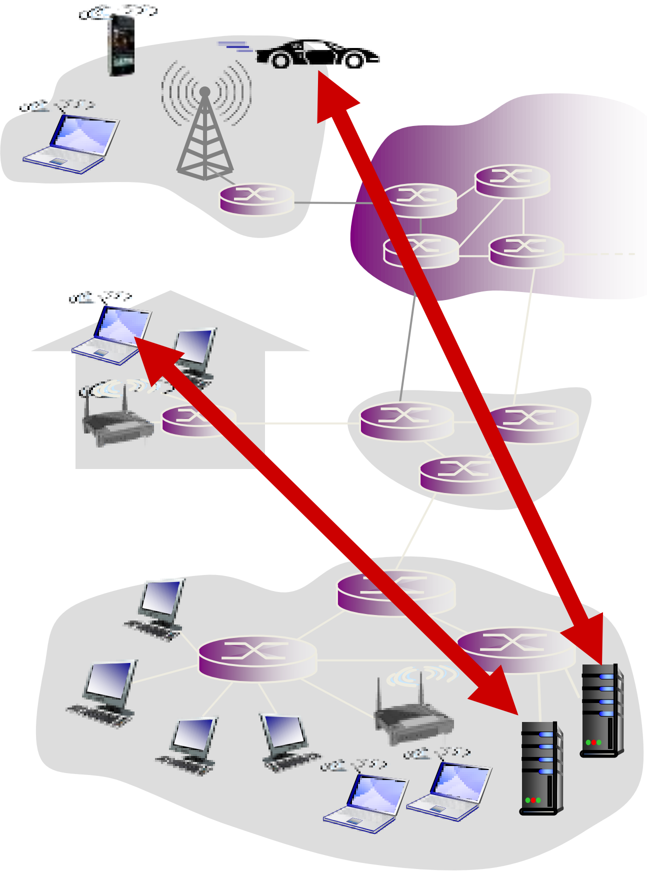
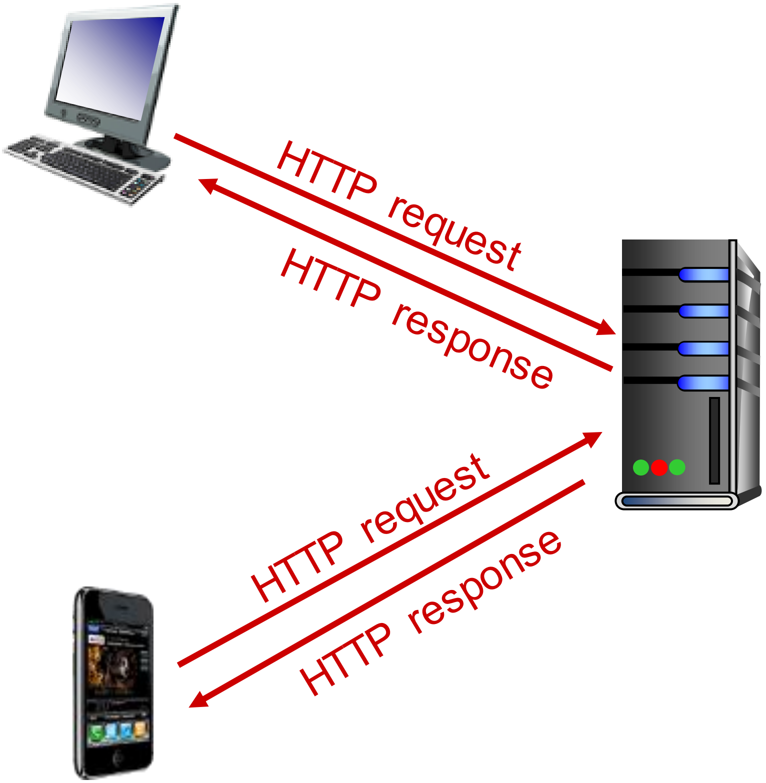
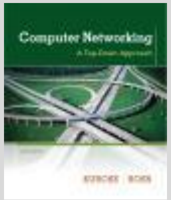
Network principles, the web and HTTP



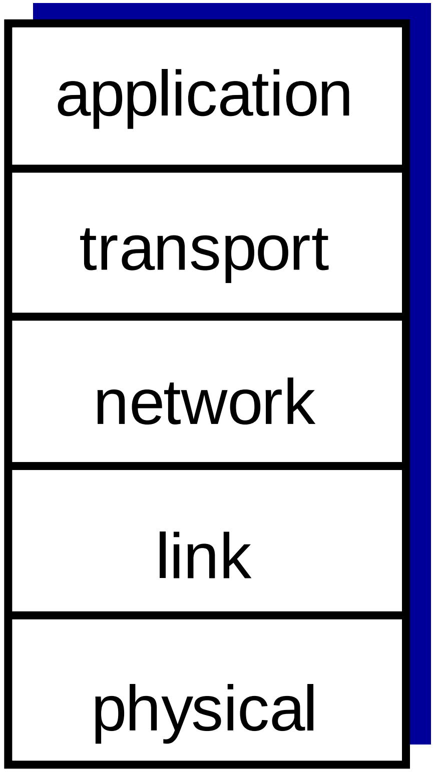


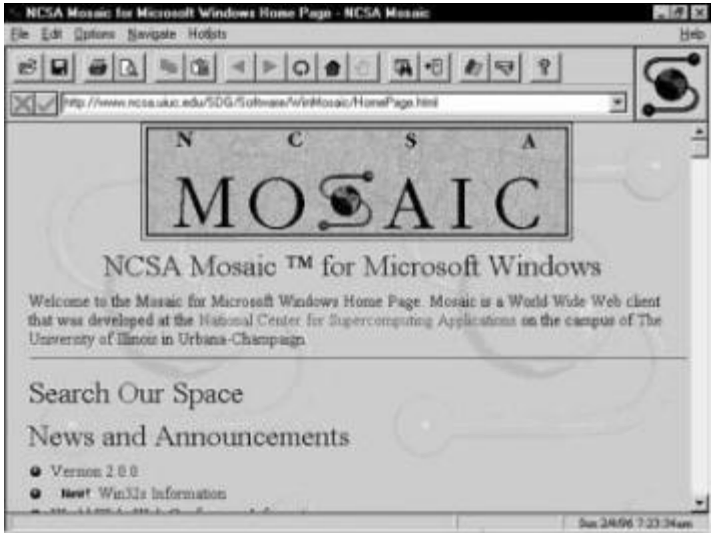
Computer Networking: A Top Down Approach
6th edition
Jim Kurose, Keith Ross
Addison-Wesley



Overview

- Chapter 2: Application Layer
  - – Many familiar services operate here
    - Web, email, Skype, P2P file sharing
  - – Socket programming
- Network architectures
  - – Client/server vs. Peer-to-peer
- Network principles
- The Web
  - – History
  - – Basic operation




NCSA Mosaic for Microsoft Windows Home Page - NCSA Mosaic
File Edit Options Navigate Hotlists Help
http://www.ncsa.uiuc.edu/SDG/Software/WinMosaic/HomePage.html
N C S A
MOSAIC

NCSA Mosaic ™ for Microsoft Windows

Welcome to the Mosaic for Microsoft Windows Home Page. Mosaic is a World Wide Web client that was developed at the National Center for Supercomputing Applications on the campus of The University of Illinois in Urbana-Champaign.

