File extensions are bad…

(see Unix magic numbers

https://www.geeksforgeeks.org/working-with-magic-numbers-in-linux/)

Metadata?

MIME type

Multipurpose Internet Mail Extensions (RFC 2045,2046)

MEDIA_TYPE/SUBTYPE

text -> text/plain, text/html, text/richtext …

image -> image/jpeg, image/png, image/svg+xml…

audio -> audio/basic, audio/ogg, audio/x-wav…

video -> video/mp4, video/ogg…

application -> application/x-apple-diskimage… …

multipart

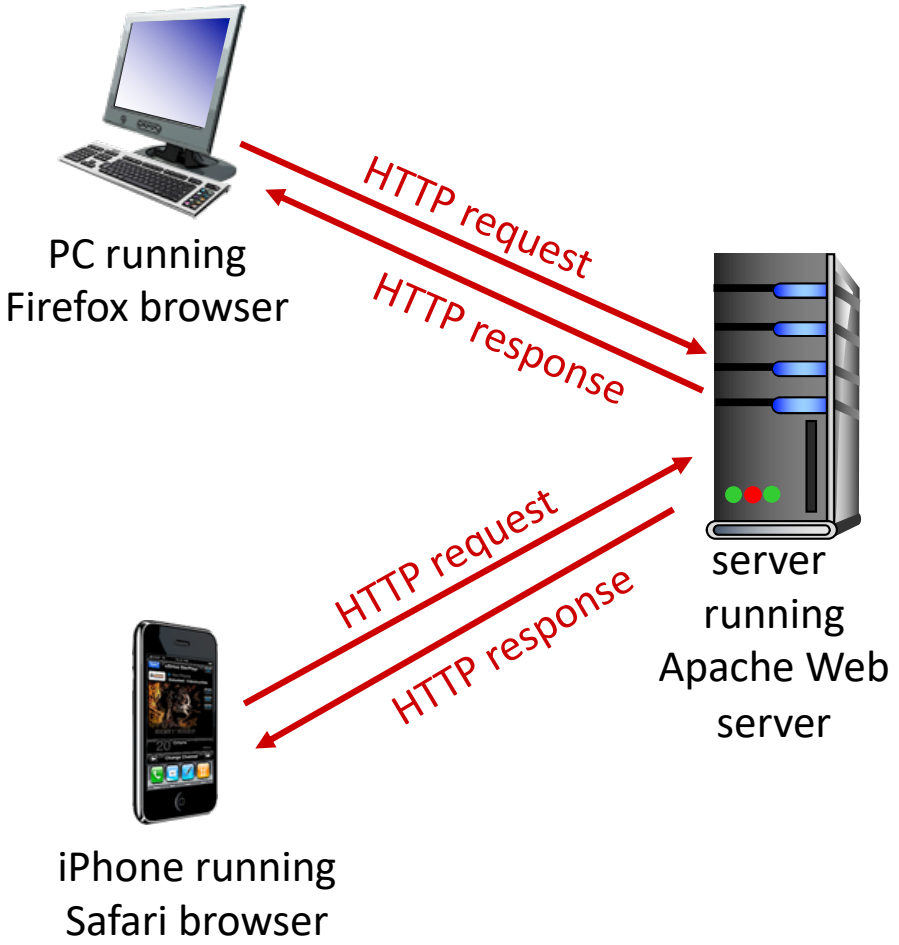
See https://www.freeformatter.com/mime-types-list.html

HTTP: basics and connections

HTTP overview

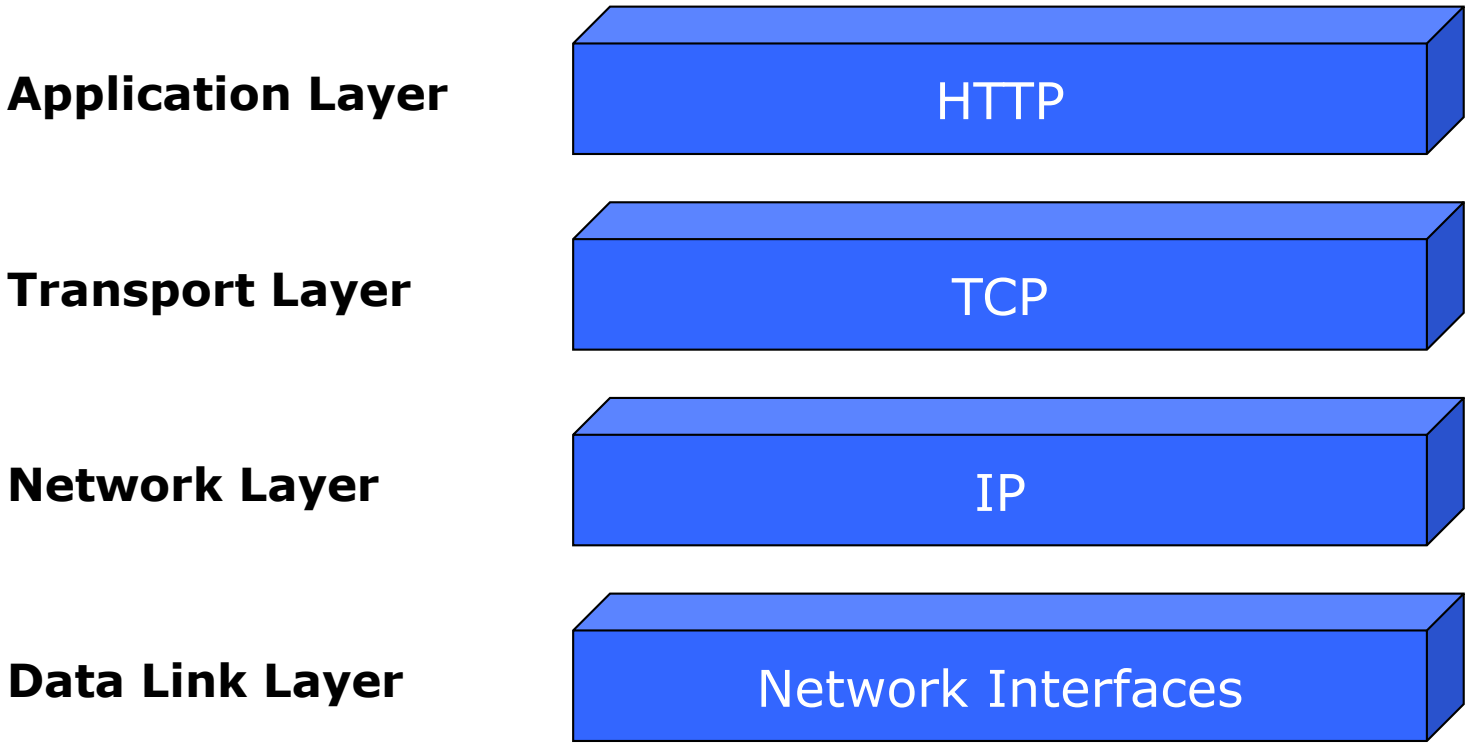
HTTP: hypertext transfer protocol

- Web's application layer protocol
- client/server model
  - client: browser that requests, receives, (using HTTP protocol) and "displays" Web objects
  - server: Web server sends (using HTTP protocol) objects in response to requests



HTTP and TCP/IP

HTTP sits atop the TCP/IP Protocol Stack



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 overview (continued)

HTTP is "stateless"

- server maintains no information about past client requests

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 overview (continued)

For comparison: FTP is "stateful"

- server maintains information about past client requests

e.g.:

- you can issue a "cd" command to move into a (remote) directory
- The next commands (e.g. "ls") will be executed with reference to that directory

Q

Where is HTTP defined ?

