上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

Chapter 2

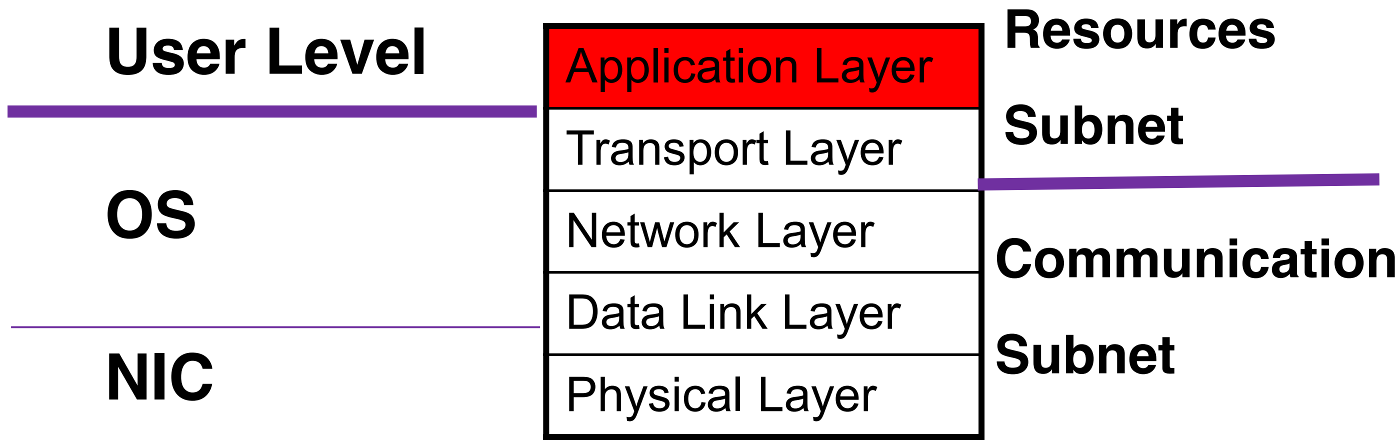
Application layer

Liping Shen 申丽萍

lpshen@sjtu.edu.cn

These slides are based upon the exceptional slides provided by Kurose and Ross

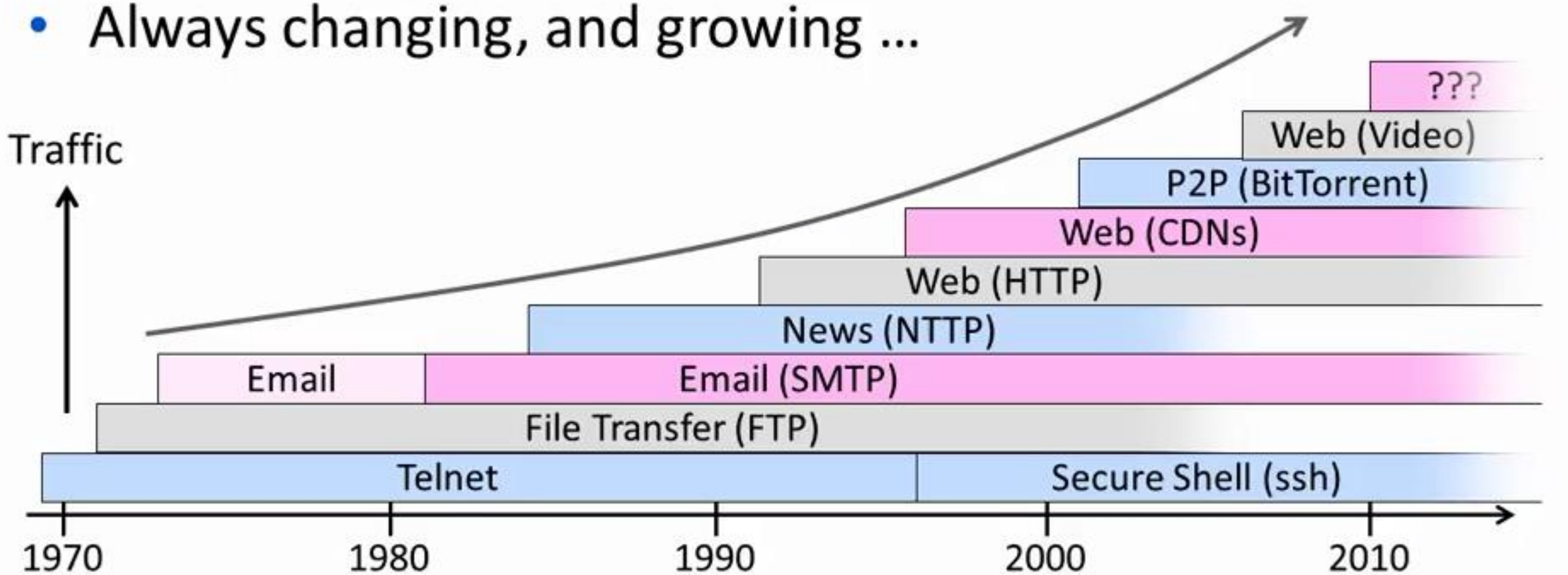
The Application Layer in the Hybrid Model



- Provide functions needed by users
- Run on end-systems instead of network core
- Convert different representations: MIME, gzip, encryption etc.
- Manage task dialogs: a series of related network interactions.

Evolution of Internet Applications

- Always changing, and growing …



Chapter 2: Goals

Our goals:

- conceptual, implementation aspects of network application protocols
  - transport-layer service models
  - client-server paradigm
  - peer-to-peer paradigm
- learn about protocols by examining popular application-level protocols
  - DNS
  - HTTP
  - FTP
  - SMTP / POP3 / IMAP
- programming network applications
  - socket API

Chapter 2: Topics

2.1 Principles of network applications

2.2 Web and HTTP

2.3 FTP

2.4 Electronic Mail

- SMTP, POP3, IMAP

2.5 DNS

2.6 P2P applications

2.7 Socket programming with TCP

2.8 Socket programming with UDP

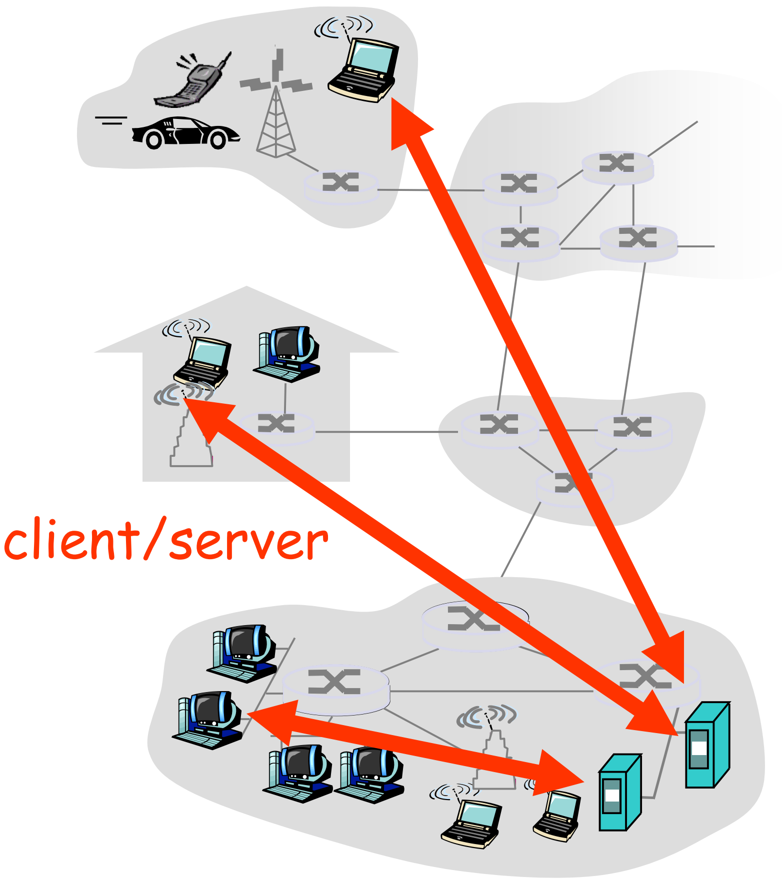
Chapter 2: Roadmap

- Principles of network applications
- DNS
- Web and HTTP
- FTP
- Electronic Mail
  - SMTP, POP3, IMAP
- P2P applications
- Socket programming with TCP
- Socket programming with UDP

Application architectures

- client-server
- peer-to-peer (P2P)
- hybrid of client-server and P2P

Client-server architecture



server:

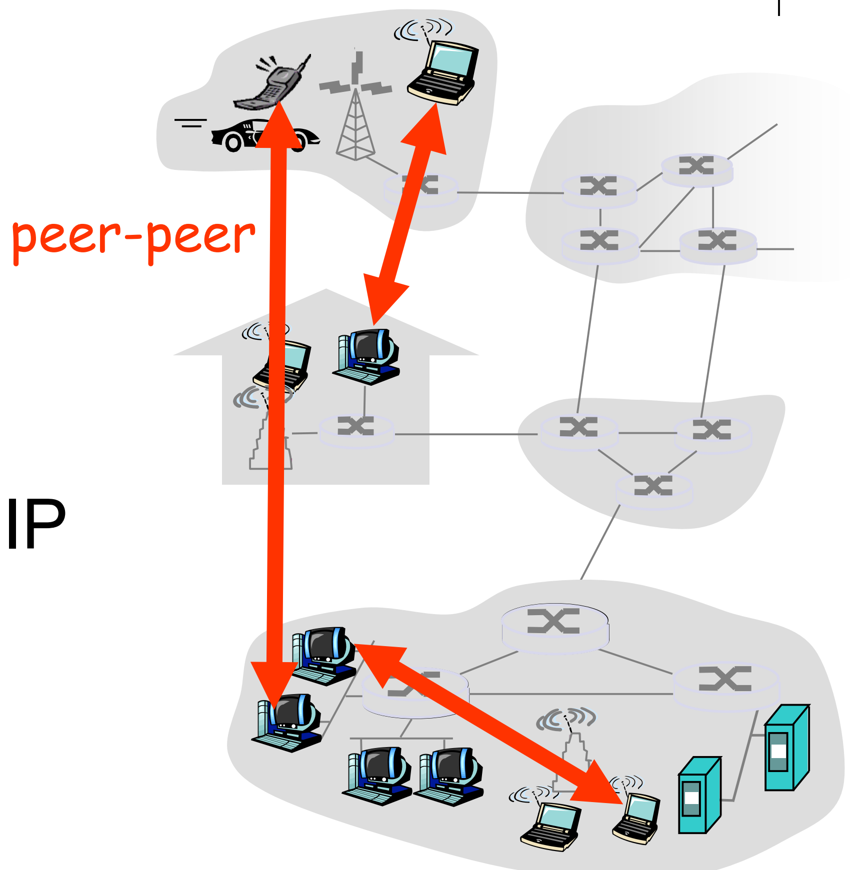
- always-on host
- permanent IP address
- server farms for scaling

clients:

