04832250 – Computer Networks (Honor Track)

A Data Communication and Device Networking Perspective

Module 1: Protocol Support for Network Applications

Prof. Chenren Xu（许辰人）

Center for Energy-efficient Computing and Applications

Computer Science, Peking University

chenren@pku.edu.cn

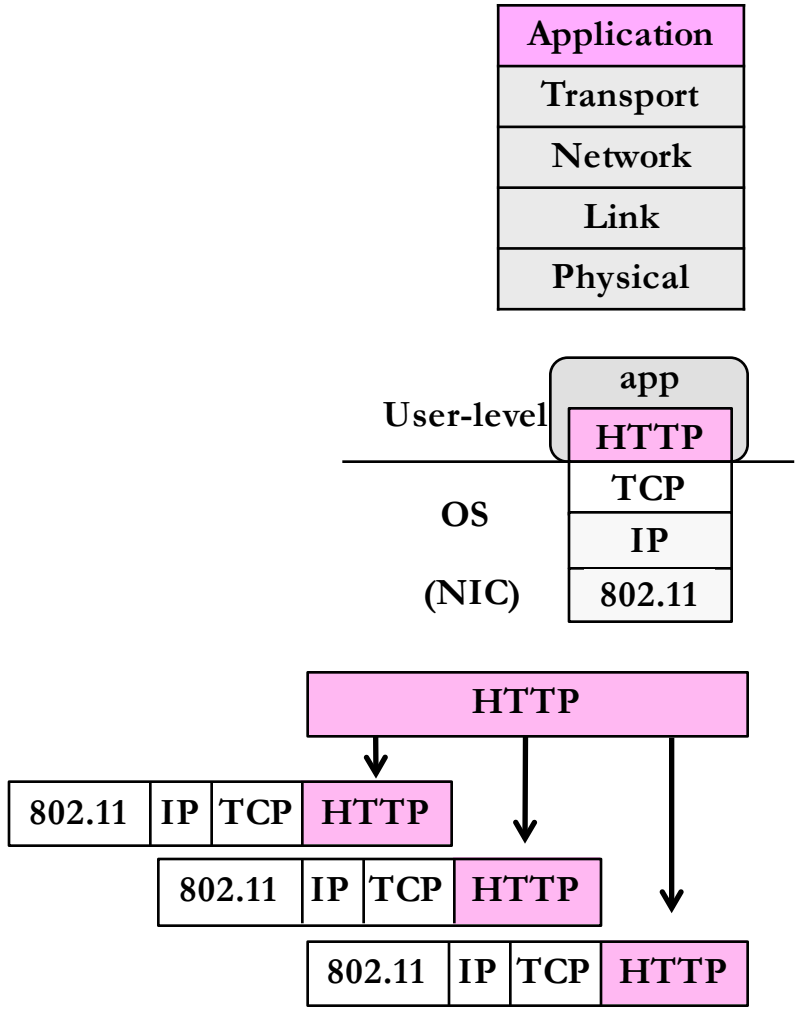
http://soar.pku.edu.cn/

Outline

- Application Layer Overview
- Domain Name System
- HTTP, the HyperText Transfer Protocol
- HTTP Performance
- HTTP Caching and Proxies
- CDNs (Content Delivery Networks)
- The Future of HTTP
- Peer-to-Peer Content Delivery (BitTorrent)