HTTP

- Three versions have been used, two are in common use and have been specified:

- The Original HTTP as defined in 1991 as HTTP 0.9
- RFC 1945 HTTP 1.0 (1996)
- RFC 2616 HTTP 1.1 (1999)

HTTP 1.1

In June 2014, RFC 2616 was retired and HTTP/1.1 was redefined by

- RFC 7230 - HTTP/1.1: Message Syntax and Routing
- RFC 7231 - HTTP/1.1: Semantics and Content
- RFC 7232 - HTTP/1.1: Conditional Requests
- RFC 7233 - HTTP/1.1: Range Requests
- RFC 7234 - HTTP/1.1: Caching
- RFC 7235 - HTTP/1.1: Authentication

We will stick to HTTP 1.1

HTTP/2

HTTP/2 (originally named HTTP/2.0) is a major revision of the HTTP network protocol used by the World Wide Web. It was derived from the earlier experimental SPDY protocol, originally developed by Google.

The changes do not require any modification to how existing web applications work, but new applications can take advantage of new features for increased speed.

HTTP/2 leaves most of HTTP 1.1's high-level syntax, such as methods, status codes, header fields, and URIs, the same.

What is new is how the data is framed and transported between the client and the server.

HTTP/3

HTTP/3 H3 is the upcoming third major version

HTTP/3 is a draft based on a previous RFC draft, then named "Hypertext Transfer Protocol (HTTP) over QUIC".

QUIC is a transport layer network protocol developed initially by Google where user space congestion control is used over the User Datagram Protocol (UDP).

https://tools.ietf.org/html/draft-ietf-quic-http-23

Q

How does HTTP work ?

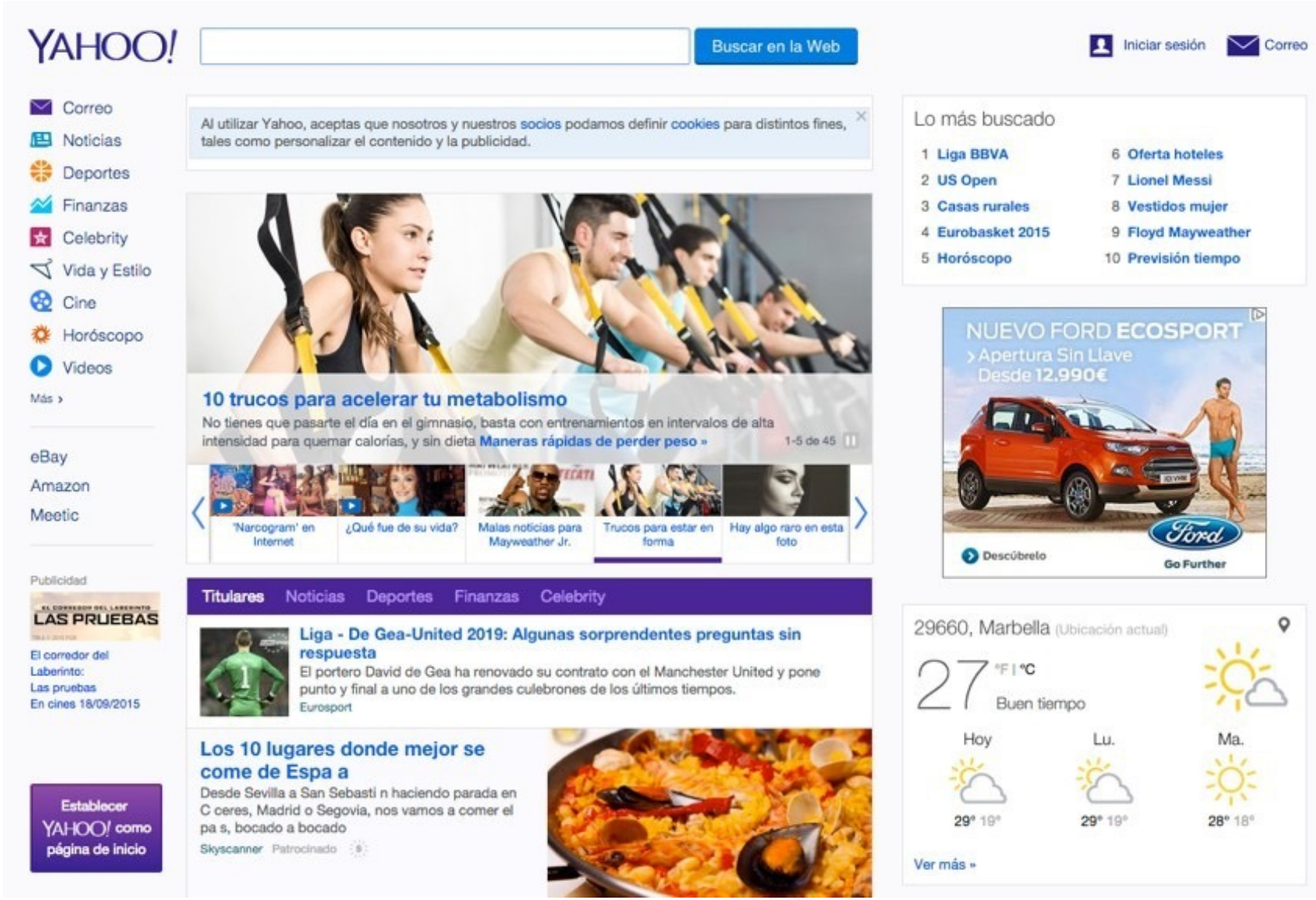
Web and HTTP

- web page consists of objects
- object can be text file, JPEG image, Flash objects, audio file,…
- a web page contains of base HTML-file which includes **several referenced objects**
- each object is addressable by a URL:
  - www.somecompany.com/someDept/pic.gif
  - www. somecompany. com is the host name
  - someDept/pic.gif is the path to the object

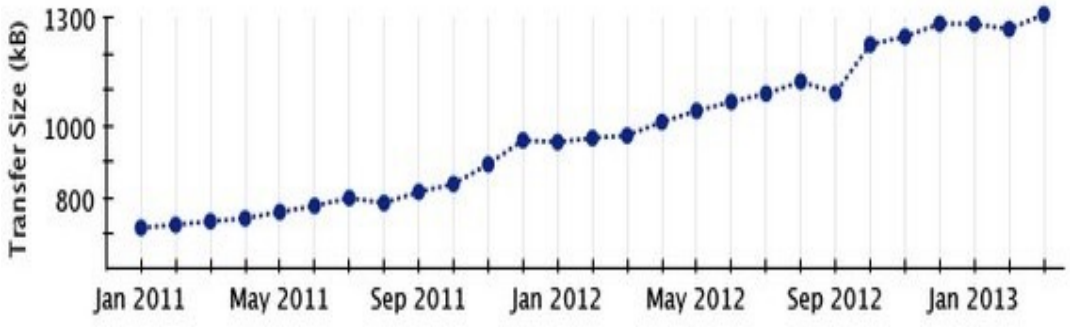
Web in 1996

- **Arts** - - *Humanities*, *Photography*, *Architecture*, ...
- **Business and Economy** [Xtra!] - - *Directory*, *Investments*, *Classifieds*, ...
- **Computers and Internet** [Xtra!] - - *Internet*, *WWW*, *Software*, *Multimedia*, ...
- **Education** - - *Universities*, *K-12*, *Courses*, ...
- **Entertainment** [Xtra!] - - *TV*, *Movies*, *Music*, *Magazines*, ...
- **Government** - - *Politics* [Xtra!], *Agencies*, *Law*, *Military*, ...
- **Health** [Xtra!] - - *Medicine*, *Drugs*, *Diseases*, *Fitness*, ...
- **News** [Xtra!] - - *World* [Xtra!], *Daily*, *Current Events*, ...
- **Recreation and Sports** [Xtra!] - - *Sports*, *Games*, *Travel*, *Autos*, *Outdoors*, ...
- **Reference** - - *Libraries*, *Dictionaries*, *Phone Numbers*, ...
- **Regional** - - *Countries*, *Regions*, *U.S. States*, ...
- **Science** - - *CS*, *Biology*, *Astronomy*, *Engineering*, ...
- **Social Science** - - *Anthropology*, *Sociology*, *Economics*, ...
- **Society and Culture** - - *People*, *Environment*, *Religion*, ...