Search Our Space

News and Announcements

- Version 2.0.0
- New! Win32s Information

Some network apps

- E-mail
- Web
- Text messaging
- Remote login
- P2P file sharing
- Multi-user network games
- Streaming stored video
  - YouTube, Hulu, Netflix
- Voice over IP
  - Skype
- Real-time video conferencing
- Social networking
- Search
- …

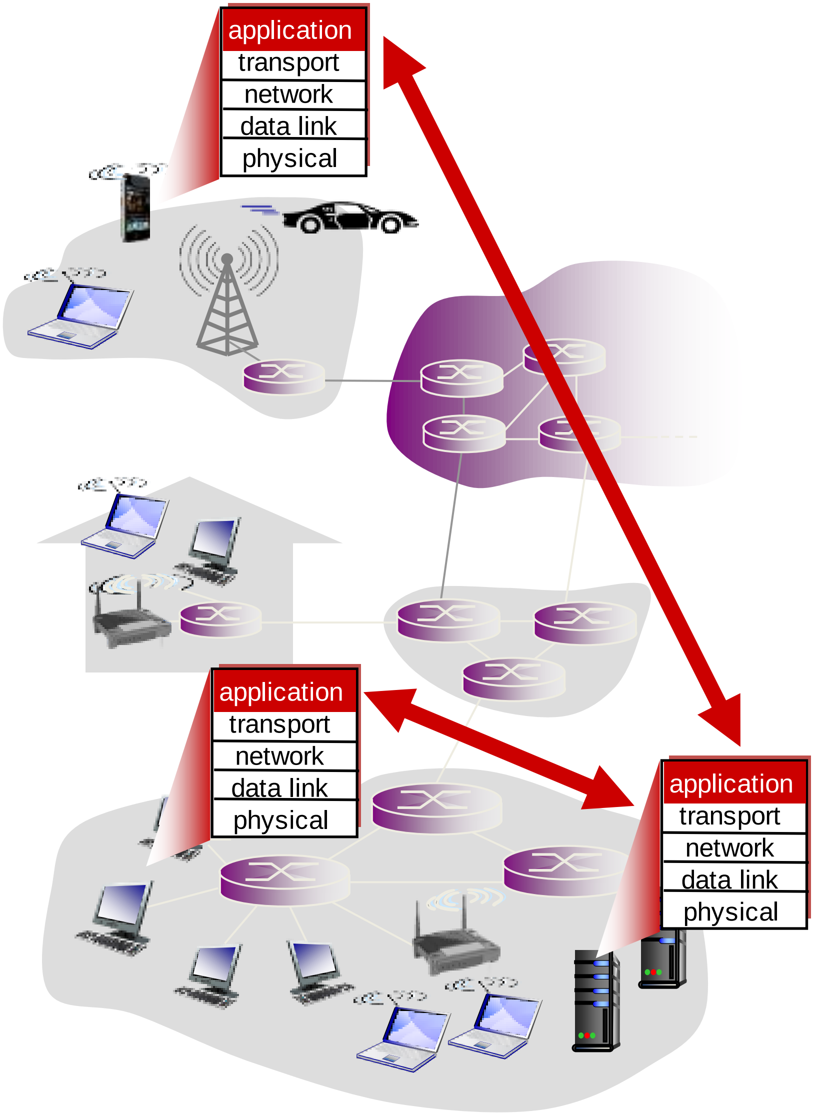
Creating a network app

Write programs that:

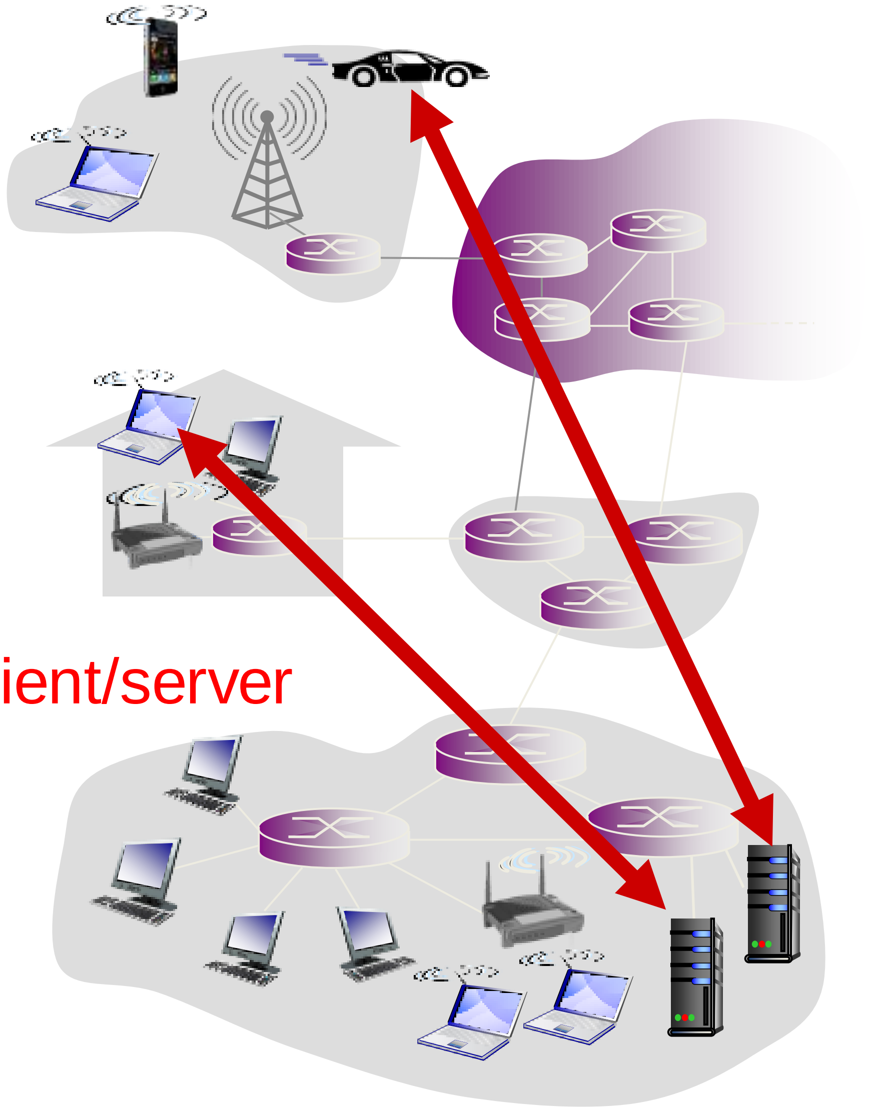
- Run on (different) *end systems*
- Communicate over network
- e.g. web server software communicates with browser software

No need to write software for network-core devices

- Network-core devices do not run user applications
- Applications on end systems allows for rapid app development



Client-server architecture



Server:

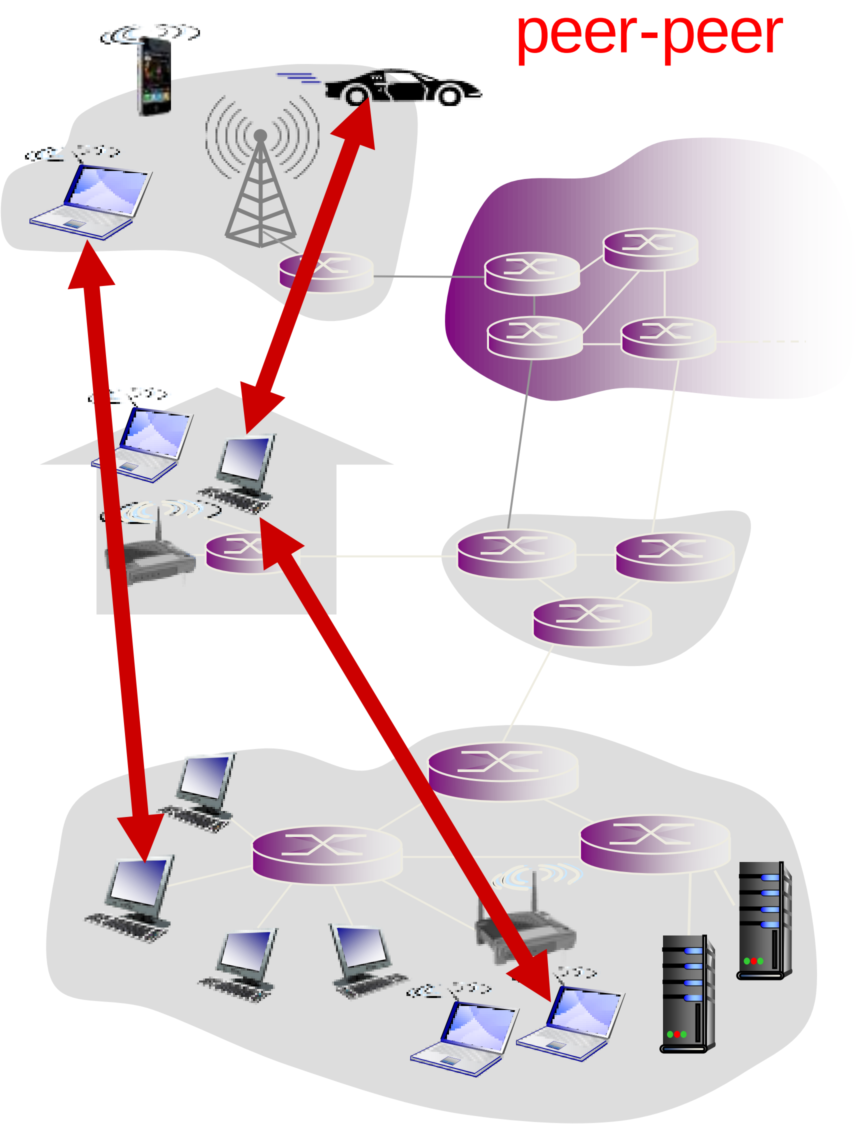
- Always-on host
- Permanent IP address
- Data centers for scaling