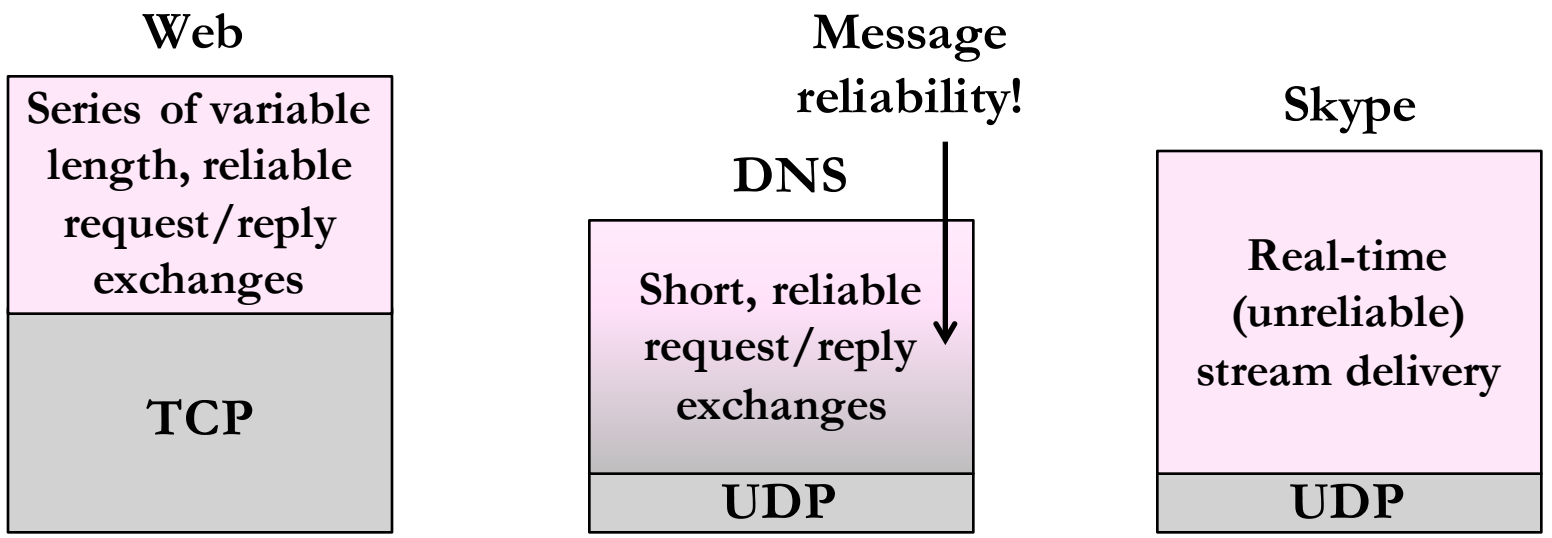
Where we are in the Course

- Starting the Application Layer!
  - Builds distributed "network services" (DNS, Web) on Transport services
- Application layer protocols are often part of an "app"
  - But don't need a GUI, e.g., DNS
- Application layer messages are often split over multiple packets
  - Or may be aggregated in a packet …



Application Communication Needs

- Vary widely with app; must build on Transport services



OSI Session/Presentation Layers

- Remember this? Two relevant concepts …

But consider part of the application, not strictly layered!

Application	– Provides functions needed by users
Presentation	– Converts different representations
Session	– Manages task dialogs
Transport	– Provides end-to-end delivery
Network	– Sends packets over multiple links
Data Link	– Sends frames of information
Physical	– Sends bits as signals

Session Concept

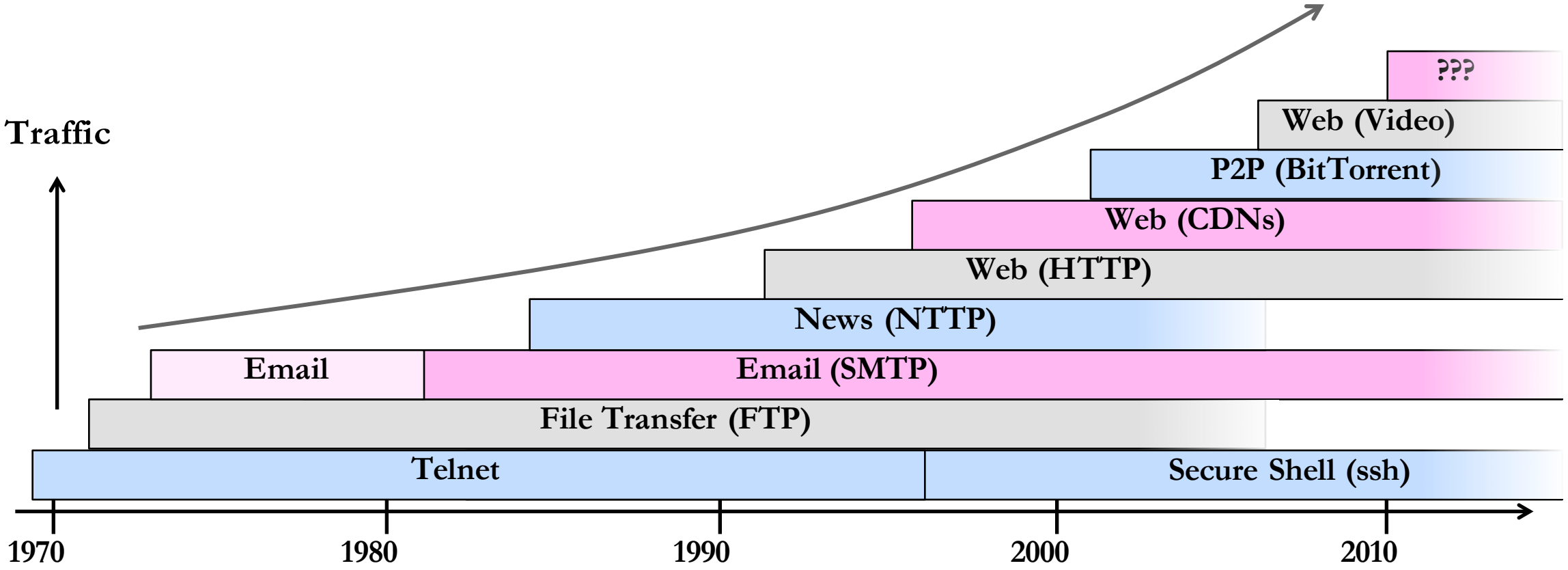
- A session is a series of related network interactions in support of an application task
  - Often informal, not explicit
- Examples:
  - Web page fetches multiple resources
  - Skype call involves audio, video, chat