- communicate with server
- may be intermittently connected
- may have dynamic IP addresses
- do not communicate directly with each other

Pure P2P architecture

- *no* always-on server
- arbitrary end systems directly communicate
- peers are intermittently connected and change IP addresses



P2P model

cost effective, highly scalable but difficult to manage

Hybrid of client-server and P2P

Skype
- voice-over-IP P2P application
- centralized server: finding address of remote party:
- client-client connection: direct (not through server)

Instant messaging
- chatting between two users is P2P
- centralized service: client presence detection/location
  - user registers its IP address with central server when it comes online
  - user contacts central server to find IP addresses of buddies

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

Client/Server Model:

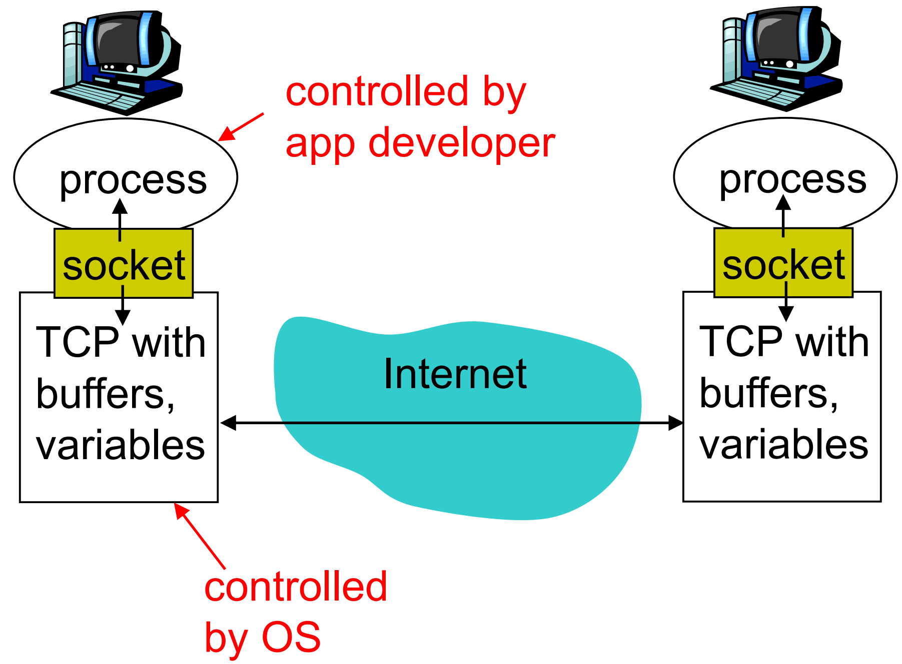
- client process: process that initiate communication
- server process: process that waits to be contacted

P2P Model:

- peer applications have both client processes & server processes

Sockets

- process sends/receives messages to/from its socket (API)
- Sockets rely on transport infrastructure
- limited control by developer
  - choice of transport protocol;
  - ability to fix a few parameters



What transport service does an app need?

Data loss

- some apps (e.g., audio) can tolerate some loss
- other apps (e.g., file transfer, telnet) require 100% reliable data transfer

Delay

- some apps (e.g., Internet telephony, interactive games) require low delay to be "effective"

Throughput

- some apps (e.g., multimedia) require minimum amount of throughput to be "effective"
- other apps ("elastic apps") make use of whatever throughput they get

Security

- encryption, data integrity, …

Transport service requirements

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
instant messaging	no loss	elastic	yes and no

Internet transport services

TCP service:

- *connection-oriented:* setup required between client and server processes
- *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 guarantees, security

UDP service:

- exposes *packet-switched* nature of Internet
- *unreliable data transfer* between sending and receiving process
- *does not provide*: connection setup, reliability, flow control, congestion control, timing, throughput guarantee, or security

applications and 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)	typically UDP
domain name system	DNS [RFC 1305]	UDP TCP: low efficiency

Identifying Processes

- to receive messages, process must have *identifier*
- host device has unique 32-bit IP address
- *Q:* does IP address of host suffice for identifying the process?
  - *A:* No, *many* processes can be running on same host
- *process identifier* = IP address + port numbers
  - Example: to send HTTP message to SJTU web server:
    - IP address: 202.120.2.102
    - 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
  - meaning of information in fields
- rules for when and how processes send & respond to messages

public-domain protocols:

- defined in RFCs
- allows for interoperability
- e.g., HTTP, SMTP

proprietary protocols:

- e.g., Skype

Chapter 2: Roadmap

- Principles of network applications
- DNS
- Web and HTTP
- FTP
- Electronic Mail
  - SMTP, POP3, IMAP
- P2P applications
- Socket programming with TCP
- Socket programming with UDP

Name and Address

People many identifiers:

- SSN, name, passport #

Hosts, routers identifiers:

- address – lower level, used by machine
  - IP address, e.g. 202.120.2.102
  - MAC address, e.g. 1C-3E-84-66-77-88
- “name” – higher level , used by humans
  - e.g. www.sjtu.edu.cn

Resolution: map between IP addresses and name.

- www.sjtu.edu.cn --> 202.120.2.119

DNS: Domain Name System RFC1304/1305

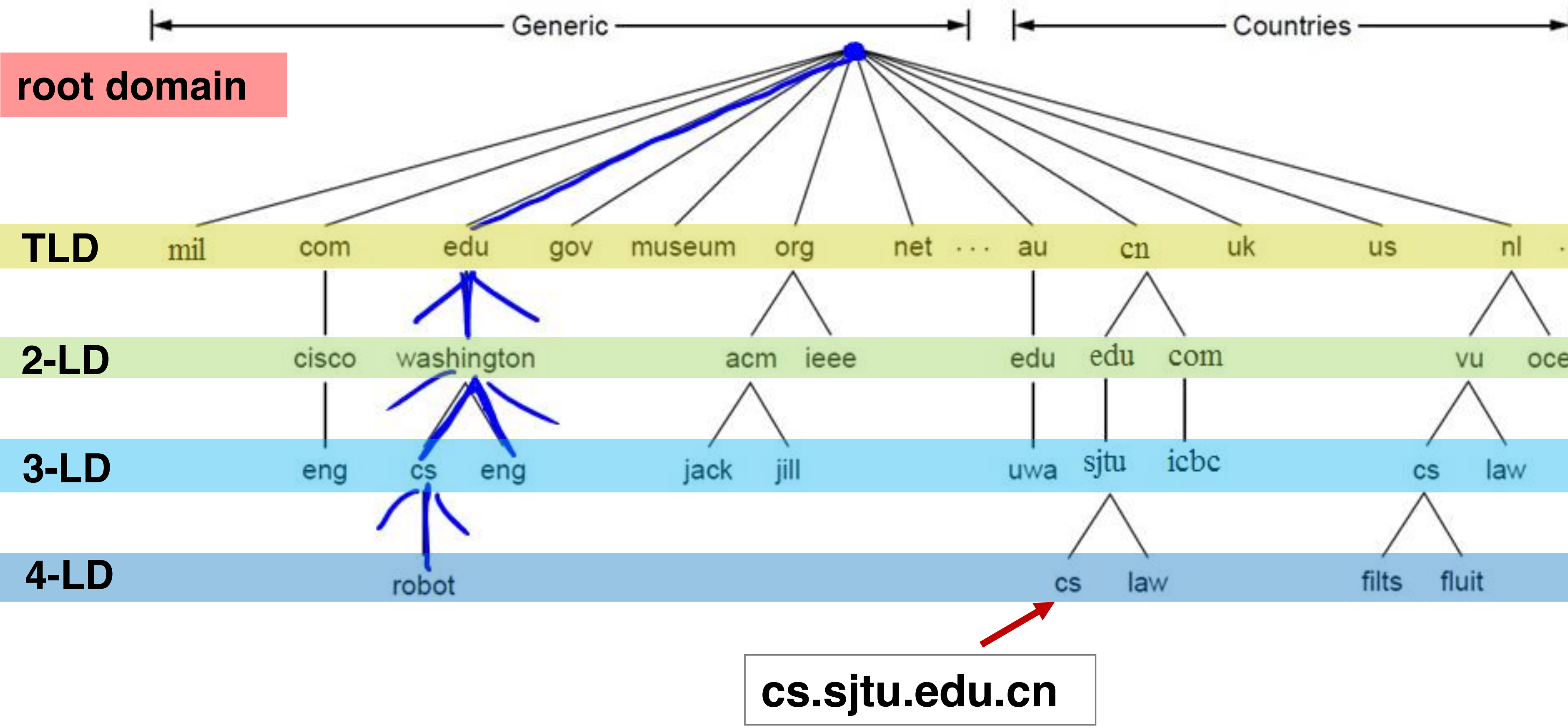
Before DNS

- Using file for resolution (hosts.txt)
- flat, un-scalable, single node failure

Domain Name System (1985~)

- *distributed database* implemented in hierarchy of many *name servers*
- *application-layer protocol* host, routers, name servers to communicate to *resolve* names (address/name translation)
- UDP on port 53