Clients:

- Communicate with server
- May be intermittently connected
- May have dynamic IP addresses
- Do not communicate directly with each other

Peer-to-Peer (P2P) architecture

- *No* always-on server
- Arbitrary end systems directly communicate
- Peers request service from other peers, provide service in return to other peers
  - *Self scalability* – new peers bring new capacity as well as demands
- Peers are intermittently connected and change IP addresses
  - Complex management



Processes communicating

Process: program running within a host

- Within same host, two processes communicate using inter-process communication (defined by OS)
- Processes in different hosts communicate by exchanging messages

Clients, Servers

Client process:

Process that initiates communication

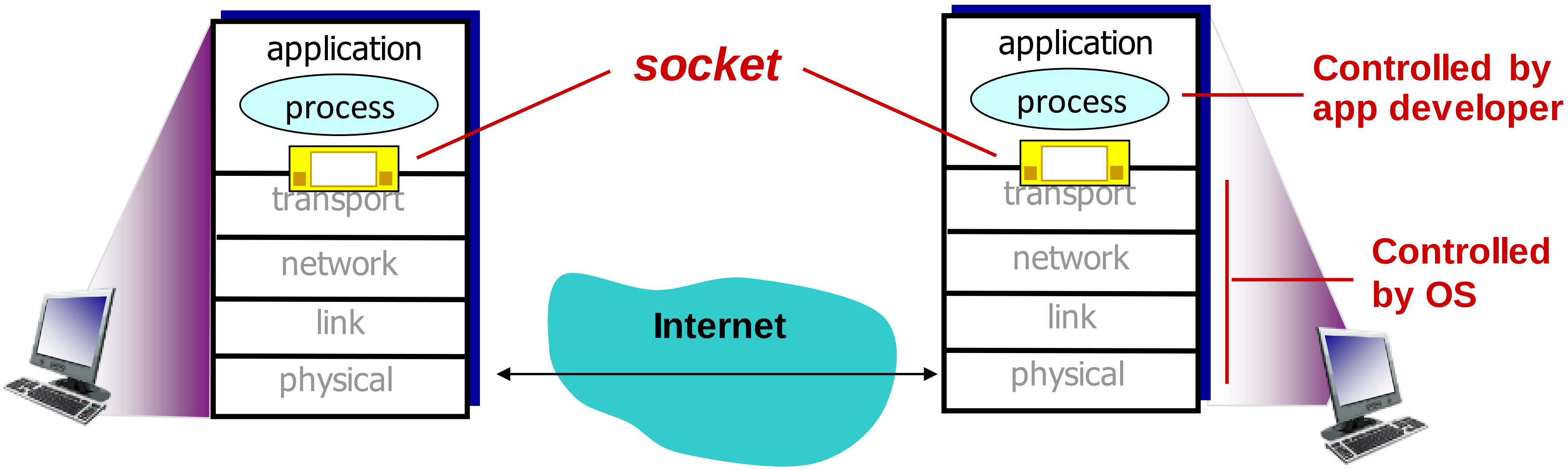
Server process:

Process that waits to be contacted

- Aside: applications with P2P architectures have client processes and server processes too

Sockets

- Process sends/receives messages to/from its socket
- Socket analogous to door
  - Sending process shoves message out door
  - Relies on transport infrastructure on other side to deliver message to socket at receiving process



Addressing processes

- To receive messages, process must have *identifier*
- Host device has unique 32-bit IP address
- *Q:* Does IP address of host on which process runs suffice for identifying the process?
- *A:* No, *many* processes can be running on same host

- *Identifier* includes both IP address and port numbers associated with process on host
- Example port numbers:
  - HTTP server: 80
  - Mail server: 25
- To send HTTP message to gaia.cs.umass.edu web server:
  - IP address: 128.119.245.12
  - Port number: 80

App-layer protocol defines

- Types of messages exchanged,
  - e.g. request, response
- Message syntax:
  - what fields in messages & how fields are delineated
- Message semantics
- Rules for when and how processes send & respond to messages

Open protocols:

- Defined in RFCs
- Allows for interoperability
- e.g. HTTP, SMTP

Proprietary protocols:

- e.g. Skype

What services does an app need?

Data integrity

- Some apps require 100% reliable data transfer
  - File transfers
  - Web transactions
- Other apps can tolerate some loss
  - Internet radio

Timing

- Some apps require low delay to be "effective"
  - Internet telephony
  - Interactive games

Throughput

- Some apps require minimum amount of throughput to be effective
  - Multimedia
- Other "elastic" apps make use of whatever throughput they get
  - File transfers
  - Electronic mail

Security

- Encryption, data integrity, end-point authentication

Requirements: common apps

application	data loss	throughput	time sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100's msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few kbps up	yes, 100's msec
text messaging	no loss	elastic	yes and no

Internet transport protocols

TCP service:

- *Reliable transport* between sending and receiving process
- *Flow control:* sender won't overwhelm receiver
- *Congestion control:* throttle sender when network overloaded
- *Does not provide:* timing, minimum throughput guarantee, security
- *Connection-oriented:* setup required between client and server processes

UDP service:

- *Unreliable data transfer* between sending and receiving process
- *Does not provide:* reliability, flow control, congestion control, timing, throughput guarantee, security, or connection setup,

Q: Why bother? Why is there a UDP?

Internet apps: transport protocols

application	application layer protocol	underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (e.g., YouTube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	TCP or UDP

Securing TCP

TCP & UDP

- No encryption
- Cleartext passwords sent traverse Internet in cleartext

SSL

- Provides encrypted TCP connection
- Data integrity
- End-point authentication

SSL is at app layer

- Apps use SSL libraries, which "talk" to TCP

SSL socket API

- Cleartext passwords sent traverse Internet encrypted
- See chapter 7

Internet history

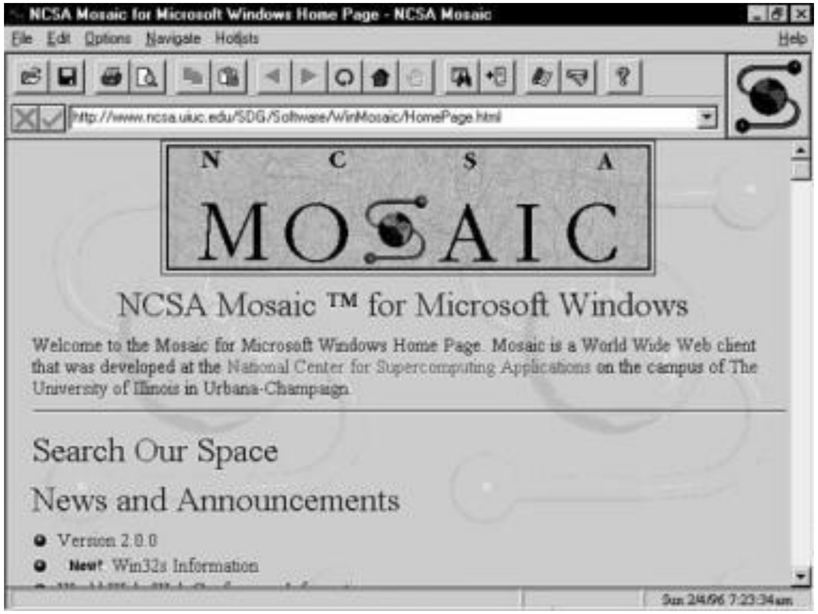
1990, 2000's: commercialization, the Web, new apps

- **Early 1990's:**
  - ARPAnet decommissioned
- **1991:**
  - NSF lifts restrictions on commercial use of NSFnet
- **Early 1990's:**
  - Web based on hypertext
  - [Bush 1945, Nelson 1960's]
- **Late 1990's:**
  - Commercialization of the web
- **2000's:**
  - More killer apps: instance messaging, P2P file sharing
  - Network security becomes important
  - Estimated 50 million hosts, 100+ million users
  - Backbone links running at Gbps