Web Today

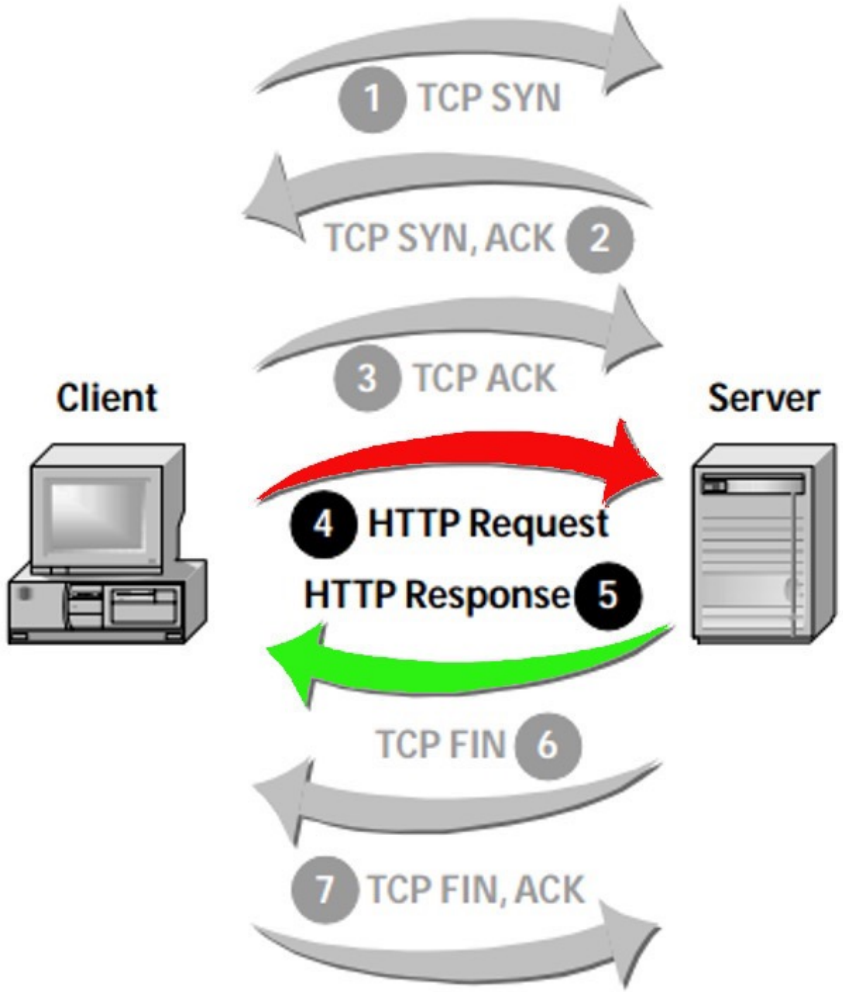

YAHOO!
Buscar en la Web
Iniciar sesión
Correo
Correo
Noticias
Deportes
Finanzas
Celebrity
Vida y Estilo
Cine
Horóscopo
Videos
Más ›
eBay
Amazon
Meetic
Publicidad
LAS PRUEBAS
El corredor del Laberinto: Las pruebas En cines 18/09/2015
Establecer YAHOO! como página de inicio
Al utilizar Yahoo, aceptas que nosotros y nuestros socios podamos definir cookies para distintos fines, tales como personalizar el contenido y la publicidad.
10 trucos para acelerar tu metabolismo
No tienes que pasarte el día en el gimnasio, basta con entrenamientos en intervalos de alta intensidad para quemar calorías, y sin dieta Maneras rápidas de perder peso »
1-5 de 45
'Narcogram' en Internet
¿Qué fue de su vida?
Malas noticias para Mayweather Jr.
Trucos para estar en forma
Hay algo raro en esta foto
Titulares
Noticias
Deportes
Finanzas
Celebrity
Liga - De Gea-United 2019: Algunas sorprendentes preguntas sin respuesta
El portero David de Gea ha renovado su contrato con el Manchester United y pone punto y final a uno de los grandes culebrones de los últimos tiempos.
Eurosport
Los 10 lugares donde mejor se come de Espa a
Desde Sevilla a San Sebasti n haciendo parada en C ceres, Madrid o Segovia, nos vamos a comer el pa s, bocado a bocado
Skyscanner Patrocinado
Lo más buscado
1 Liga BBVA
2 US Open
3 Casas rurales
4 Eurobasket 2015
5 Horóscopo
6 Oferta hoteles
7 Lionel Messi
8 Vestidos mujer
9 Floyd Mayweather
10 Previsión tiempo
NUEVO FORD ECOSPORT
› Apertura Sin Llave
Desde 12.990€
Descúbrelo
Ford
Go Further
29660, Marbella (Ubicación actual)
27 °F | °C
Buen tiempo
Hoy
Lu.
Ma.
29° 19°
29° 19°
28° 18°
Ver más »

Our applications are complex, and growing...



Content Type	Desktop		Mobile	
	Avg # of requests	Avg size	Avg # of requests	Avg size
HTML	10	56 KB	6	40 KB
Images	56	856KB	38	498KB
Javascript	15	221KB	10	146KB
CSS	5	36 KB	3	27 KB
Total	86+	1169+KB	57+	711+KB

HTTP requires a TCP connection



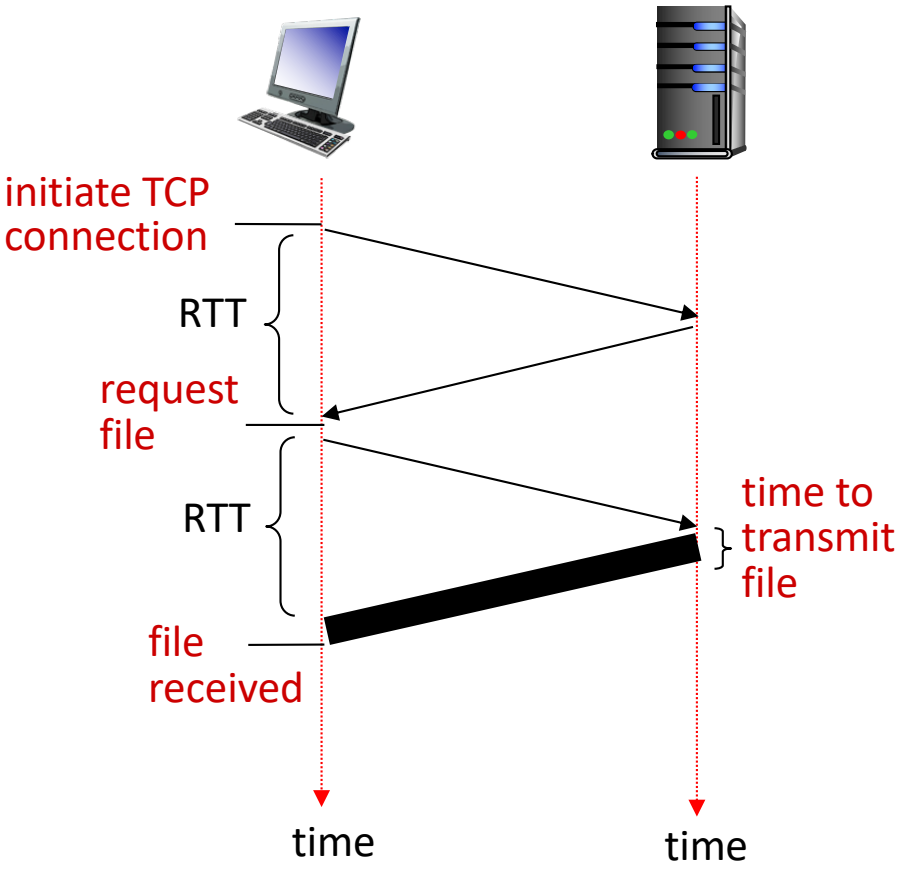
Before systems can exchange HTTP messages, they must establish a TCP connection. Steps 1, 2, and 3 in this example show the connection establishment. Once the TCP connection is available, the client sends the server an HTTP request. The final two steps, 6 and 7, show the closing of the TCP connection.