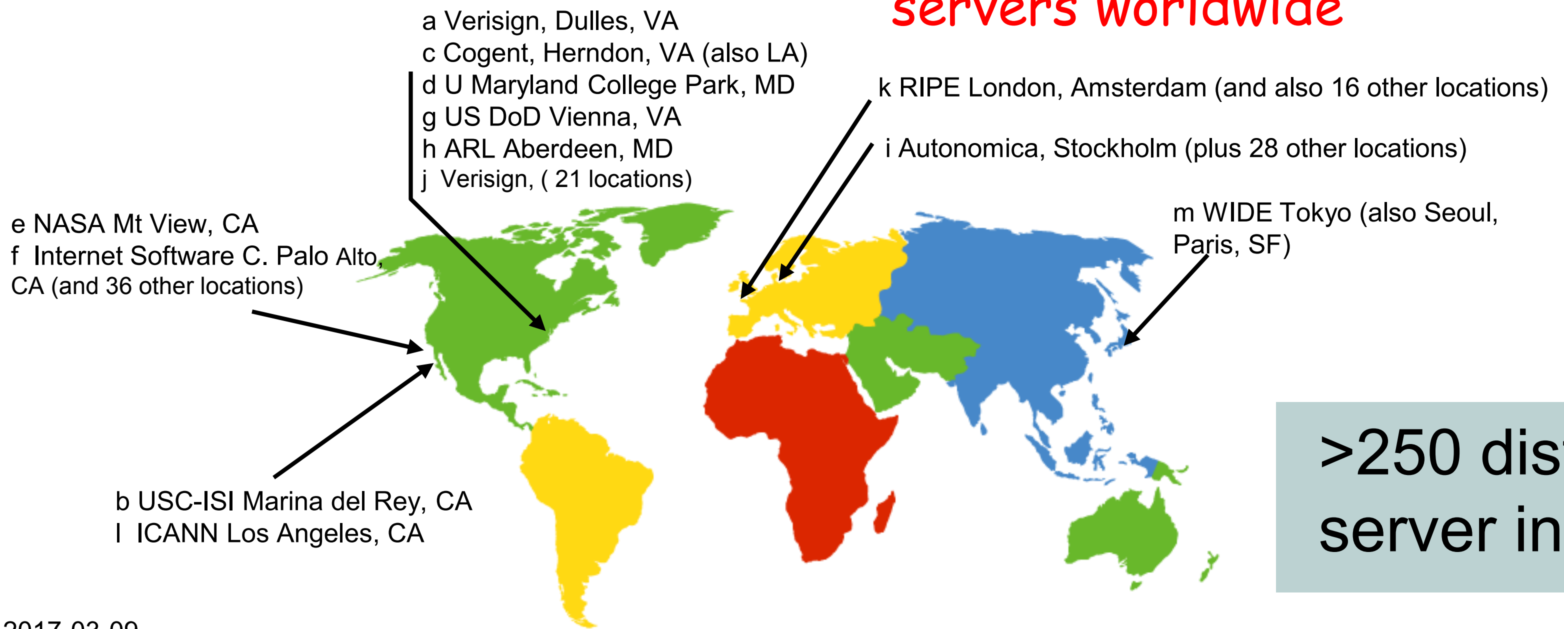
Distributed, Hierarchical Database



DNS: root name servers

- Contacted by local name server that can not resolve name
  - contacts authoritative name server if name mapping not known
  - gets mapping
  - returns mapping to local name server
- Every local name server needs to be configured with root name servers

13 root name servers worldwide



>250 distributed server instances

DNS: root name servers

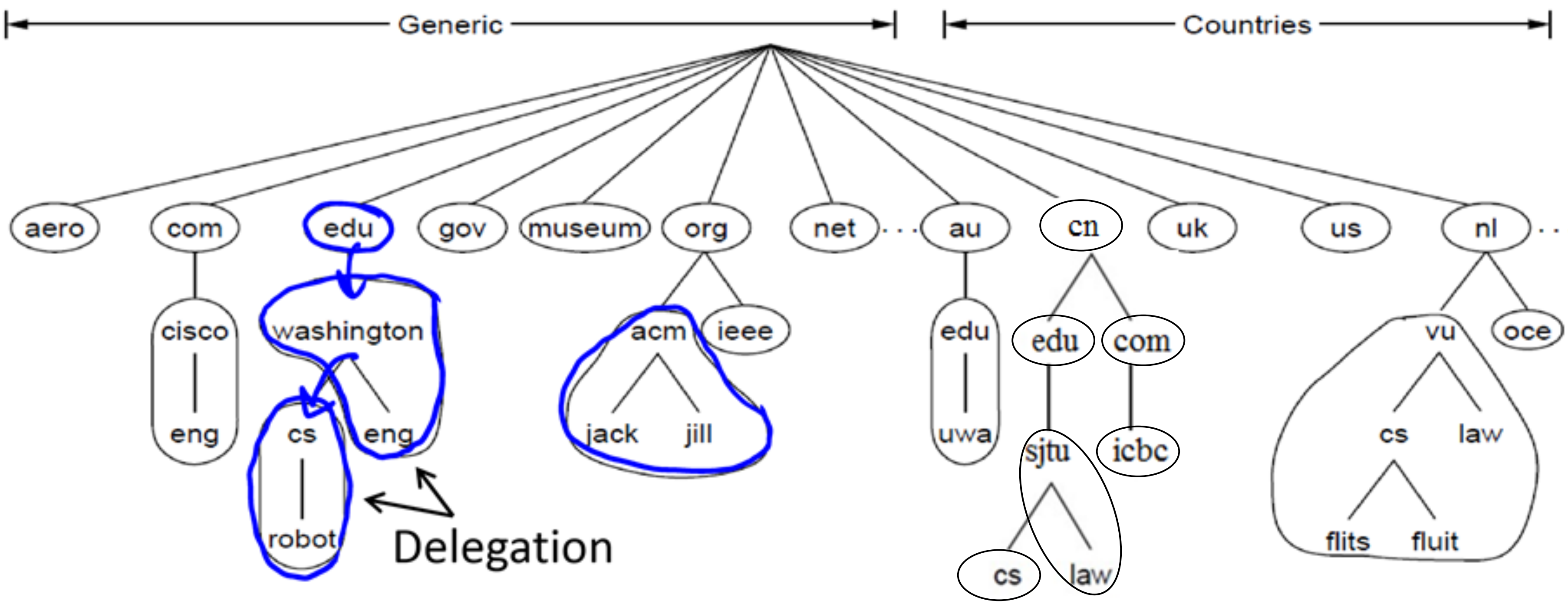
More than 250 distributed server instances reached by IP anycast

TLDs (Top-Level Domains)

- Run by ICANN (Internet Corp. for Assigned Names and Numbers)
- 22+ generic TLDs
  - Initially .com, .edu, .gov., .mil, .org, .net
  - Added .aero, .museum, etc.
  - Different TLDs have different usage policies
- ~250 country code TLDs
  - Two letters, e.g., “.cn”,
  - International characters since 2010，e.g. “.中国”

Zones and Authoritative Servers

- Divide the namespace into a collection of non-overlapping zones, and let each zone be taken care of by one or more name servers (Authority of that zone).



Local Name Server

- Does not strictly belong to hierarchy
- Could be gateway or AP or ISP host or public DNS (e.g. google DNS).
  - also called “default name server”
- Client needs to be configured with local name server manually or via DHCP.
- When host makes DNS query, query is sent to its local DNS server
  - acts as proxy, forwards query into hierarchy

DNS Resource Records

- Distributed DBs(zone authorative servers) store resource records (RR) that give information for its domain names.

RR format: `(name, ttl, type, value)`

Type	Associated entity	Description
SOA	Zone	Holds information on the represented zone
A	Host	IPv4 address of a host
AAAA	Host	IPv6 address of a host
MX	Domain	Refers to a mail server to handle mail addressed to this node
NS	Zone	Refers to a name server that implements the represented zone
CNAME	Node	Symbolic link with the primary name of the represented node
PTR	Host	Contains the canonical name of a host

DNS Resource Records Example

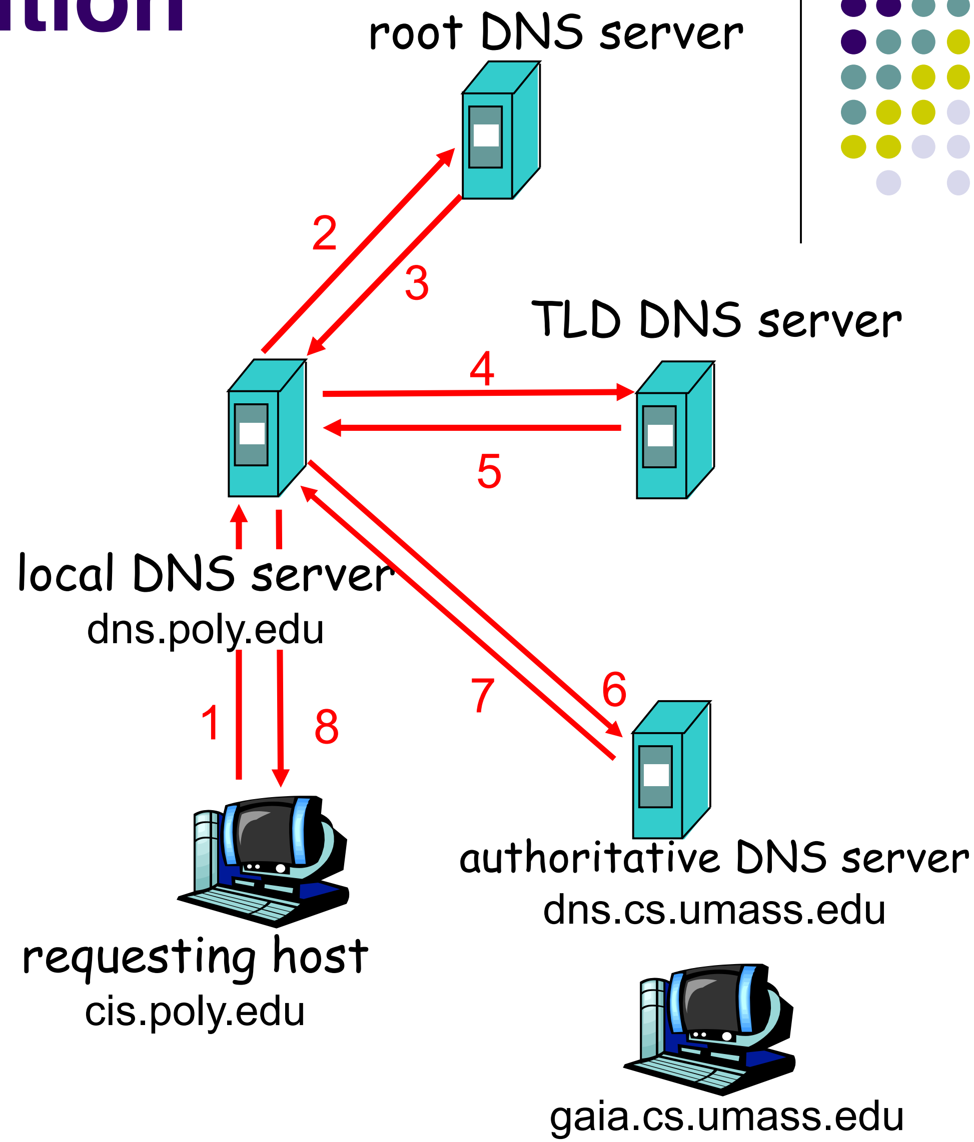
a uthoritative data for cs.vu.nl

cs.vu.nl.	86400	IN	SOA	star boss (9527,7200,7200,241920,86400)	
cs.vu.nl.	86400	IN	MX	1 zephyr	
cs.vu.nl.	86400	IN	MX	2 top	
cs.vu.nl.	86400	IN	NS	star	Name server
star	86400	IN	A	130.37.56.205	
zephyr	86400	IN	A	130.37.20.10	
top	86400	IN	A	130.37.20.11	IP addresses of computers
www	86400	IN	CNAME	star.cs.vu.nl	
ftp	86400	IN	CNAME	zephyr.cs.vu.nl	
flits	86400	IN	A	130.37.16.112	
flits	86400	IN	A	192.31.231.165	
flits	86400	IN	MX	1 flits	
flits	86400	IN	MX	2 zephyr	
flits	86400	IN	MX	3 top	
rowboat		IN	A	130.37.56.201	
		IN	MX	1 rowboat	
		IN	MX	2 zephyr	Mail gateways
little-sister		IN	A	130.37.62.23	
laserjet		IN	A	192.31.231.216	