PROPRIETE CERN
This machine is a se-
DO NOT POWE
DOWN!!
PROPRIETE CERN

A short history of the web

- 1989 Tim Berners-Lee at CERN
- 1990 HTTP/0.9, HTML, URLs, first text-based browser
- 1993 Marc Andreesen releases NCSA Mosaic, graphical browser
- 1993 CERN agrees to release protocol royalty-free
- 1994 Andreesen forms Netscape
- 1994 W3C formed, standardizing protocols, encouraging interoperability


NCSA Mosaic for Microsoft Windows Home Page - NCSA Mosaic
File
Edit
Options
Navigate
Hotlists
Help
http://www.ncsa.uiuc.edu/SDG/Software/WinMosaic/HomePage.html
N C S A
MOSAIC
Sun 2/4/96 7:23:34am

NCSA Mosaic ™ for Microsoft Windows

Welcome to the Mosaic for Microsoft Windows Home Page. Mosaic is a World Wide Web client that was developed at the National Center for Supercomputing Applications on the campus of The University of Illinois in Urbana-Champaign.

Search Our Space

News and Announcements

- Version 2.0.0
- New! Win32s Information

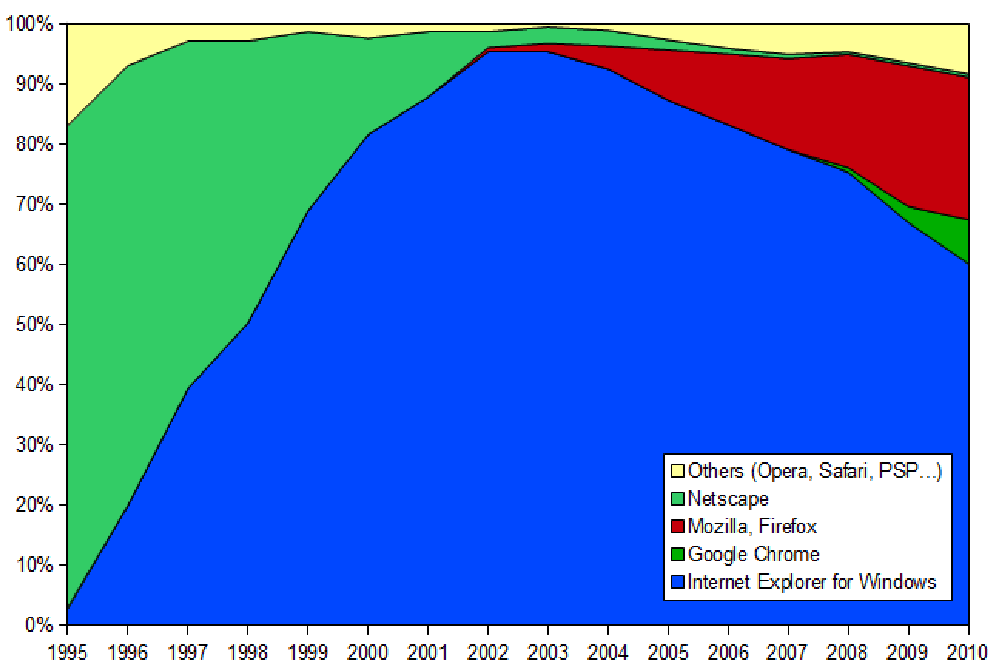
A short history of the web

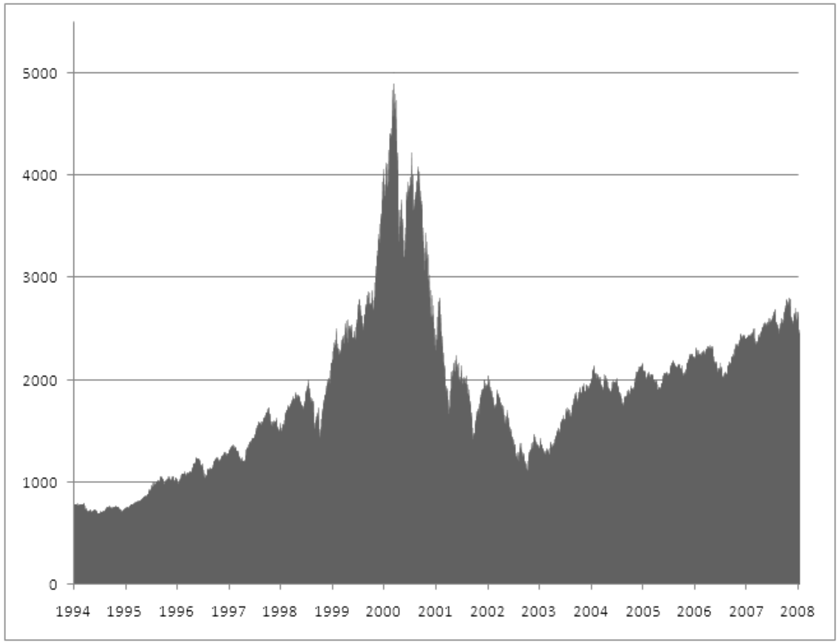
- 1994+ Browser wars between Netscape and IE
- 1990s-2000 Dot com era

Microsoft Internet Explorer

Netscape

BROWSERS WAR





"In the Web's first generation, Tim Berners-Lee launched the Uniform Resource Locator (URL), Hypertext Transfer Protocol (HTTP), and HTML standards with prototype Unix-based servers and browsers.

A few people noticed that the
Web might be better than Gopher.

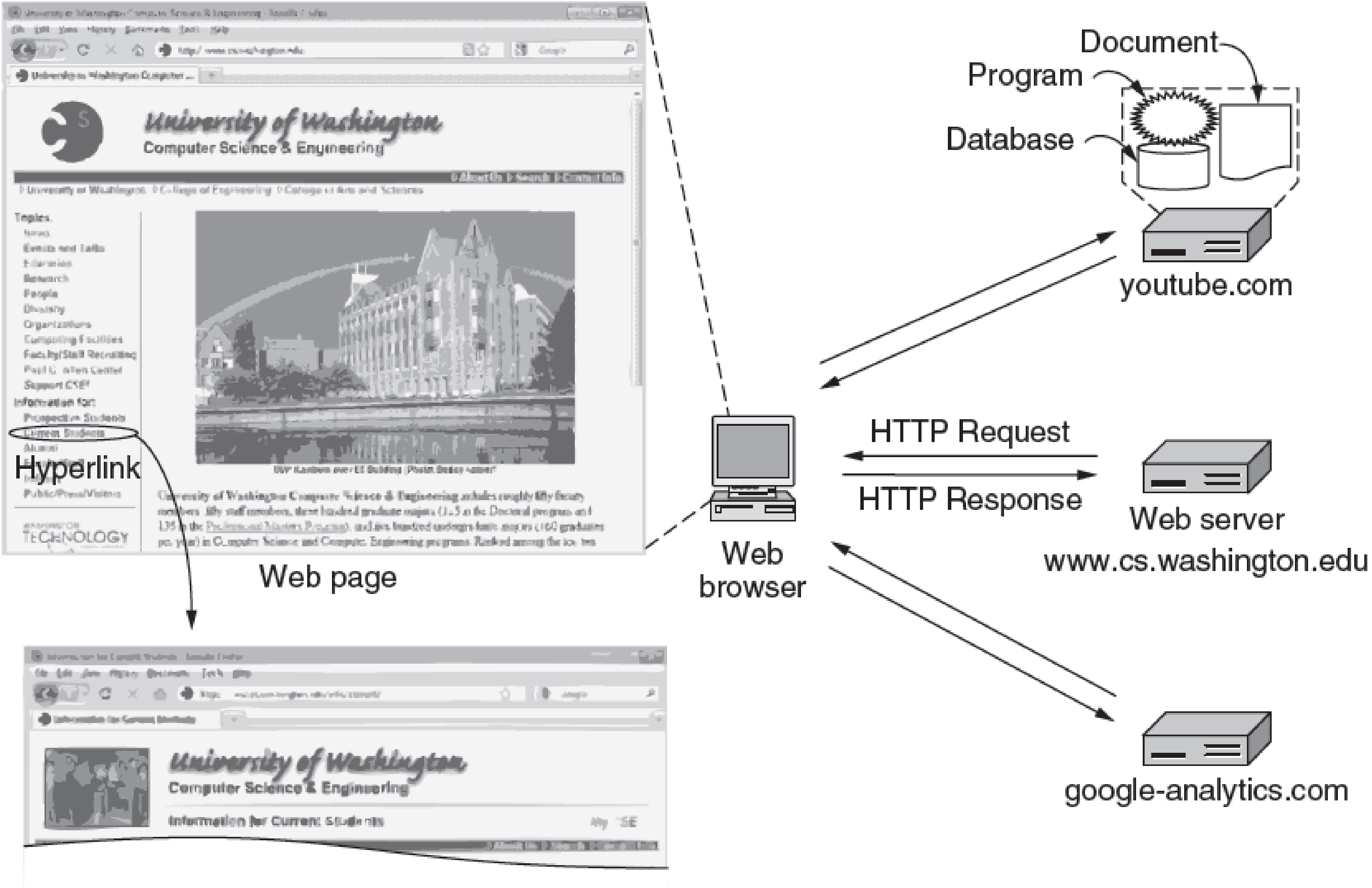
In the second generation, Marc Andreessen and Eric Bina developed NCSA Mosaic at the University of Illinois.