Non-persistent HTTP: response time

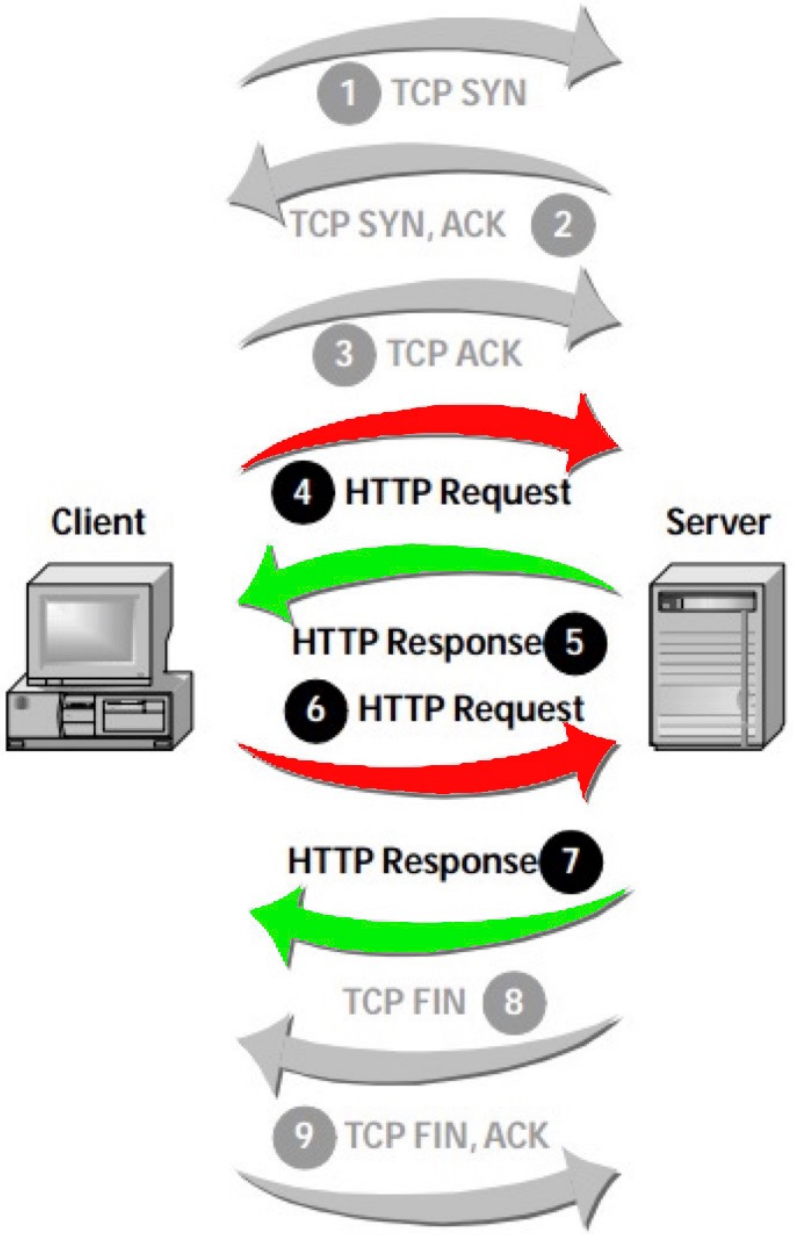
RTT (round trip time): time for a small packet to travel from client to server and back

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



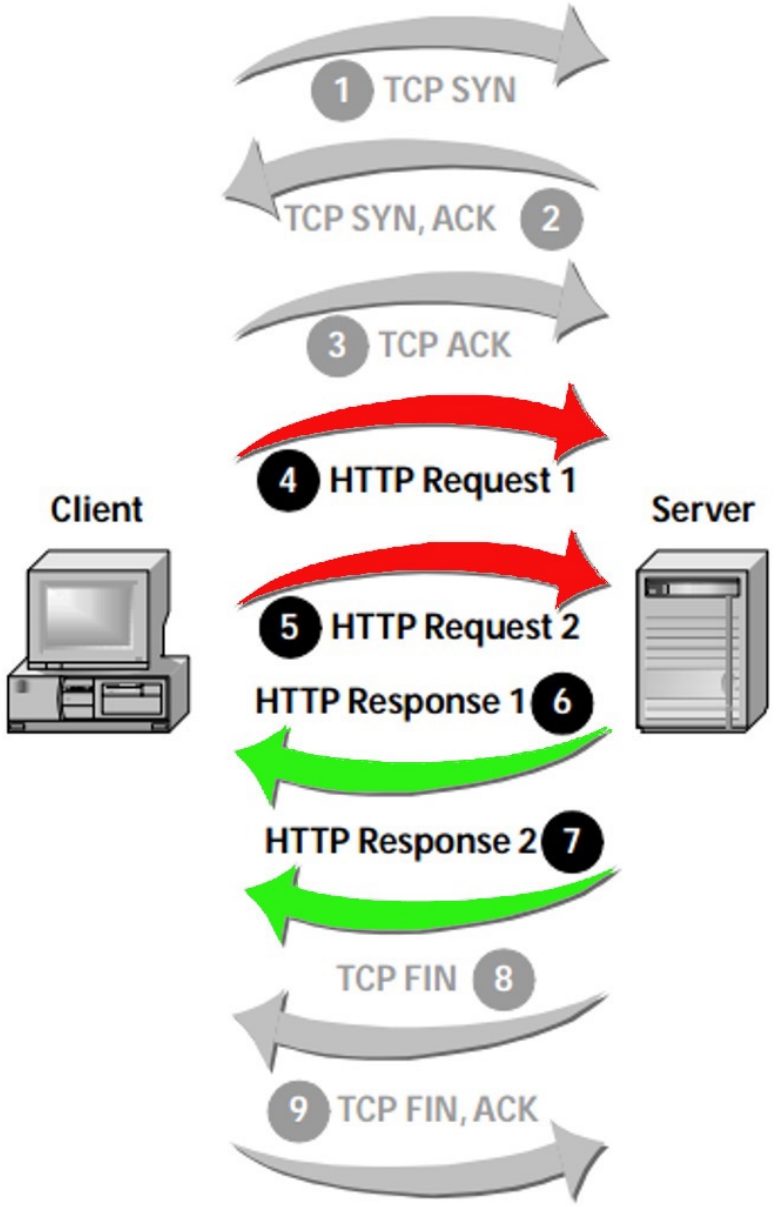
Persistent Connection (HTTP 1.1)



With persistent connections, a client can issue many HTTP requests over a single TCP connection. The first request is in step 4, which the server answers in step 5. In step 6 the client continues by sending the server another request on the same TCP connection. The server responds to this request in step 7 and then closes the TCP connection.

Pipelining (HTTP 1.1)

Pipelining lets an HTTP client issue new requests without waiting for responses from its previous messages. In the figure, the client sends its first request in step 4. It immediately follows that with a second request in step 5. The client does not wait for the server's response, which arrives in step 6.