DNS name resolution

iterated query

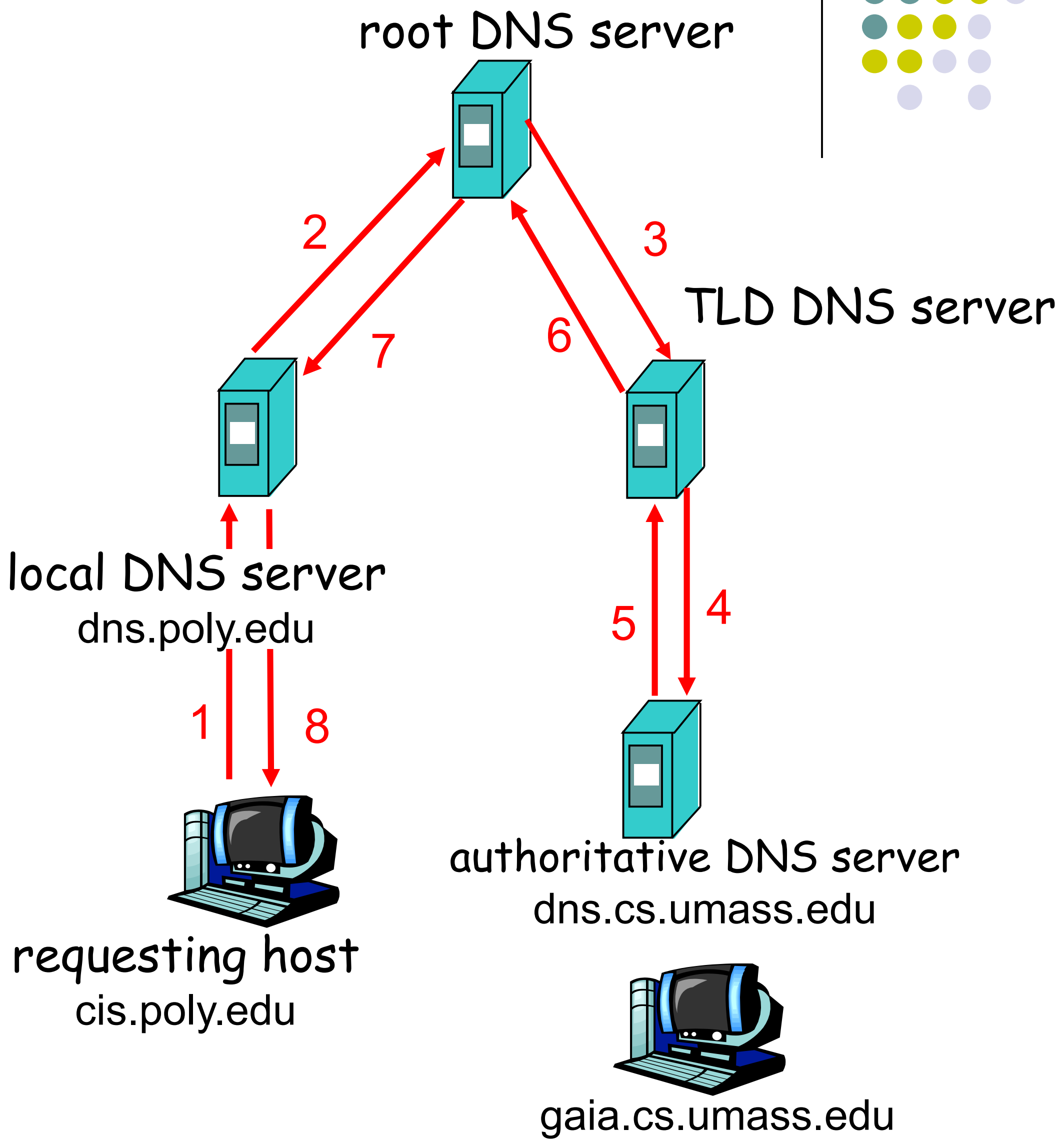
- contacted server replies with name of server to contact.
- "I don't know this name, but ask this server".
- server file and forget, easy to build high load servers.



DNS name resolution

recursive query

- puts burden of name resolution on contacted name server.
- once (any) name server learns mapping, it *caches* mapping for better performance.

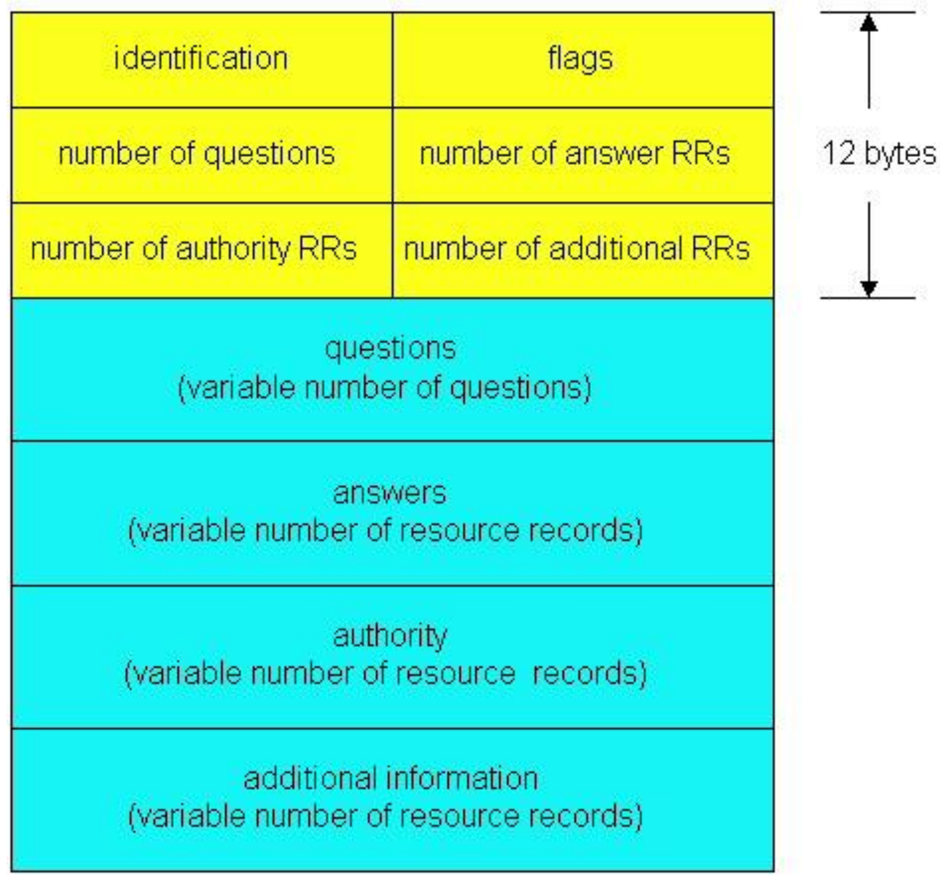


DNS protocol

- Query and response messages
  - Built on UDP messages, port 53
  - ARQ for reliability; server is stateless!
  - Messages linked by a 16-bit ID field
- Service reliability via replicas
  - Run multiple nameservers for domain

DNS protocol, messages

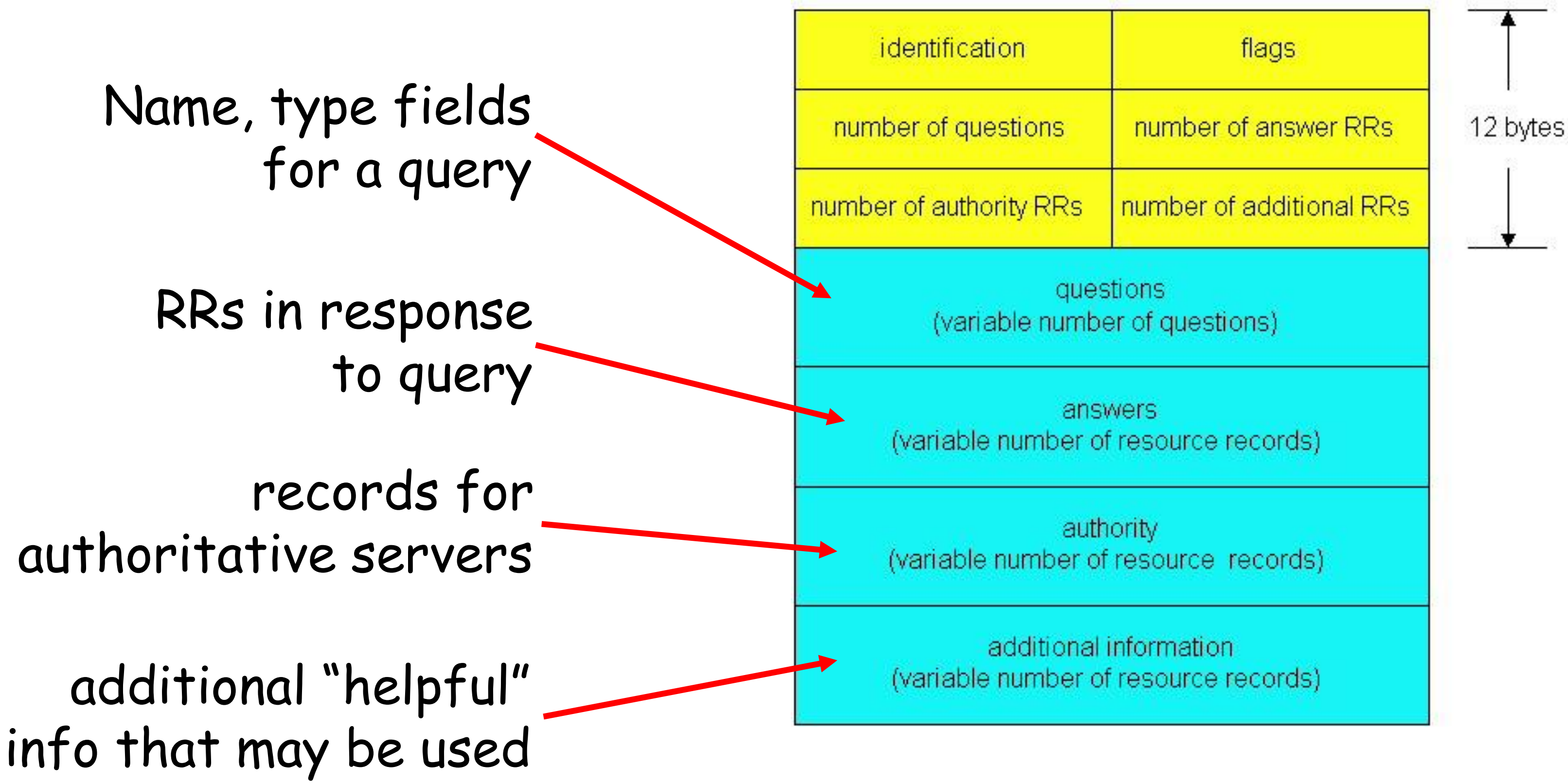
DNS protocol : *query* and *reply* messages, both with same *message format*



msg header

- identification: 16 bit # for query, reply to query uses same #
- flags:
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative

DNS protocol, messages



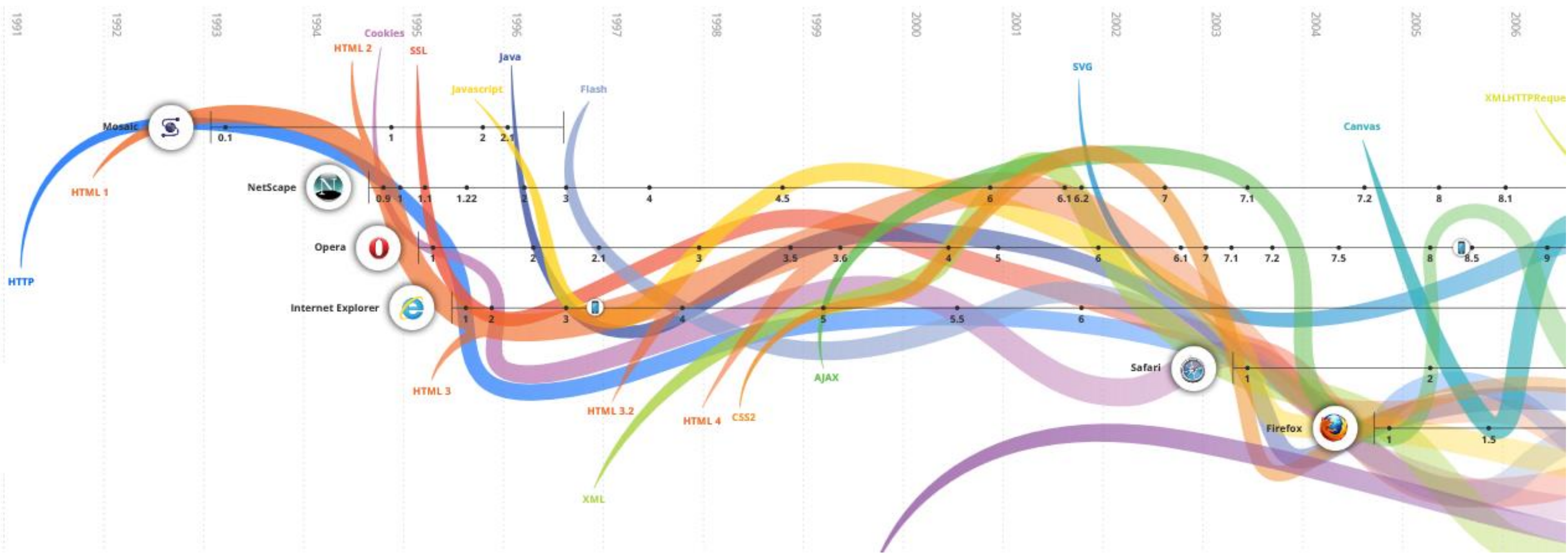
DNS Security

- Security is a major issue
  - DNS can be attacked in several Ways
    - Man-in-the-middle attack redirects to wrong site!
  - Not part of initial protocols .
- DNSSEC (DNS Security Extensions)
  - Long under development, now partially deployed.

Chapter 2: Roadmap

- Principles of network applications
- DNS
- Web and HTTP
- FTP
- Electronic Mail
    - SMTP, POP3, IMAP
- P2P applications
- Socket programming with TCP
- Socket programming with UDP

Evolution of the Web



Source: http://www.evolutionoftheweb.com

World wide web

First, a review…

- The World Wide Web consists of a vast, worldwide collection of documents or web page which consists of multiple objects
- object can be HTML file, JPEG image, Java applet, audio file,…
- web page consists of base HTML-file which includes several embedded or referenced objects (hyperlink)
- each object is addressable by a URL

protocol://computer_name:port/document_name

Examples: http://www.sjtu.edu.cn/index.html

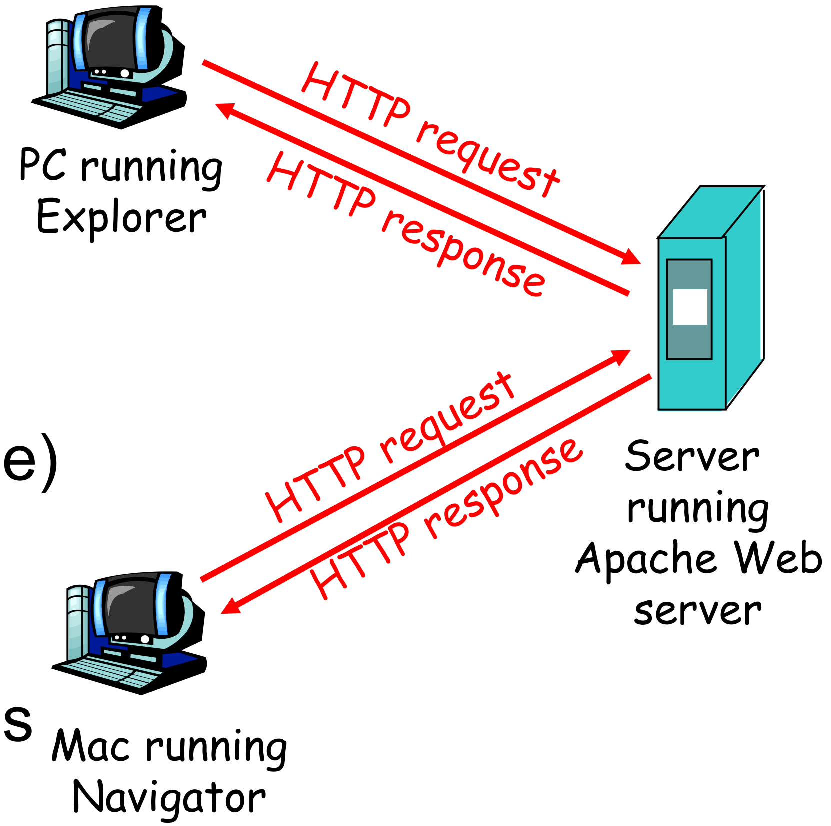
ftp://ftp.cs.sjtu.edu.cn/shen-lp/CompuNet/chap1.pdf

mailto: lpshen@sjtu.edu.cn

WWW--client/server model

Client/server interaction with http

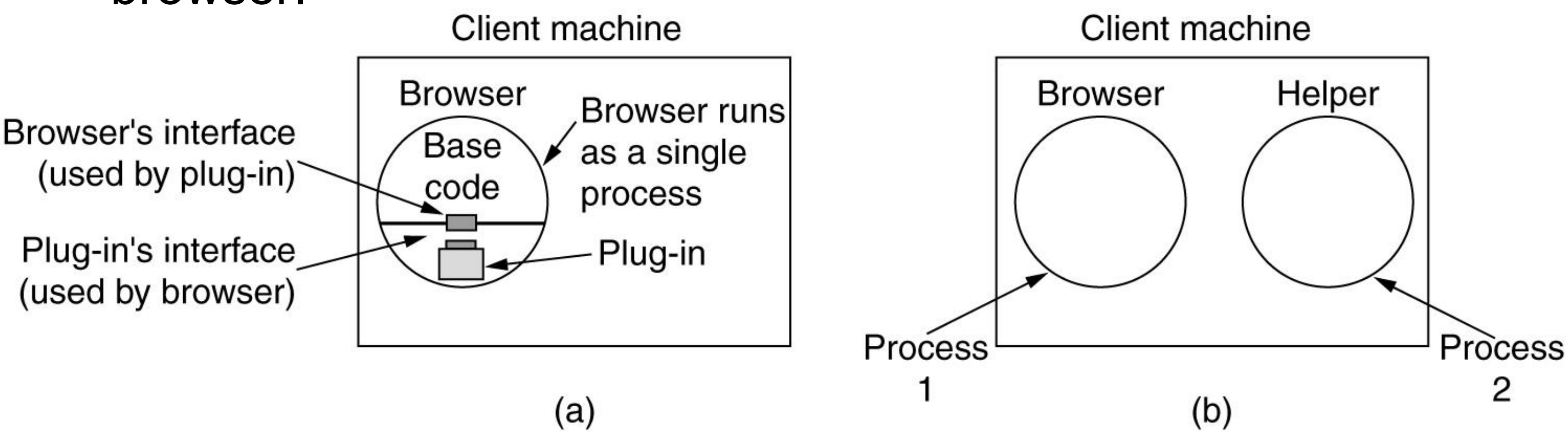
- Resolve the server to IP address (DNS)
- Set up TCP connection
- Send HTTP request for the page
- (Await HTTP response for the page)
- Execute / fetch embedded resources/render
- Clean up any idle TCP connections



The Client Side

Multipurpose Internet Mail Extensions

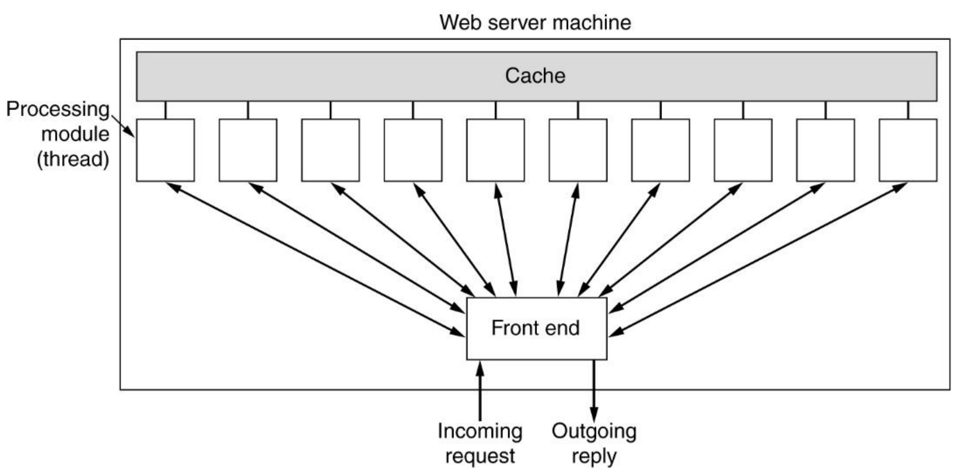
- A browser can interpret built-in MIME types such as text/html.
- When encountering a page it can't understand, the browser consults a table which associates a MIME type with a viewer how to display the page.
- There are two kinds of viewers:
  - **Plug-ins** run inside the browser. After the plug-in has done its job it is removed from the browser's memory.
  - **Helpers** are large programs that exist independently of the browser.