Several million then suddenly noticed that the
Web might be better than sex.

In the third generation, Andreessen and Bina left NCSA to found Netscape..."

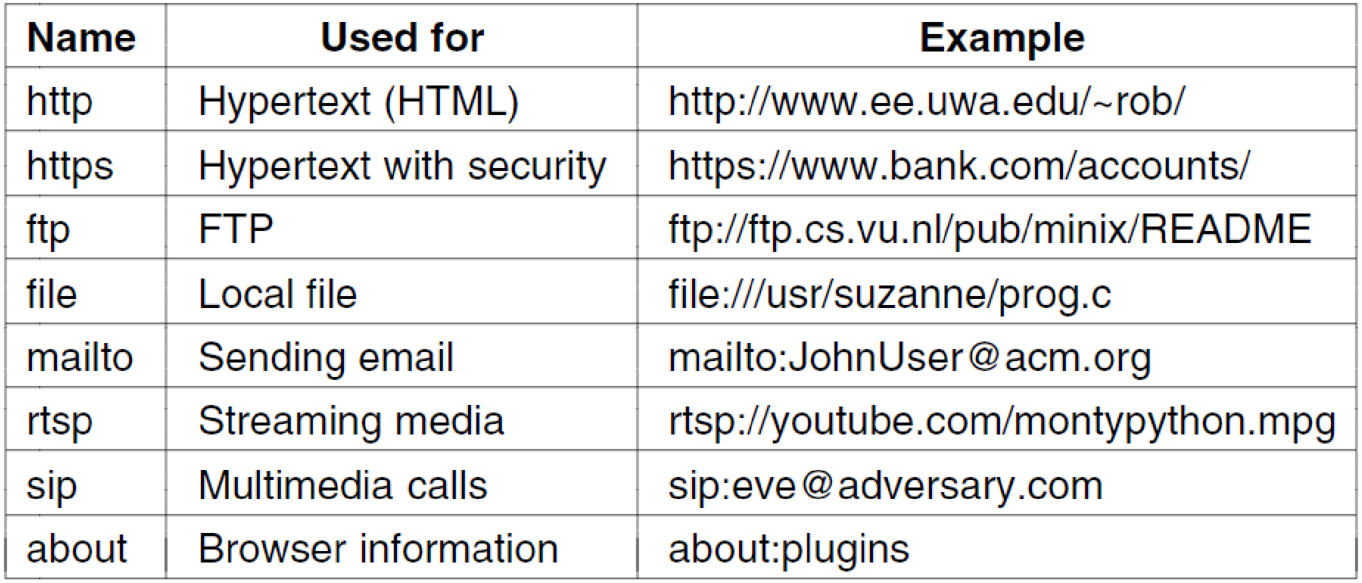
Microsoft and Netscape open some new fronts in escalating Web Wars
By Bob Metcalfe, InfoWorld, August 21, 1995, Vol. 17, Issue 34.

Architecture of the web



Web components: finding stuff

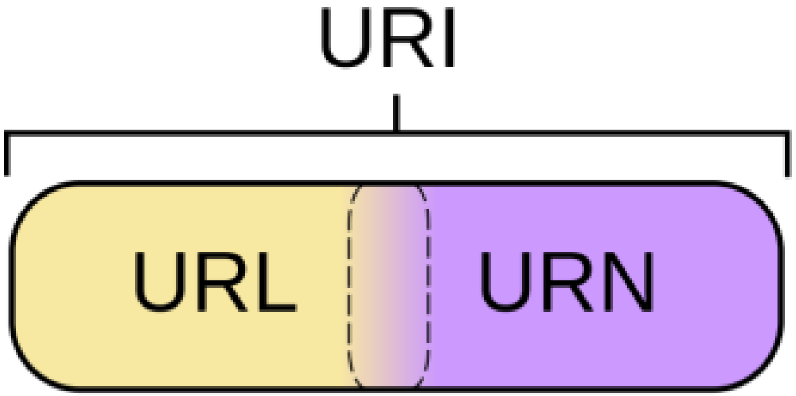
- Uniform Resource Locator (URL)
  - A page's worldwide name
  - Three parts:
    - Protocol (scheme)
    - DNS name of machine
    - Hierarchical name that models a file directory structure



Name	Used for	Example
http	Hypertext (HTML)	http://www.ee.uwa.edu/~rob/
https	Hypertext with security	https://www.bank.com/accounts/
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
file	Local file	file:///usr/suzanne/prog.c
mailto	Sending email	mailto:JohnUser@acm.org
rtsp	Streaming media	rtsp://youtube.com/montypython.mpg
sip	Multimedia calls	sip:eve@adversary.com
about	Browser information	about:plugins

Web components: finding stuff

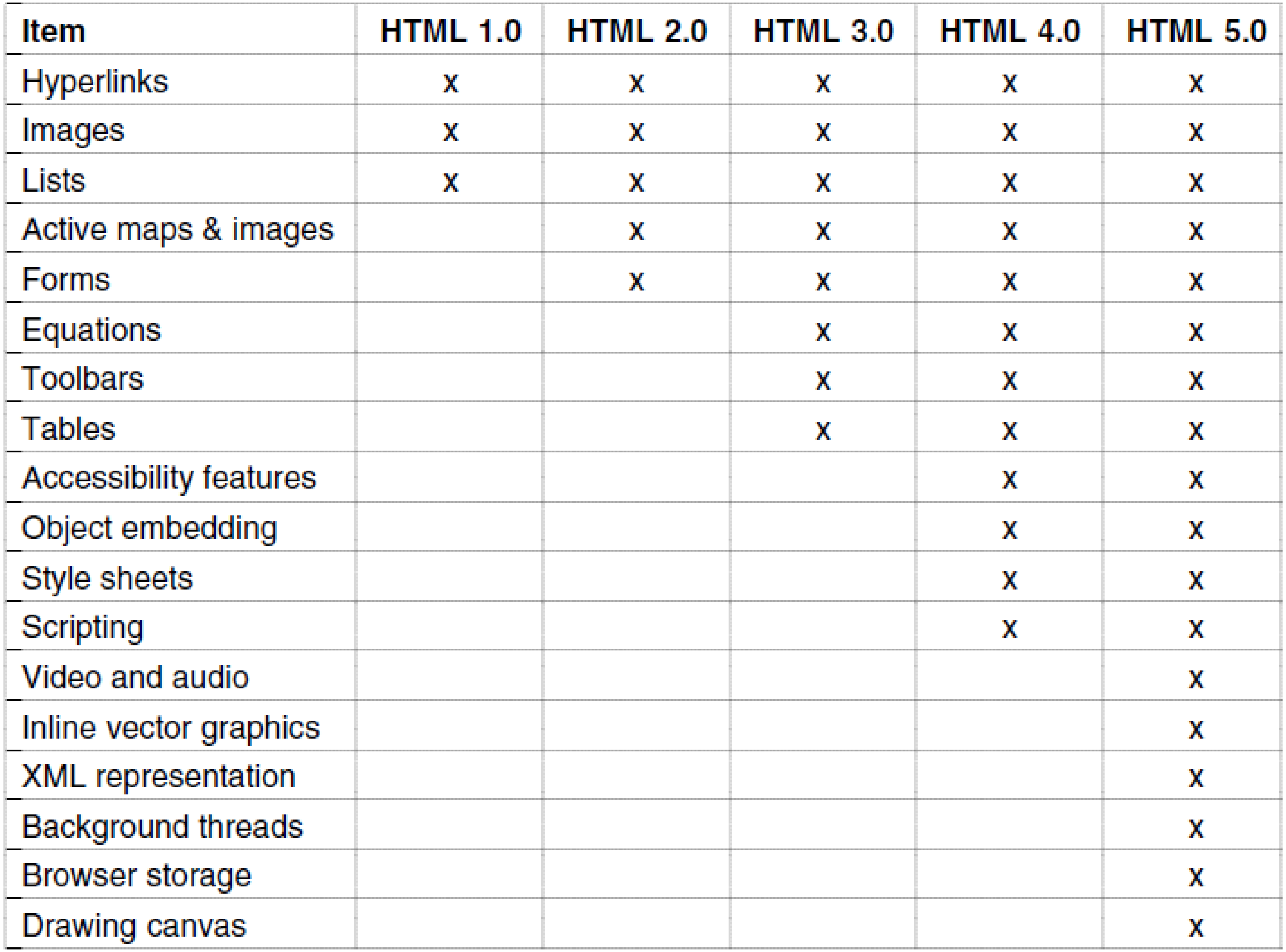
- URL points to one specific host
- Uniform Resource Identifier (URI)
  - Say what you want, not necessarily where from
  - Uniform Resource Locators (URL)
    - http://www.amazon.com/Last-Unicorn-Peter-S-Beagle/dp/0451450523
  - Uniform Resource Name (URN)
    - urn:isbn:0451450523