Presentation Concept

- Apps need to identify the type of content, and encode it for transfer
  - These are Presentation functions
- Examples:
  - Media (MIME) types, e.g., image/jpeg, identify the type of content
  - Transfer encodings, e.g., gzip, identify the encoding of the content
  - Application headers are often simple and readable versus packed for efficiency

Evolution of Internet Applications

- Always changing, and growing …

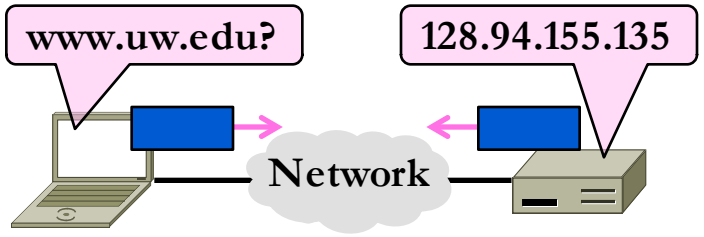


Evolution of Internet Applications cont'd

- For a peek at the state of the Internet:
  - Akamai's State of the Internet Report (quarterly)
  - Cisco's Visual Networking Index
  - Mary Meeker's Internet Report
- Robust Internet growth, esp. video, wireless and mobile
  - Most traffic is video, will be 90% of Internet in a few years
  - Wireless and mobile traffic are overtaking wired traffic
  - Growing attack traffic

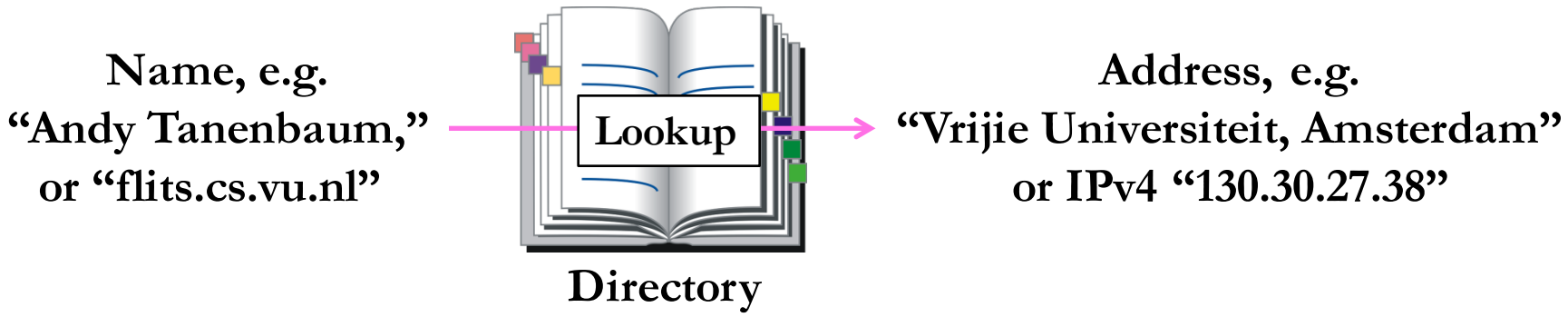
Outline

- Application Layer Overview
- Domain Name System
  - Human-readable host names, and more
  - The distributed namespace and name resolution
- HTTP, the HyperText Transfer Protocol
- HTTP Performance
- HTTP Caching and Proxies
- CDNs (Content Delivery Networks)
- The Future of HTTP
- Peer-to-Peer Content Delivery (BitTorrent)