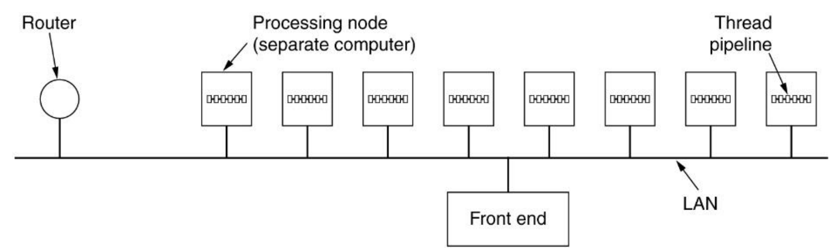
(a) A browser plug-in. (b) A helper application.

The Server Side

- Most servers need to process many incoming requests. Disk access time and CPU processing will limit the server capacity. Solutions:
  - Cache to eliminate the disk access
  - multithreaded servers with multiple disks sprays requests over multiple threads and multiple disks.
  - **Server farms** sprays requests over multiple CPUs**.**



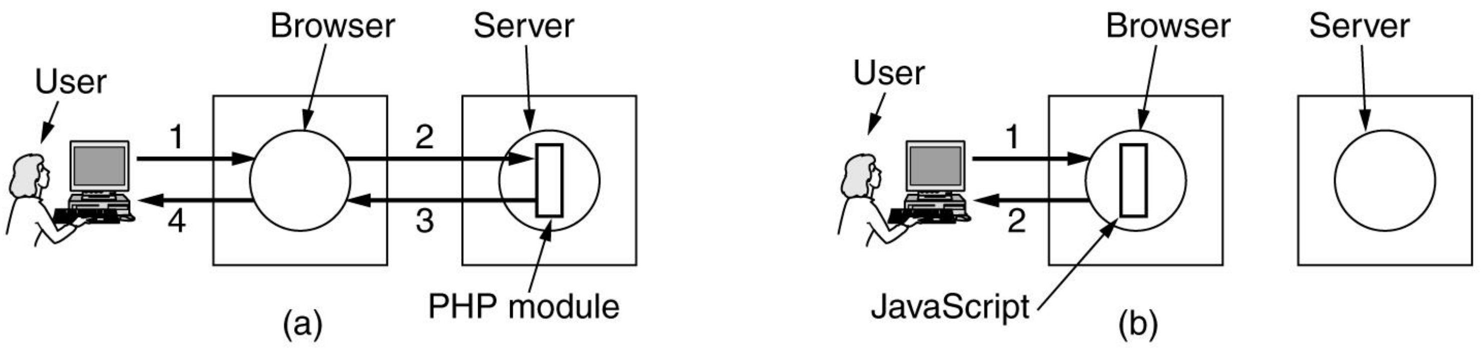
A multithreaded Web server



A Server Farm

Static and Dynamic Web Pages

- Static Web Documents: Web pages are established when authoring (e.g. html)
- Dynamic Web Documents： Web pages are generated by program execution either at server-side (e.g. PHP, ASP, JSP) or client-side (e.g. JavaScript, applet, ActiveX controls)



(a) Server-side scripting with PHP. (b) Client-side scripting with JavaScript.

HTTP (HyperText Transfer Protocol)

Uses TCP:

- client initiates TCP connection (creates socket) to server, port 80
- server accepts TCP connection from client
- HTTP 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 re-mended.

HTTP request message

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

carriage return character

line-feed character

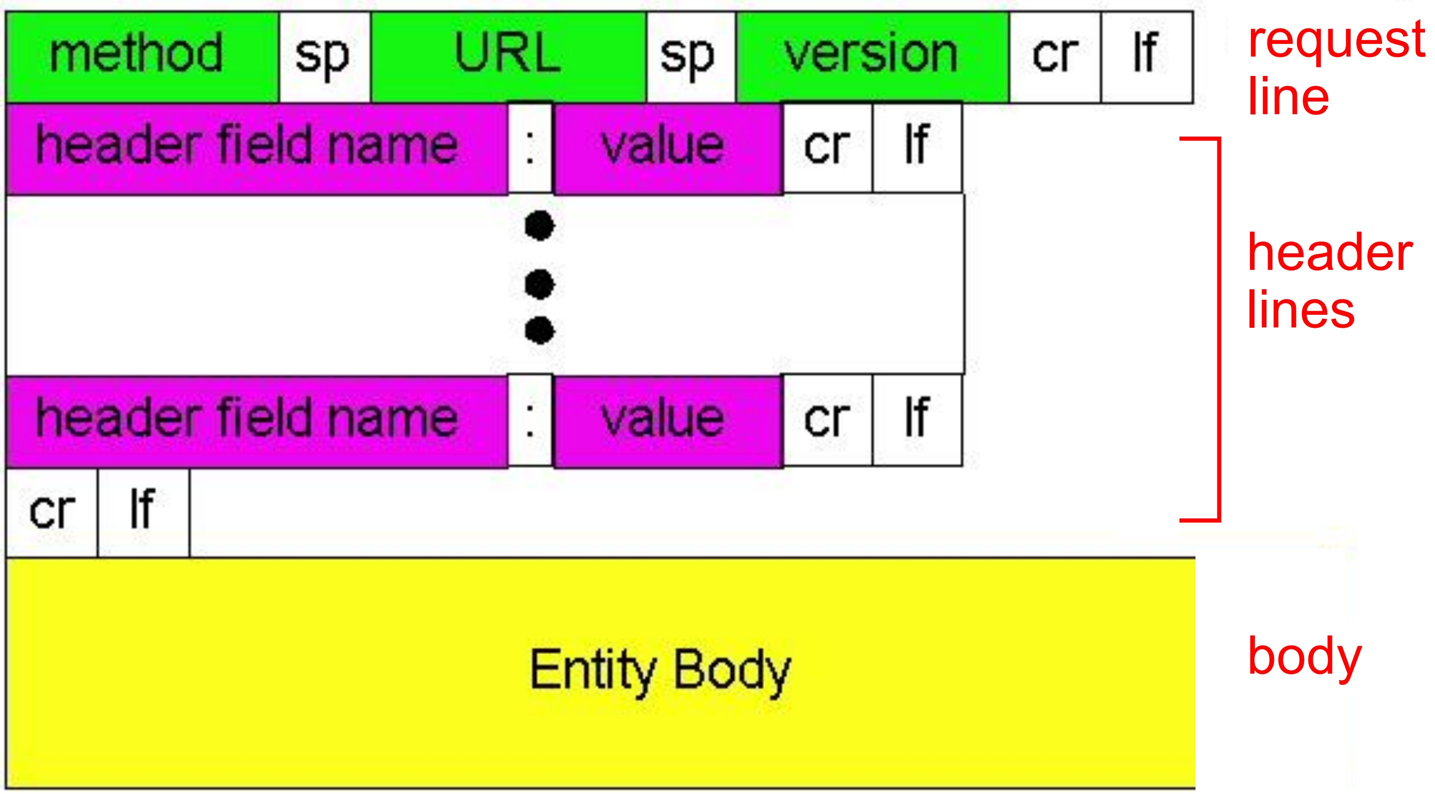
request line (GET, POST, HEAD commands)

header lines

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

```
GET /index.html HTTP/1.1\r\n
Host: www.sjtu.edu.cn\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
```

HTTP request message: general format



Method types

HTTP/1.0

- GET
- POST
- HEAD

HTTP/1.1

- GET, POST, HEAD
- PUT
- DELETE

Method	Description
GET	Request to read a Web page
HEAD	Request to read a Web page's header
PUT	Request to store a Web page
POST	Append to a named resource (e.g., a Web page)
DELETE	Remove the Web page
TRACE	Echo the incoming request
CONNECT	Reserved for future use
OPTIONS	Query certain options

Uploading user input

POST method:

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

URL method:

- uses GET method
- input is uploaded in URL field of request line:
- Example: search “weather” in Baidu.com:

the URL to the server :
http://www.baidu.com/s?wd=weather&rsv_bp=0&rsv_spt=3&inputT=8768

HTTP response message

status line (protocol status code status phrase)

header lines

data, e.g., requested HTML file

```
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 ...
```

HTTP response status codes

- Every request gets a response consisting of a status line, and possibly additional information.
- The status line contains a three-digit status code telling whether the request was satisfied, and if not, why not.

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

HTTP 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
Host	Request	The server's DNS name
Authorization	Request	A list of the client's credentials
Cookie	Request	Sends a previously set cookie back to the server
Date	Both	Date and time the message was sent
Upgrade	Both	The protocol the sender wants to switch to
Server	Response	Information about the server
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
Last-Modified	Response	Time and date the page was last changed
Location	Response	A command to the client to send its request elsewhere
Accept-Ranges	Response	The server will accept byte range requests
Set-Cookie	Response	The server wants the client to save a cookie

Trying out HTTP (client side) for yourself

1. Telnet to your favorite Web server:

```
telnet cis.poly.edu 80
```

opens TCP connection to port 80 (default HTTP server port) at cis.poly.edu. anything typed in sent to port 80 at cis.poly.edu

2. type in a GET HTTP request:

```
GET /~ross/ HTTP/1.1
Host: cis.poly.edu
```

by typing this in (hit carriage return twice), you send this minimal (but complete) GET request to HTTP server

3. look at response message sent by HTTP server!

(or use Wireshark !)

Cookies- keeping "state" (RFC 2965)

- The Web is stateless: servers do not keep track of their clients. However, state may be desirable in many cases.
- Solution: a server drops a **cookie** at the client side containing server state relevant for that client. Every time the client access the server, includes all the cookies set by that server.
- Cookies are just files or strings, may contain up to five fields.

Domain	Path	Content	Expires	Secure
toms-casino.com	/	CustomerID=497793521	15-10-02 17:00	Yes
joes-store.com	/	Cart=1-00501;1-07031;2-13721	11-10-02 14:22	No
aportal.com	/	Prefs=Stk:SUNW+ORCL;Spt:Jets	31-12-10 23:59	No
sneaky.com	/	UserID=3627239101	31-12-12 23:59	No

Some examples of cookies.

Cookies: keeping "state" (cont.)