Web components: HTML

- HyperText Markup Language (HTML)
  - Represents hypertext documents in ASCII form
  - Format text, add images, embed hyperlinks
  - Web browser renders
- Simple and easy to learn
  - Hack up in any text editor
  - Or use a fancy authoring program
- Web page
  - Base HTML file references objects
  - Each object has its own URL

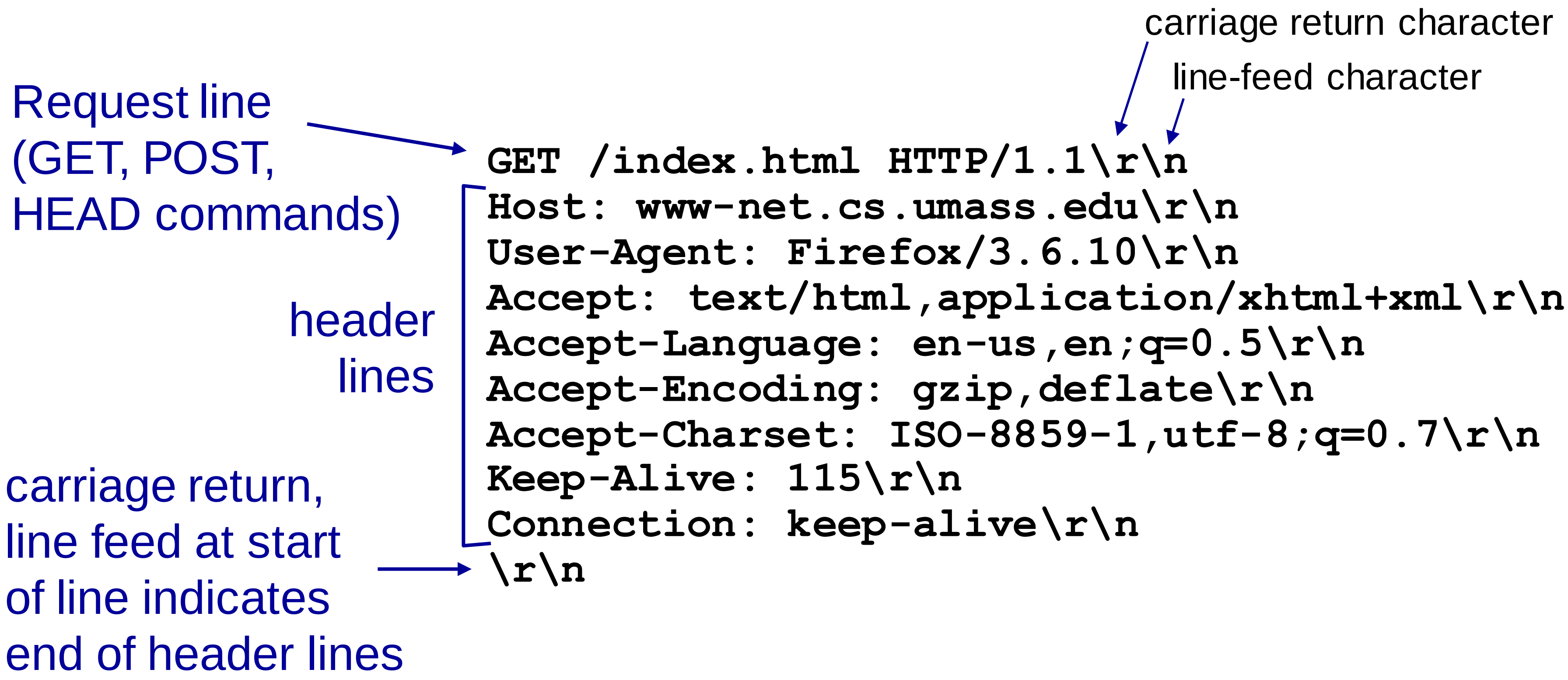
HTML versions



Item	HTML 1.0	HTML 2.0	HTML 3.0	HTML 4.0	HTML 5.0
Hyperlinks	x	x	x	x	x
Images	x	x	x	x	x
Lists	x	x	x	x	x
Active maps & images		x	x	x	x
Forms		x	x	x	x
Equations			x	x	x
Toolbars			x	x	x
Tables			x	x	x
Accessibility features				x	x
Object embedding				x	x
Style sheets				x	x
Scripting				x	x
Video and audio					x
Inline vector graphics					x
XML representation					x
Background threads					x
Browser storage					x
Drawing canvas					x

Web components: HTTP

- HyperText Transfer Protocol (HTTP)
  - Simple request-response protocol
  - Runs over TCP on port 80
  - ASCII format request and response messages
  - A stateless protocol