HTTP connections

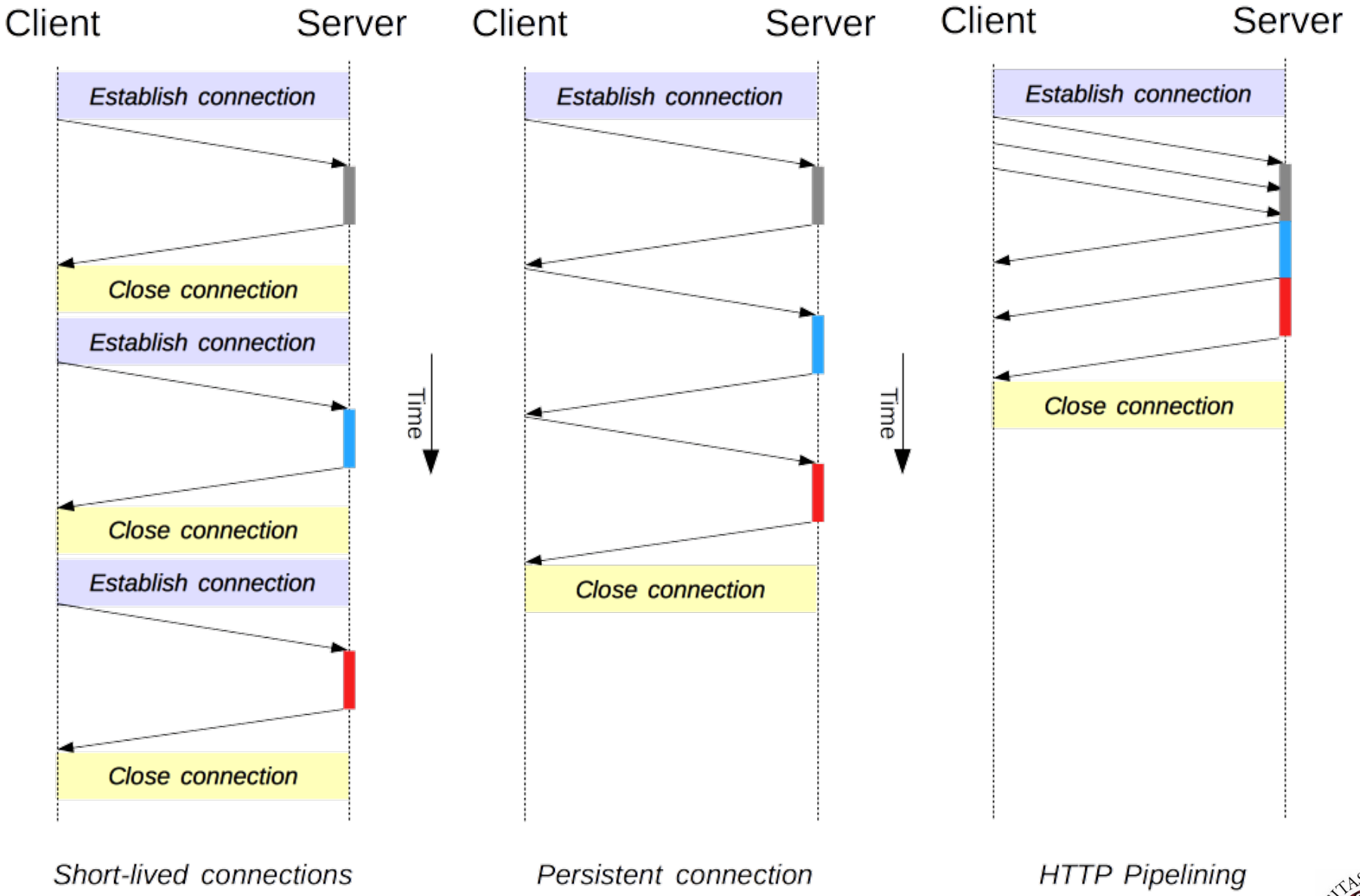
non-persistent HTTP

- at most one object sent over TCP connection
- connection then closed
- downloading multiple objects required multiple connections

persistent HTTP

- multiple objects can be sent over single TCP connection between client and server

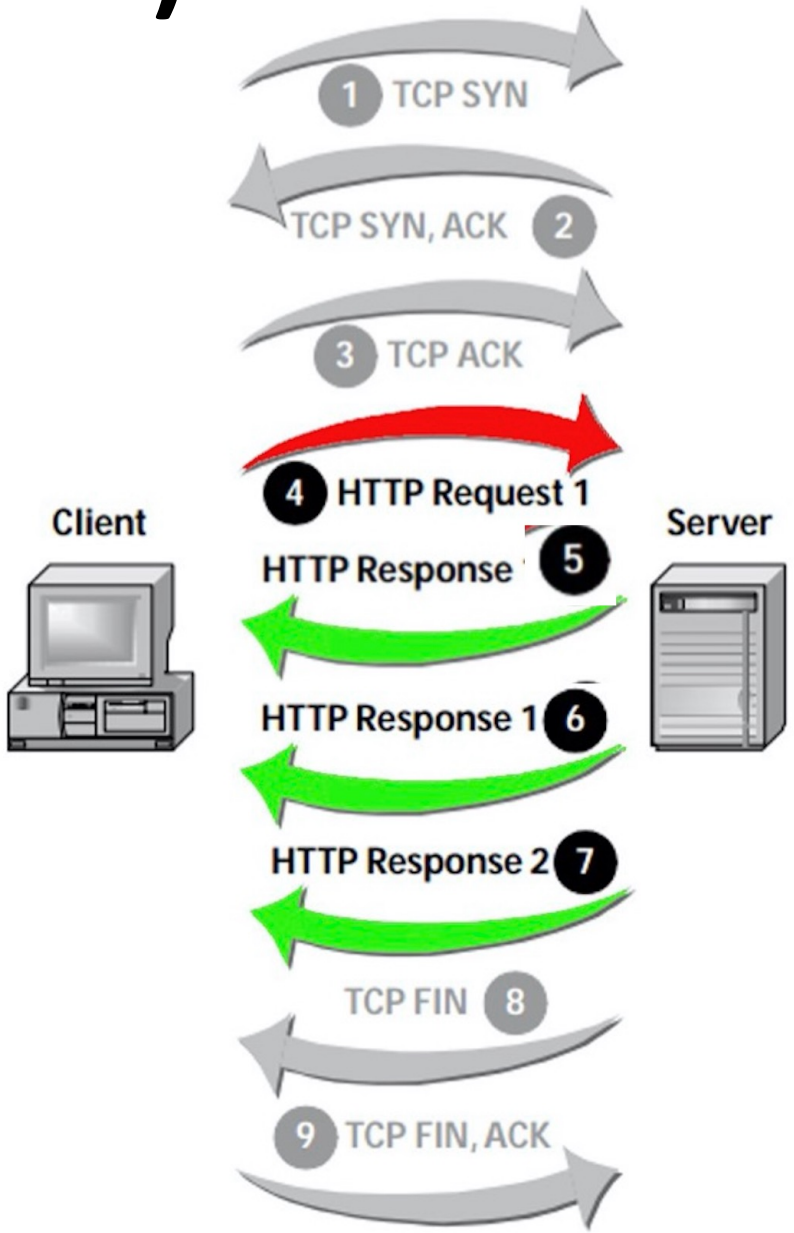
HTTP connections



Server Push (HTTP 1.2)

HTTP/2 Server Push allows an HTTP/2-compliant server to send resources to a HTTP/2-compliant client before the client requests them. It is, for the most part, a performance technique that can be helpful in loading resources preemptively.

HTTP/2 Server Push is not a notification mechanism from server to client. Instead, pushed resources are used by the client when it may have otherwise produced a request to get the resource anyway; this can result in wasted bandwidth if said pushed resources go unused by the client, however.



Q

How is the HTTP request structured ?