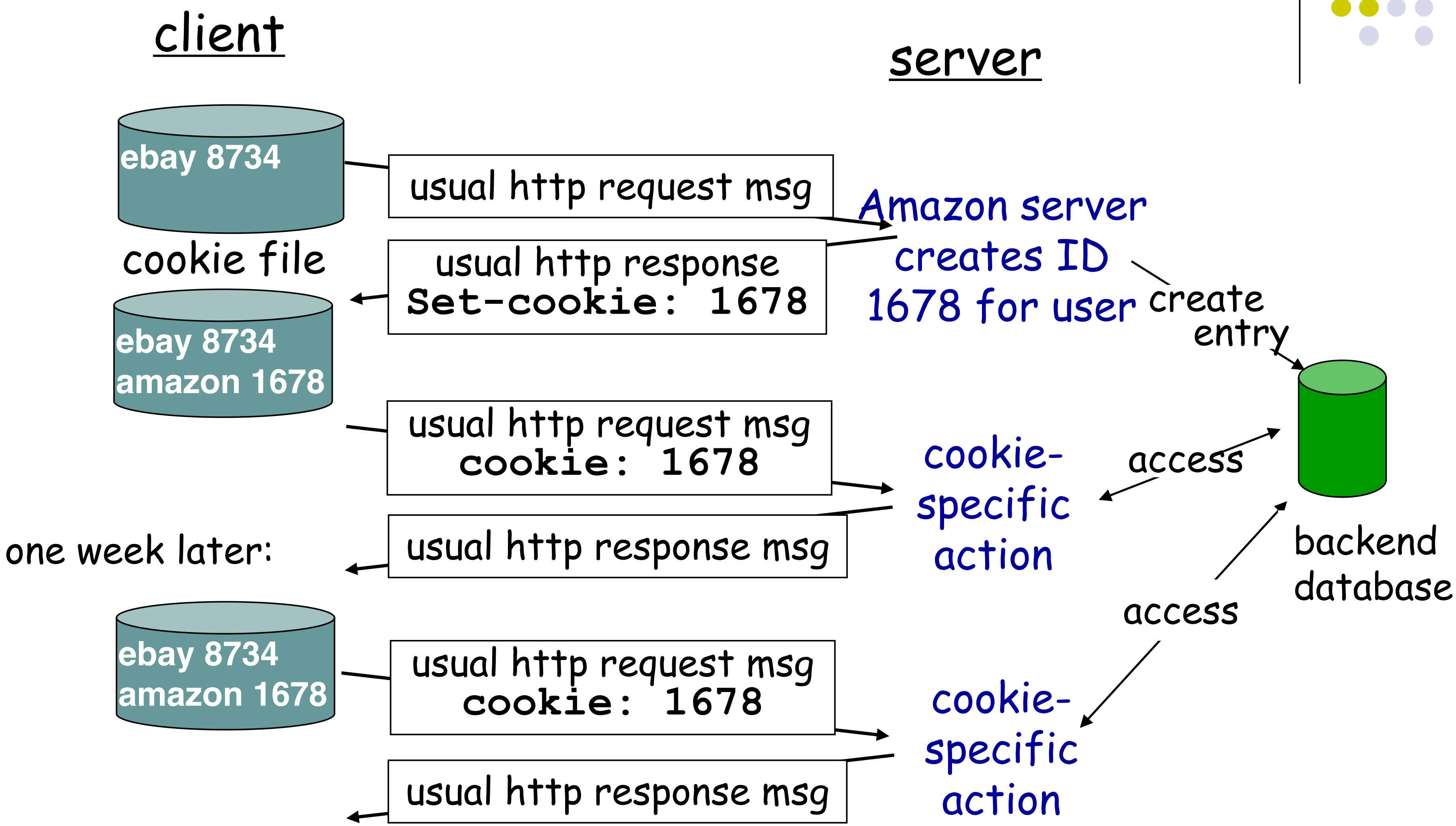
four components:

1) cookie header line of HTTP *response* message
2) cookie header line in HTTP *request* message
3) cookie file kept on user's host, managed by user's browser
4) back-end database at Web site

what cookies can bring:

- authorization
- shopping carts
- recommendations
- user session state

Cookies: keeping "state" (cont.)



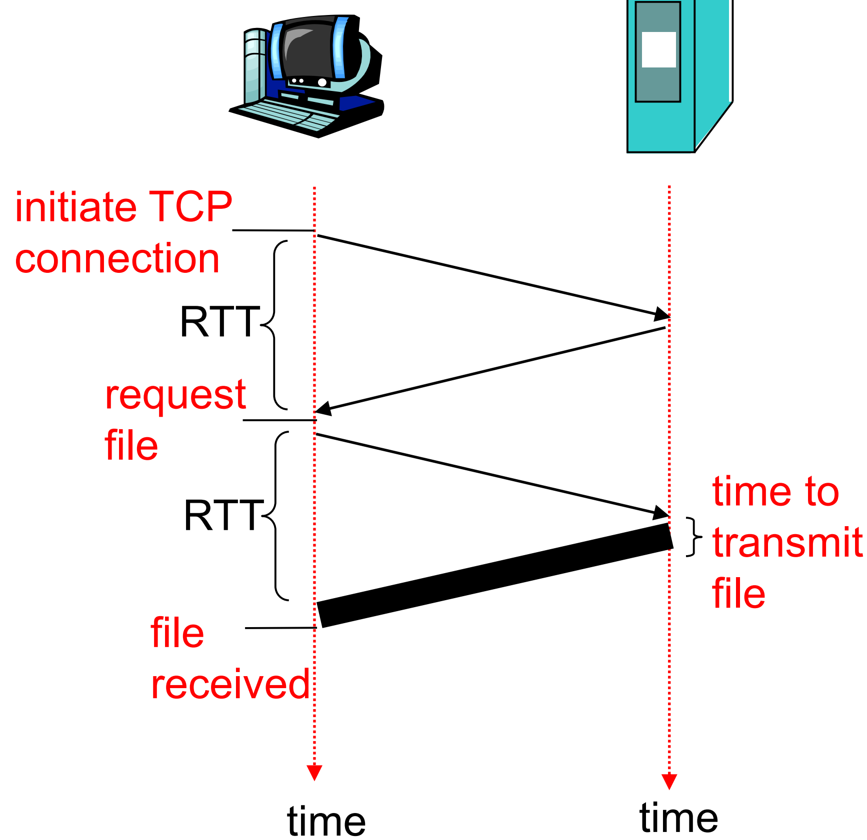
HTTP Performance: Page Load Time

Round Trip Time (RTT):

time for a small packet to travel from client to server and back.

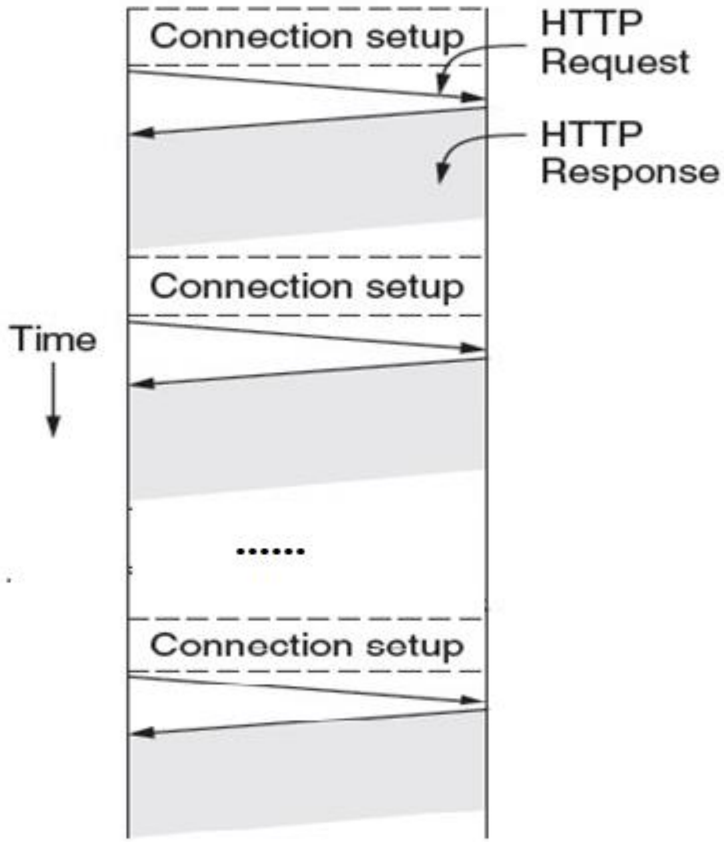
Page Load Time (PLT):

- 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

Nonpersistent HTTP:
one object sent over one TCP connection



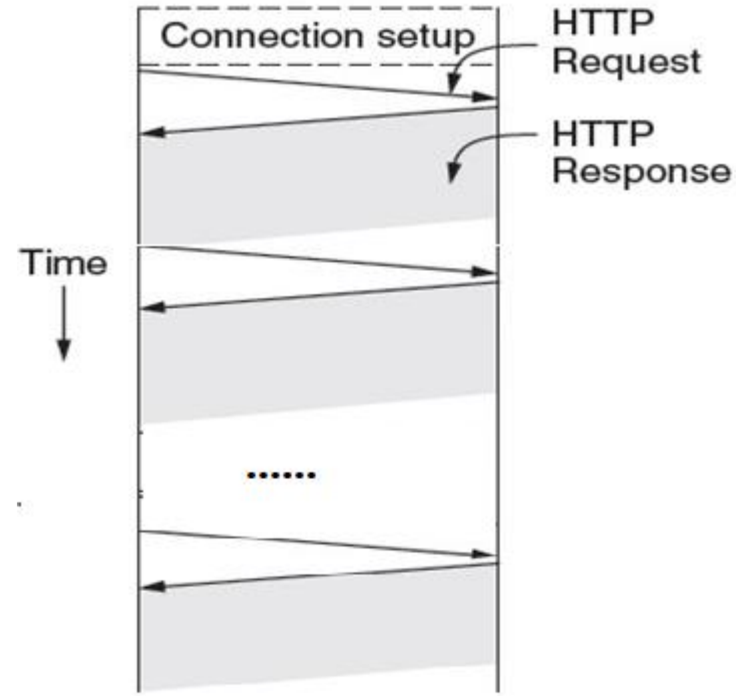
Page Load Time =

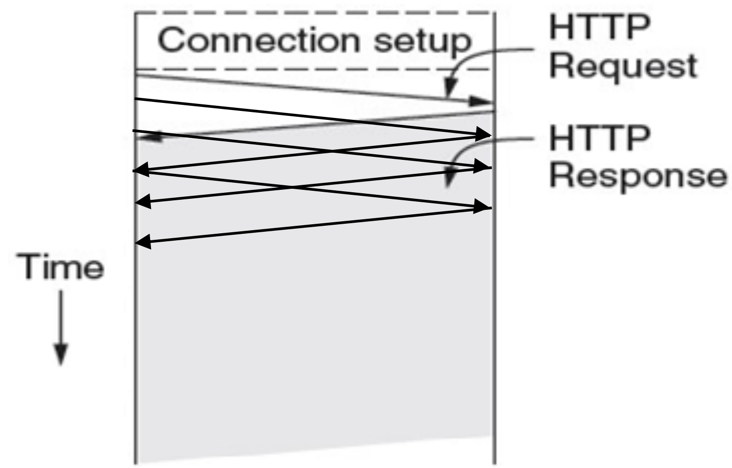
N*RTT + transmit time

N is number of objects

Ways to Decrease PLT

- Browser runs multiple (8, say) HTTP instances in parallel
- Persistent HTTP
  - 1 TCP connection to 1 server
  - use it for multiple HTTP requests (in parallel)
  - Widely used as part of HTTP/1.1
- Move content closer to client
  - Caching, and proxies
  - CDN
- SPDY(“speedy”),experimental protocol for a faster web