carriage return character

line-feed character

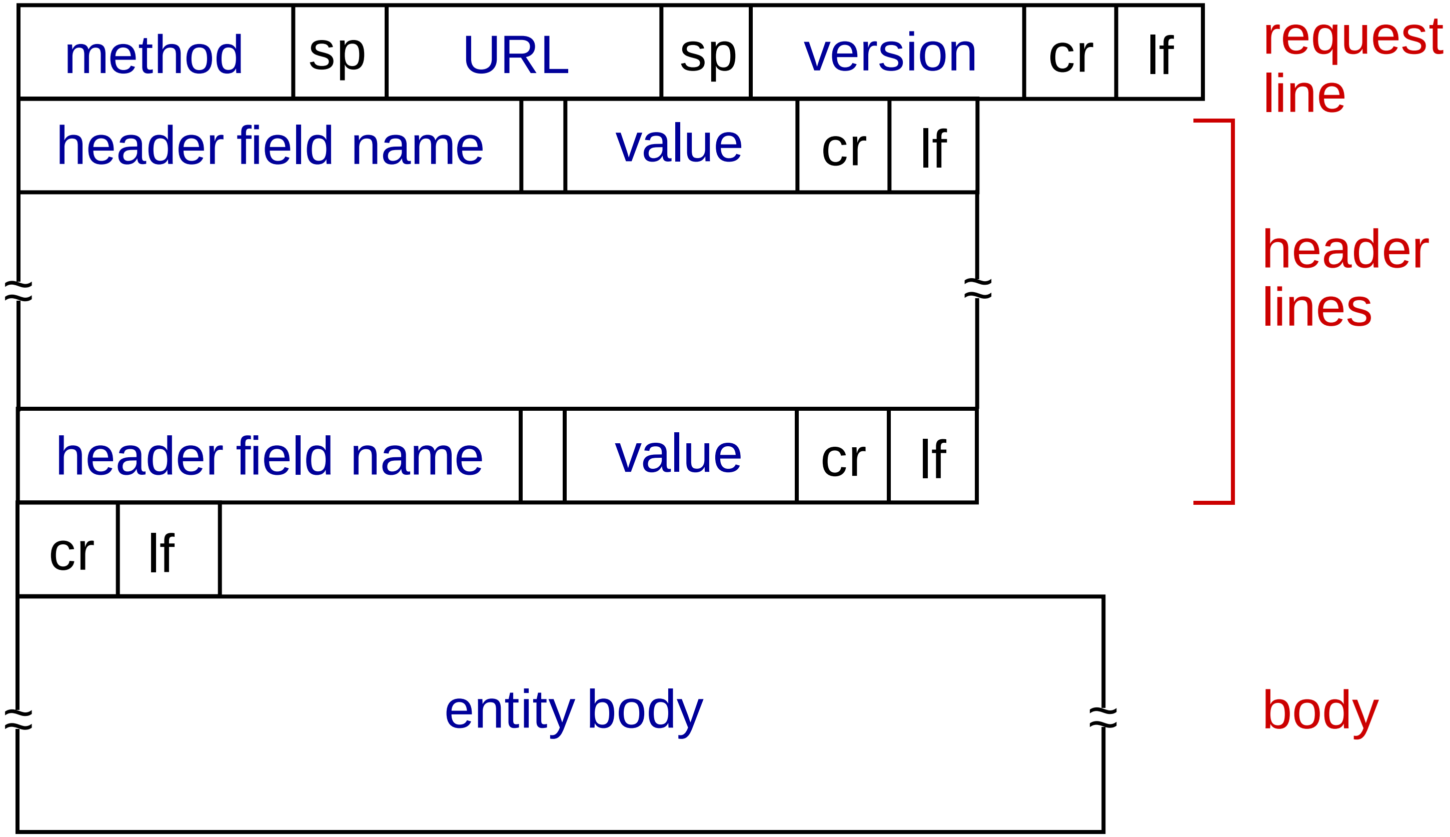
Request line (GET, POST, HEAD commands)

```
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

header lines

carriage return, line feed at start of line indicates end of header lines

HTTP message format



method sp URL sp version cr lf

header field name value cr lf

header field name value cr lf

cr lf

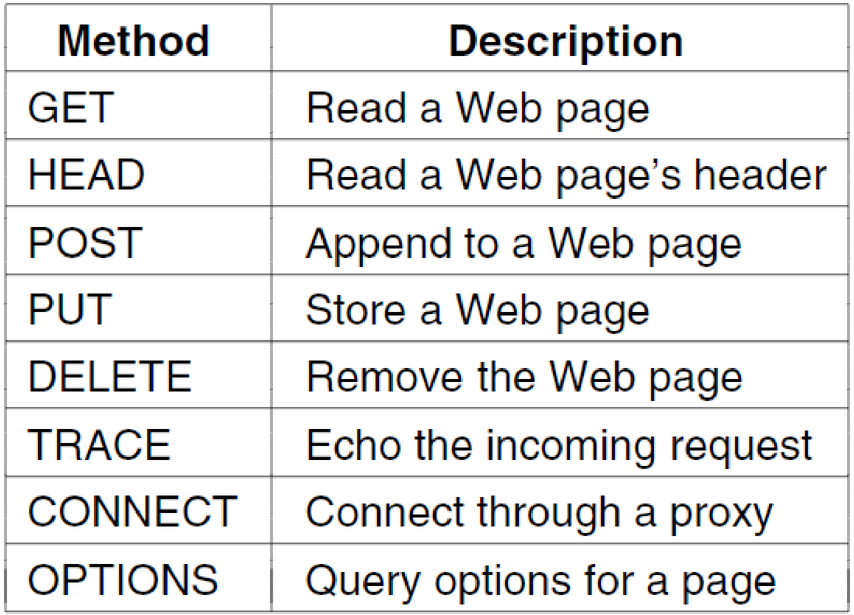
entity body

request line

header lines

body

Request methods

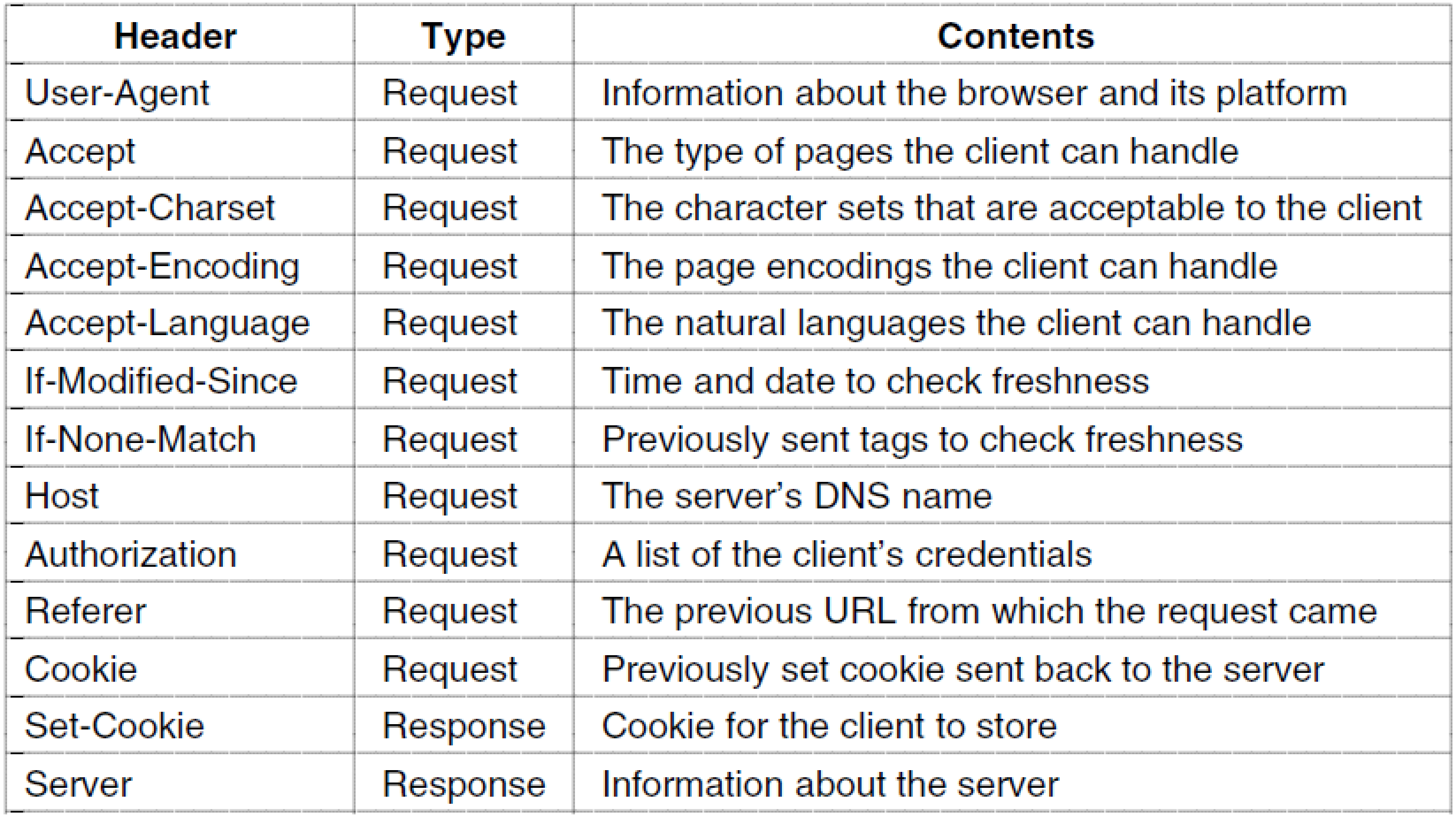


Method	Description
GET	Read a Web page
HEAD	Read a Web page’s header
POST	Append to a Web page
PUT	Store a Web page
DELETE	Remove the Web page
TRACE	Echo the incoming request
CONNECT	Connect through a proxy
OPTIONS	Query options for a page