HTTP Requests

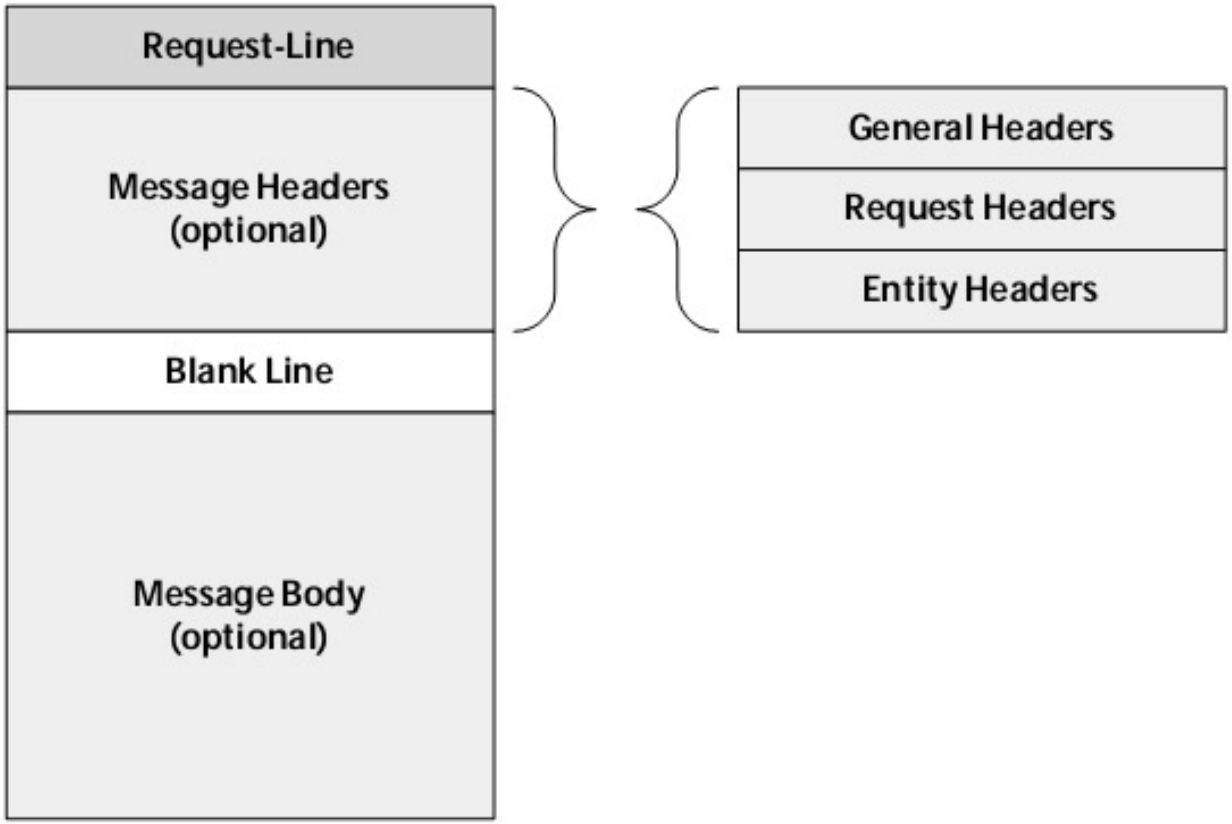
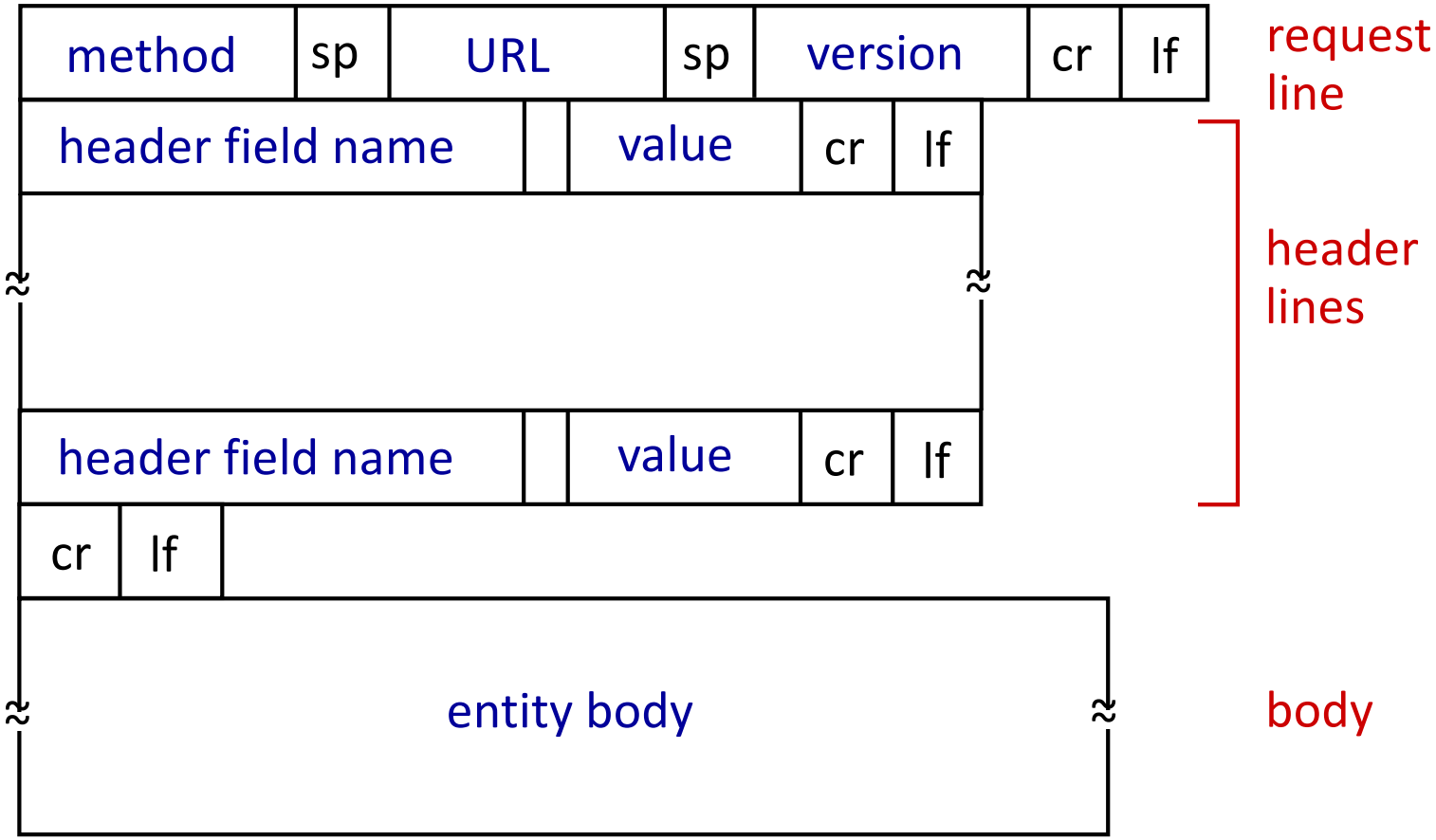


Figure 3.1 ▶
An HTTP request begins with a Request-Line and may include headers and a message body. The headers can describe general communications, the specific request, or the included message body.