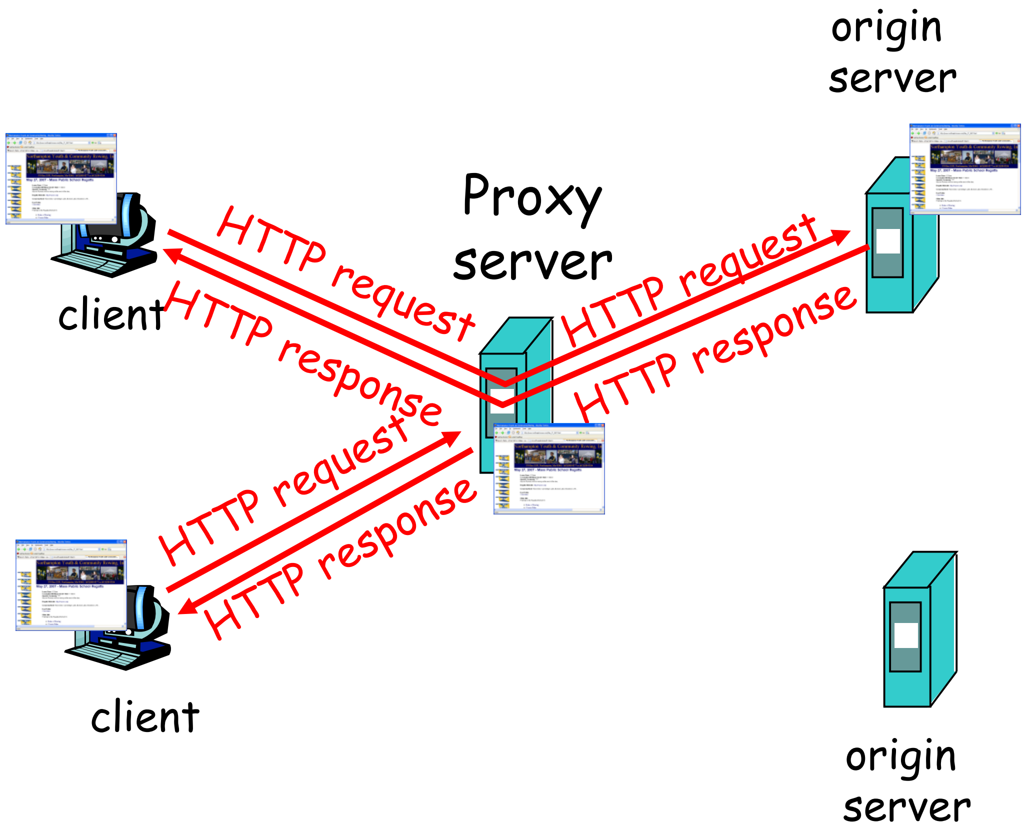




Web caches (proxy server)

Goal: satisfy client request without involving origin server

- browser sends all HTTP requests to cache
  - Object not in cache: cache requests object from origin server, then returns object to client
  - else cache returns object



More about Web caching

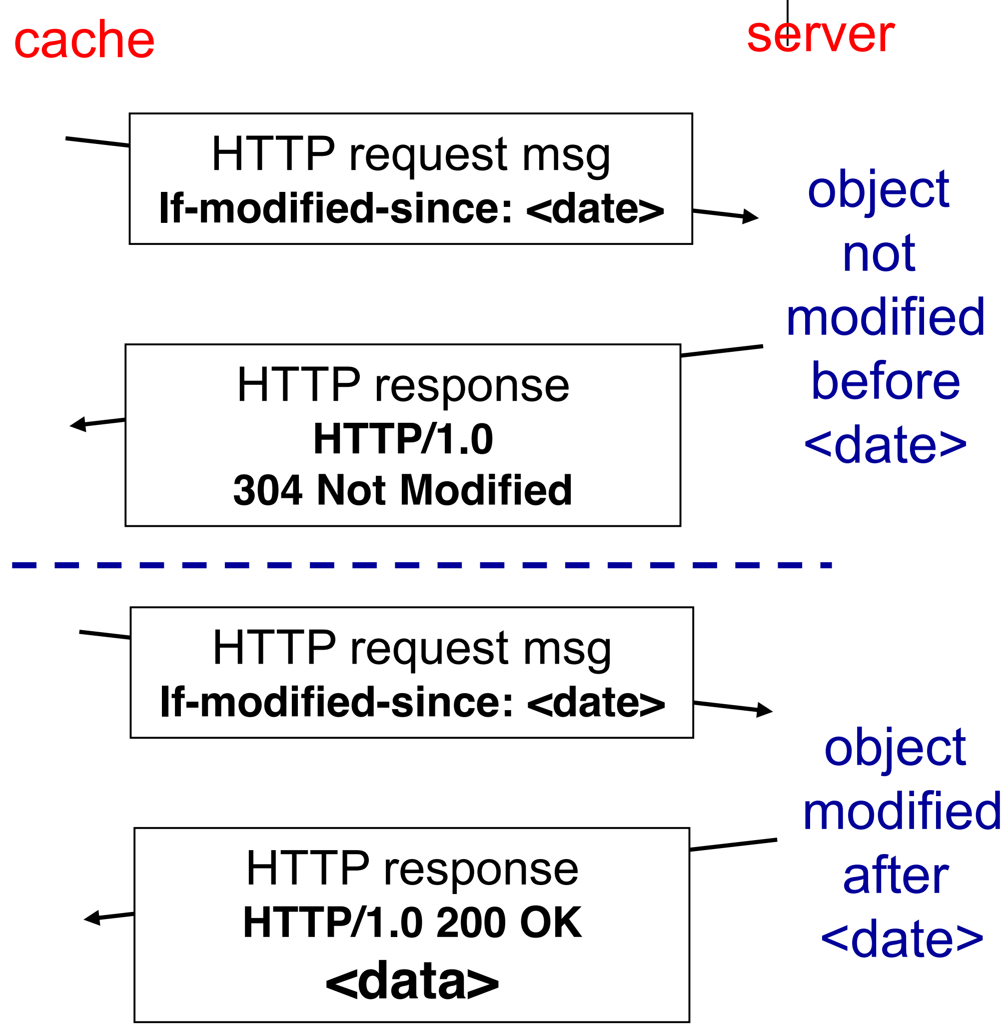
- cache acts as both client and server
- Cache can be installed anywhere: server side, proxy, client side
- Typically cache is installed by ISP (university, company, residential ISP)

Why Web caching?

- reduce response time for client request
- reduce traffic on an institution's access link.
- reduce load on origin servers

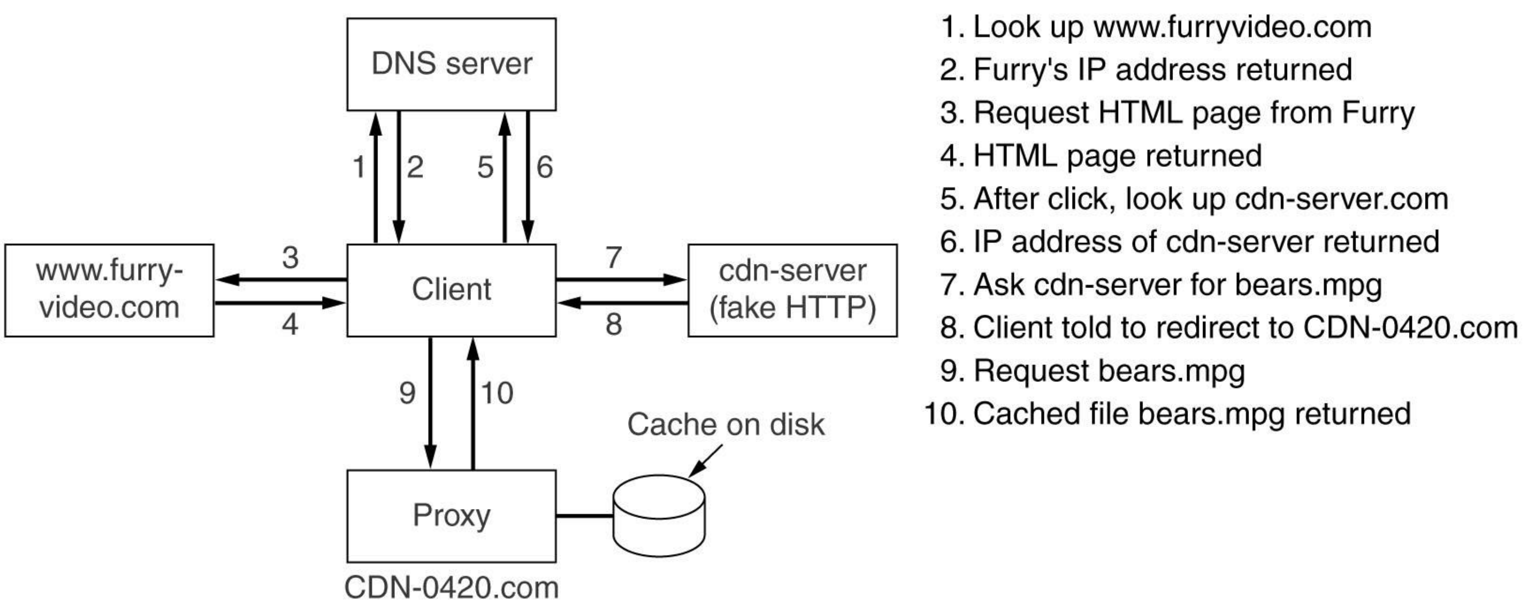
Conditional GET

- Goal: don't send object if cache has up-to-date cached version
- cache: specify date of cached copy in HTTP request
  `If-modified-since: <date>`
- server: response contains no object if cached copy is up-to-date:
  `HTTP/1.0 304 Not Modified`



Content Delivery Networks

- Replicate Web pages on a bunch of servers.
- Efficient distribution of popular content
- Faster delivery for clients



Steps in looking up *www.furry-video.com* which is a page containing references to replicated web pages (identified as *http://cdn-server.com/...*)

Assignment2

- Watch MOOC video online, reference to chapter2 of textbook and ppts, use WireShark to observe the packets about protocols of DNS, HTTP, FTP, SMTP, BitTorrent. Answer following questions for each of DNS, HTTP, FTP, SMTP, BitTorrent:
  - Is it reliable or not? connection oriented or not? use TCP or UDP?
  - What transport service does the app need?
  - Why is there a UDP?
  - What is the interaction model of the protocol: Client/Server or P2P, and how?
  - What is the message format and semantics?
- next class:
  - Discussion in groups
  - Finish the quiz each group
  - Group presentation