```
GET /rfc.html HTTP/1.1
Host: www.ietf.org
User-agent: Mozilla/4.0
```

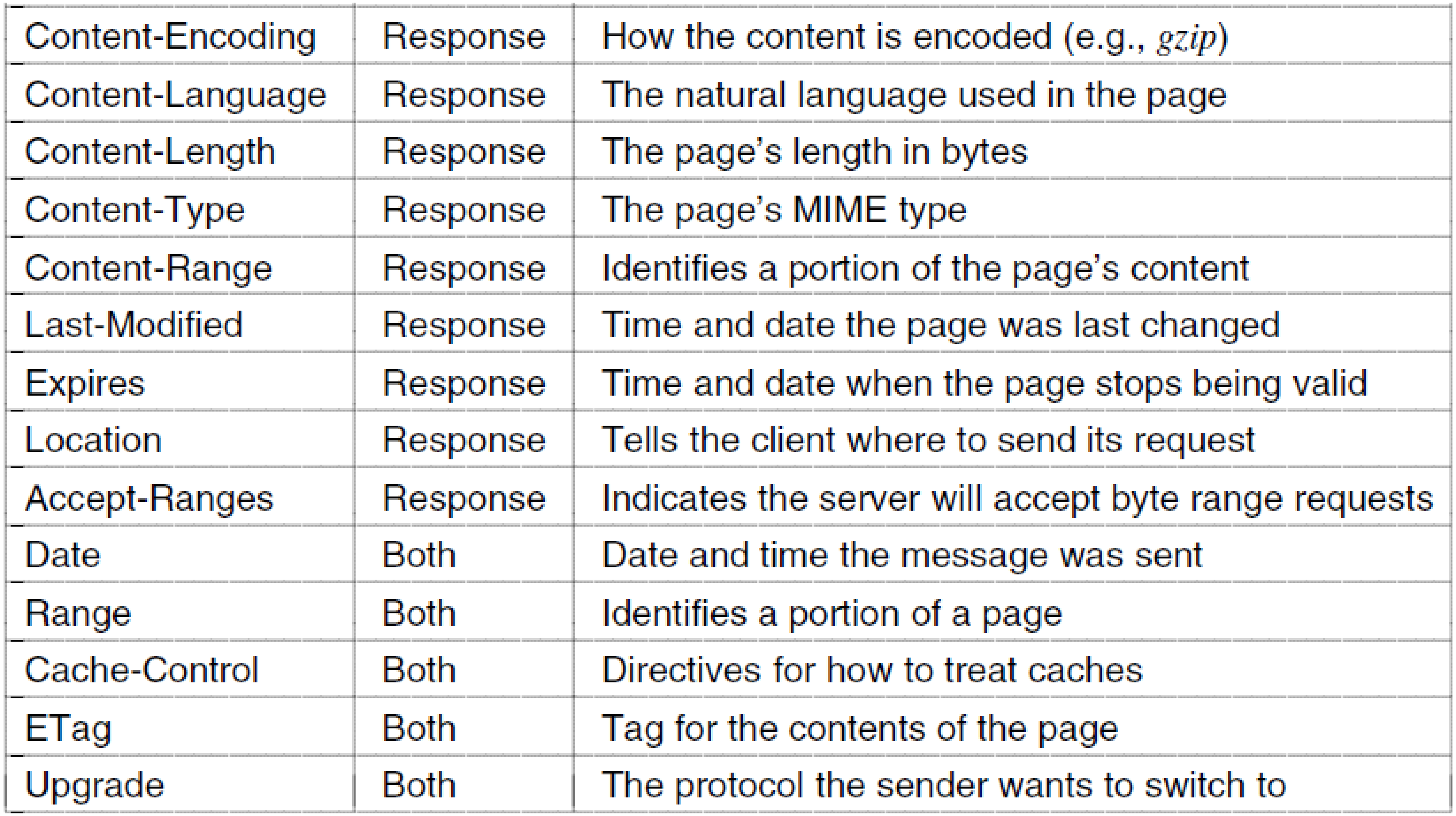
```
POST /login.html HTTP/1.1
Host: www.store.com
User-agent: Mozilla/4.0
Content-Length: 27
Content-Type: application/x-www-form-urlencoded

userid=joe&password=guessme
```

Message headers



Header	Type	Contents
User-Agent	Request	Information about the browser and its platform
Accept	Request	The type of pages the client can handle
Accept-Charset	Request	The character sets that are acceptable to the client
Accept-Encoding	Request	The page encodings the client can handle
Accept-Language	Request	The natural languages the client can handle
If-Modified-Since	Request	Time and date to check freshness
If-None-Match	Request	Previously sent tags to check freshness
Host	Request	The server's DNS name
Authorization	Request	A list of the client's credentials
Referer	Request	The previous URL from which the request came
Cookie	Request	Previously set cookie sent back to the server
Set-Cookie	Response	Cookie for the client to store
Server	Response	Information about the server

Message headers

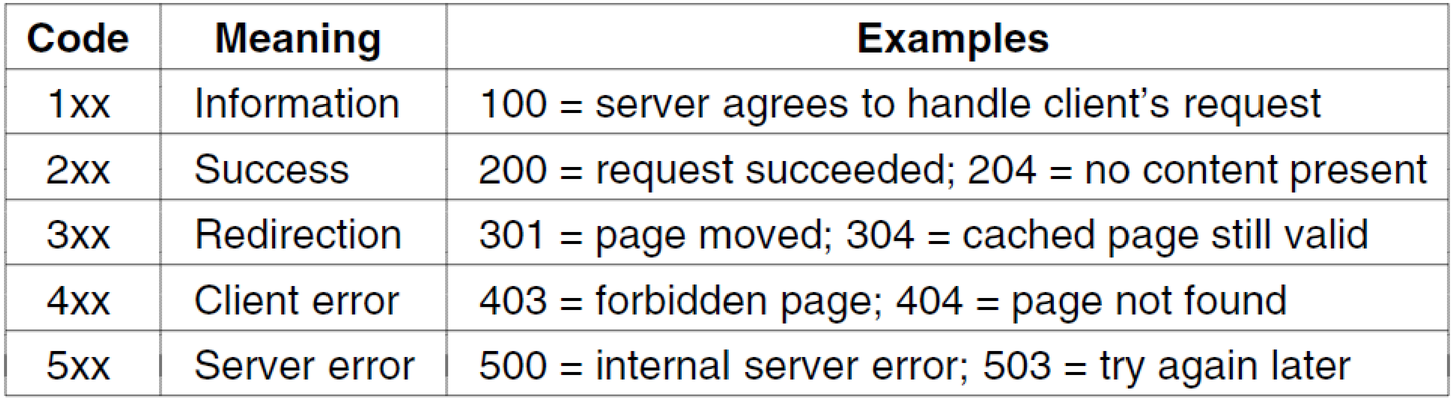


Content-Encoding	Response	How the content is encoded (e.g., *gzip*)
Content-Language	Response	The natural language used in the page
Content-Length	Response	The page's length in bytes
Content-Type	Response	The page's MIME type
Content-Range	Response	Identifies a portion of the page's content
Last-Modified	Response	Time and date the page was last changed
Expires	Response	Time and date when the page stops being valid
Location	Response	Tells the client where to send its request
Accept-Ranges	Response	Indicates the server will accept byte range requests
Date	Both	Date and time the message was sent
Range	Both	Identifies a portion of a page
Cache-Control	Both	Directives for how to treat caches
ETag	Both	Tag for the contents of the page
Upgrade	Both	The protocol the sender wants to switch to

HTTP response

- Response from server
  - Status line:
    - Protocol version, status code, status phrase
  - Response headers: extra info
  - Body: optional data

```
HTTP/1.1 200 OK
Date: Thu, 17 Nov 2011 15:54:10 GMT
Server: Apache/2.2.16 (Debian)
Last-Modified: Wed, 14 Sep 2011 17:04:27 GMT
Content-Length: 285

<html> …
```



Code	Meaning	Examples
1xx	Information	100 = server agrees to handle client’s request
2xx	Success	200 = request succeeded; 204 = no content present
3xx	Redirection	301 = page moved; 304 = cached page still valid
4xx	Client error	403 = forbidden page; 404 = page not found
5xx	Server error	500 = internal server error; 503 = try again later

Summary

- Architectures for network apps
  - Client/server, Peer-to-peer (P2P)
  - Process-to-process communication via sockets
- Services needed by network apps
  - TCP / UDP
- The Worldwide Web
  - History
  - Basic components:
    - HTTP
    - HTML
    - URLs
- Next time: HTTP and web in-depth