HTTP request message: general format

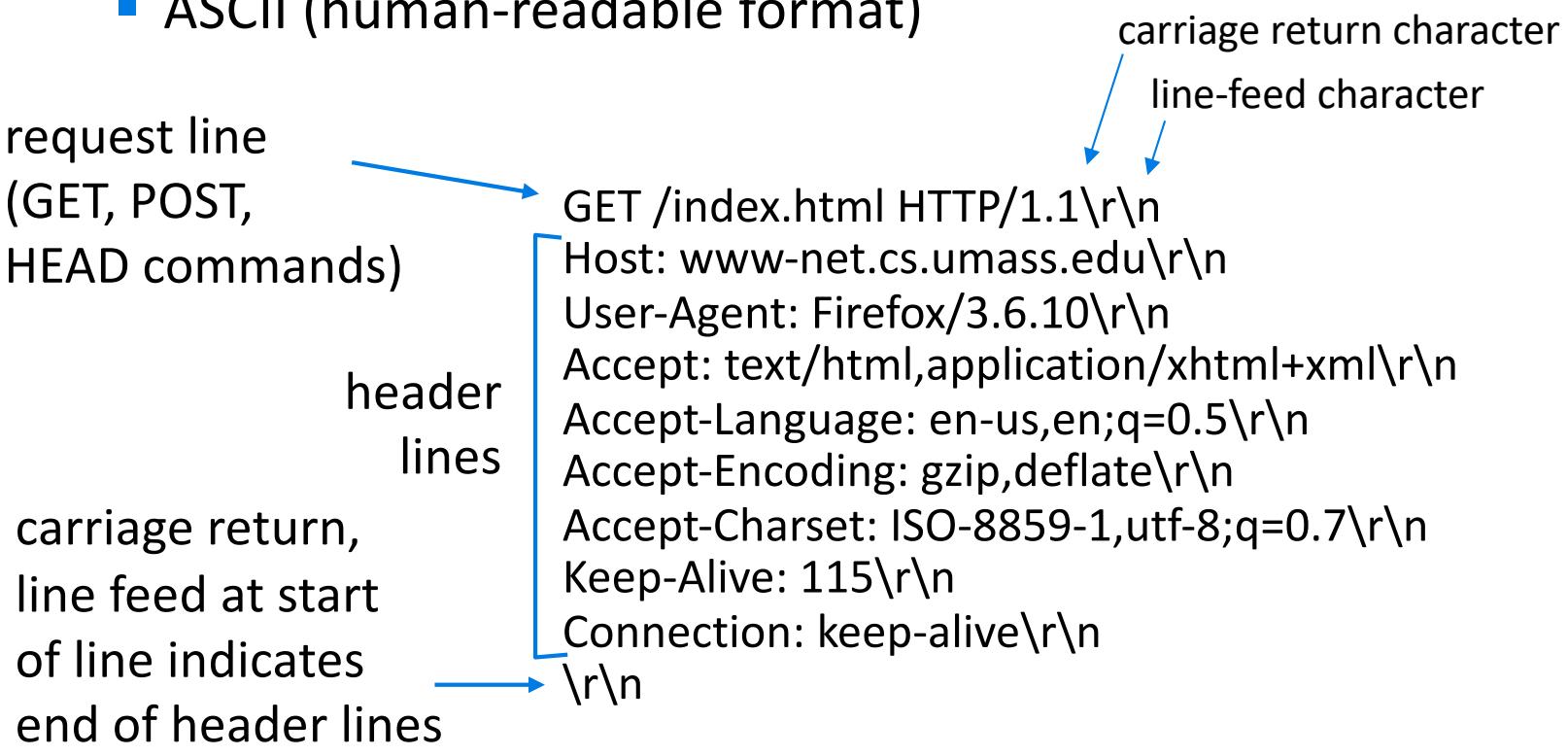


HTTP requests and responses Messages

- HTTP requests and responses are both types of Internet Messages (RFC 822), and share a general format:
  - **A Start Line, followed by a CRLF**
    - Request Line for requests
    - Status Line for responses
  - **Zero or more Message Headers**
    - field-name ":" [field-value] CRLF
  - **An empty line**
    - Two CRLFs mark the end of the Headers
  - **An optional Message Body if there is a payload**
    - All or part of the "Entity Body" or "Entity"

HTTP request message

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



Method types

HTTP/1.0:

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

HTTP/1.1:

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

A Closer Look at the Request Methods

- **GET**
  - By far most common method
  - Retrieves a resource from the server
  - Supports passing of query string arguments
- **HEAD**
  - Retrieves only the Headers associated with a resource but not the entity itself
  - Highly useful for protocol analysis, diagnostics
- **POST**
  - Allows passing of data in entity rather than URL
  - Can transmit of far larger arguments that GET
  - Arguments not displayed on the URL

More Request Methods, cont.

- **OPTIONS**
  - Shows methods available for use on the resource (if given a path) or the host (if given a "*")
- **TRACE**
  - Diagnostic method for assessing the impact of proxies along the request-response chain
- **CONNECT**
  - A common extension method for Tunneling other protocols through HTTP
- **PUT, DELETE**
  - Used in HTTP publishing (e.g., WebDav)

Web-based Distributed Authoring and Versioning (WebDAV, RFC 4918) is a set of methods based on the Hypertext Transfer Protocol (HTTP) that facilitates collaboration between users in editing and managing documents and files stored on World Wide Web servers.

Q

Which types of headers are there?

A Closer Look at HTTP Headers

Headers come in four major types, some for requests, some for responses, some for both:

- **General Headers**
  - Provide info about messages of both kinds
- **Request Headers**
  - Provide request-specific info
- **Response Headers**
  - Provide response-specific info
- **Entity Headers**
  - Provide info about request and response entities
- Extension headers are also possible

General Headers

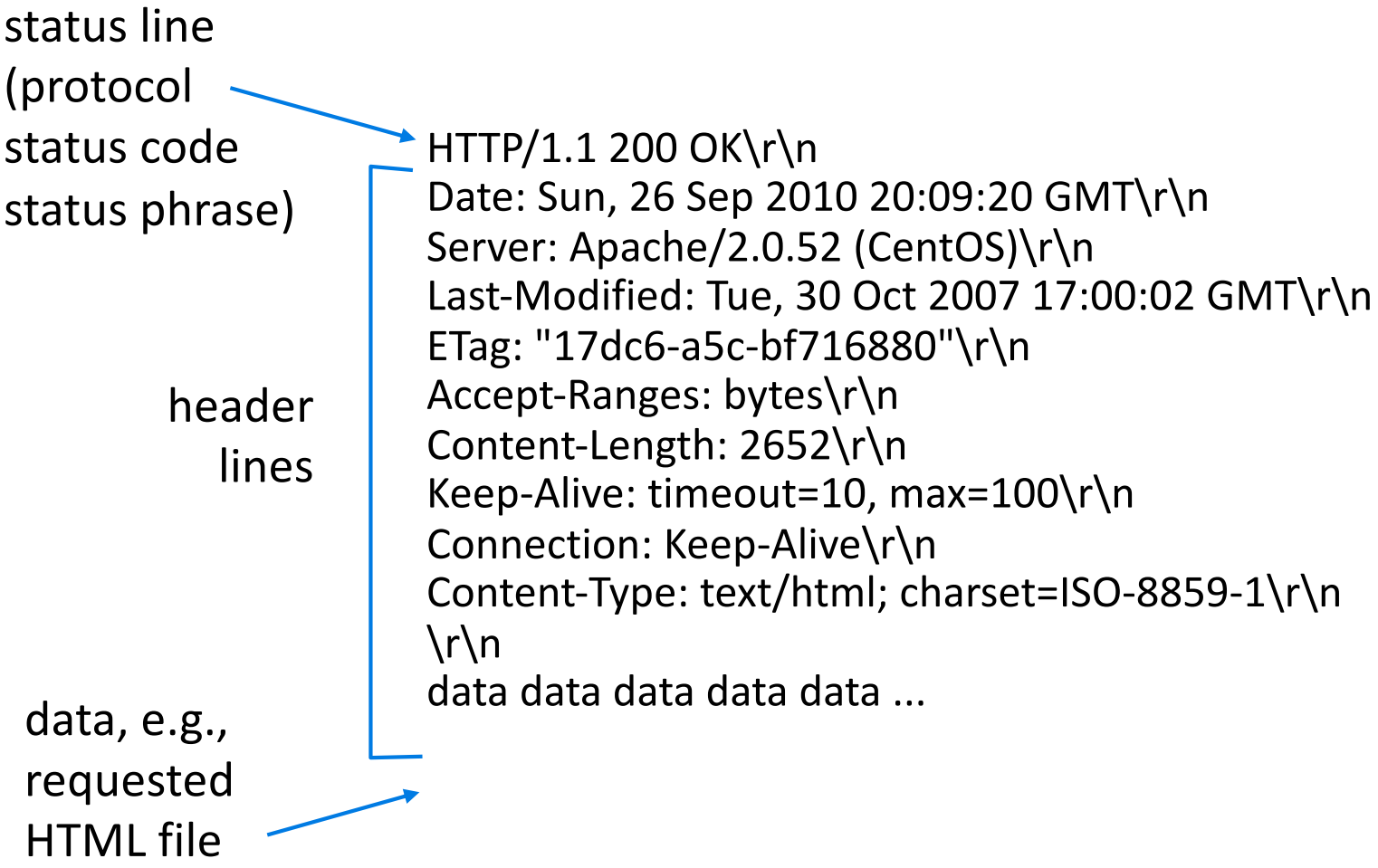
- **Connection** – lets clients and servers manage connection state
  - Connection: Keep-Alive
  - Connection: close
- **Date** – when the message was created
  - Date: Sat, 31-May-03 15:00:00 GMT
- **Via** – shows proxies that handled message
  - Via: 1.1 www.myproxy.com (Squid/1.4)
- **Cache-Control** – Among the most complex of headers, enables caching directives
  - Cache-Control: no-cache

Request Headers

- **Host** – The hostname (and optionally port) of server to which request is being sent
- **Referer** – The URL of the resource from which the current request URI came
  - Referer: http://www.host.com/login.asp
- **User-Agent** – Name of the requesting application, used in browser sensing
  - User-Agent: Mozilla/4.0 (Compatible; MSIE 6.0)
- **Accept** and its variants – Inform servers of client's capabilities and preferences
  - Enables content negotiation
  - Accept: image/gif, image/jpeg;q=0.5
  - Accept- variants for Language, Encoding, Charset
- **Cookie** How clients pass cookies back to the servers that set them
  - Cookie: id=23432;level=3

HTTP: response

HTTP response messages



A Closer Look at the Status Line

- **Consists of three major parts:**
- The HTTP Version
  - Just like third part of Request Line
- Status Code
  - **5 groups of 3 digit integers indicating the result of the attempt to satisfy the request**The Reason Phrase followed by the CRLF
  - Short textual description of the status code

Table 3.2 HTTP Status Code Categories

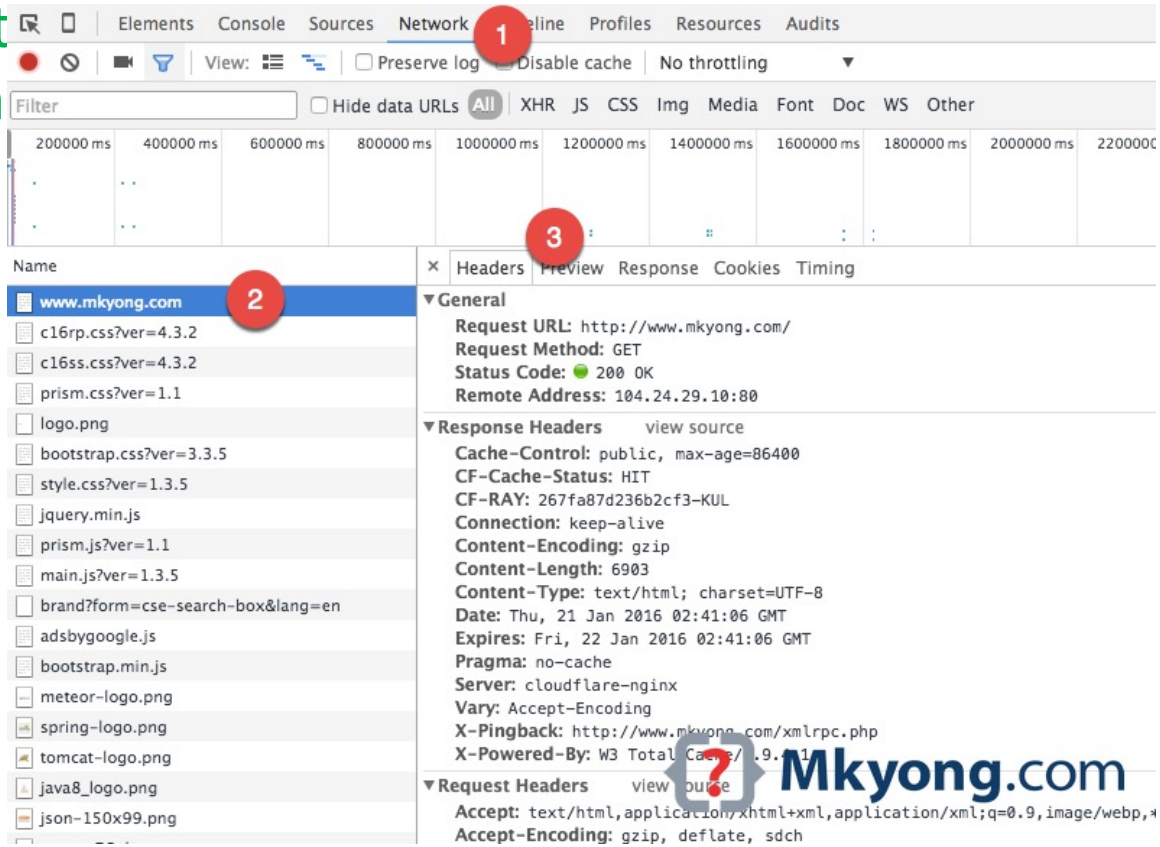
Status Code	Meaning
100-199	Informational; the server received the request but a final result is not yet available.
200-299	Success; the server was able to act on the request successfully.
300-399	Redirection; the client should redirect the request to a different server or resource.
400-499	Client error; the request contained an error that prevented the server from acting on it successfully.
500-599	Server error; the server failed to act on a request even though the request appears to be valid.

HTTP response status codes examples

- status code appears in 1st line in server-to-client response message.
- some sample codes:
  - 200 OK
    - request succeeded, requested object later in this msg
  - 301 Moved Permanently
    - requested object moved, new location specified later in this msg (Location:)
  - 400 Bad Request
    - request msg not understood by server
  - 404 Not Found
    - requested document not found on this server
  - 505 HTTP Version Not Supported

How to view HTTP headers in Google Chrome?

- In Chrome, visit a URL, right click, select Inspect to open the developer tools.
- Select Network tab.
- Reload the page, select any HTTP request on the left panel, and the HTTP headers will be displayed on the right panel.



https://mkyong.com/computer-tips/how-to-view-http-headers-in-google-chrome/

Making a simple HTTP request using Telnet

- Open a TCP connection to a host
  - Can borrow telnet protocol to do this, by pointing it at the default HTTP port (80)
  - C:\>telnet www.google.com 80
- Ask for a resource using a minimal request syntax:
  - GET / HTTP/1.1 <CRLF>
  - Host: www.google.ps <CRLF>
  - <CRLF>
- A Host header is required for HTTP 1.1 connections, though not for HTTP 1.0

Making a simple HTTP request using Telnet

ronchet — telnet www.google.com 80 — 80×24

```
[MR-MBP-14955:~ ronchet$ telnet www.google.com 80
Trying 216.58.206.36...
Connected to www.google.com.
Escape character is '^]'.
GET / HTTP/1.1
Host: www.google.com

HTTP/1.1 200 OK
Date: Sun, 09 Feb 2020 08:54:19 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
Server: gws
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
Set-Cookie: 1P_JAR=2020-02-09-08; expires=Tue, 10-Mar-2020 08:54:19 GMT; path=/;
 domain=.google.com; Secure
Set-Cookie: NID=197=BZn2lJOlNIiKeLkhSA3YoTAgo5E0aCh8Sj1ileUG2a8d5Cw_lSQVcZ0j0hH8
3nb18ieVoFlVem5lvbWiB4zH0EHXAoTS_Bc4P2OxmPPjuyFMwyvmPkTX24R2cO9BiRbrbKCirX7C2JrK
fkbpXbJnGWo00zW9Nxp2p0XZOjWLP7o; expires=Mon, 10-Aug-2020 08:54:19 GMT; path=/;
domain=.google.com; HttpOnly
Accept-Ranges: none
Vary: Accept-Encoding
```

Response Headers

- **Server** – The server's name and version
  - Server: Microsoft-IIS/5.0
  - Can be problematic for security reasons

- **Set-Cookie** – This is how a server sets a cookie on a client
  - Set-Cookie: id=234; path=/shop; expires=Sat, 31-May-03 15:00:00 GMT; secure

Entity Headers

- **Allow** – Lists the request methods that can be used on the entity
  - Allow: GET, HEAD, POST
- **Location** – Gives the alternate or new location of the entity
  - Used with 3xx response codes (redirects)
  - Location: http://latemar.science.unitn.it/segue/filebrowser.php
- **Content-Encoding** – specifies encoding performed on the body of the response
  - Used with HTTP compression
  - Corresponds to Accept-Encoding request header
  - Content-Encoding: gzip
- **Content-Length** – The size of the entity body in bytes
- **Content-Location** – The actual if different than its request URL
- **Content-Type** – specifies Media (MIME) type of the entity body