Names and Addresses

- Names are higher-level identifiers for resources
- Addresses are lower-level locators for resources
  - Multiple levels, e.g. full name → email → IP address → Ethernet address
- Resolution (or lookup) is mapping a name to an address



Before the DNS – HOSTS.TXT

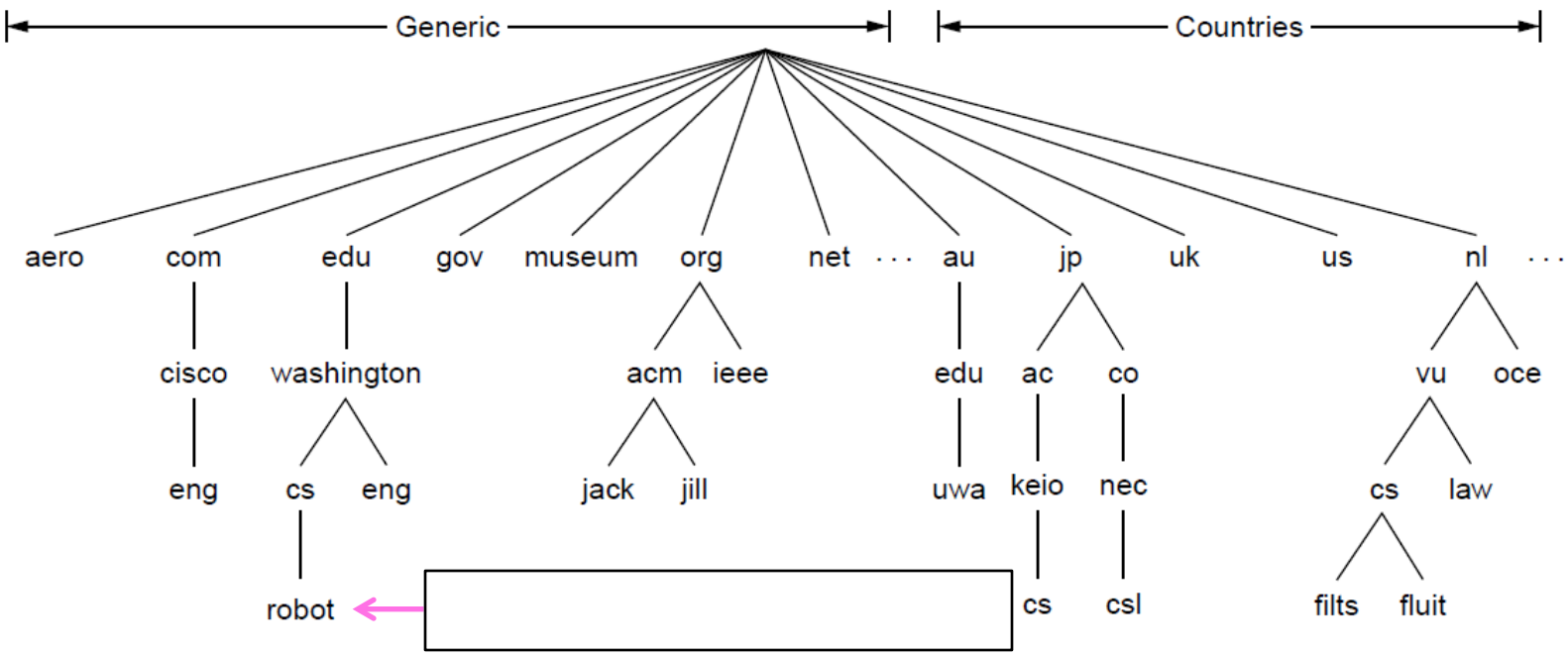
- Directory was a file HOSTS.TXT regularly retrieved for all hosts from a central machine at the NIC (Network Information Center)
- Names were initially flat, became hierarchical (e.g., lcs.mit.edu) ~85
- Neither manageable nor efficient as the ARPANET grew …

DNS

- A naming service to map between host names and their IP addresses (and more)
  - www.cmu.edu → 128.2.42.52
- Goals:
  - Easy to manage (esp. with multiple parties)
  - Efficient (good performance, few resources)
- Approach:
  - Distributed directory based on a hierarchical namespace
  - Automated protocol to tie pieces together

DNS Namespace

- Hierarchical, starting from "." (dot, typically omitted)

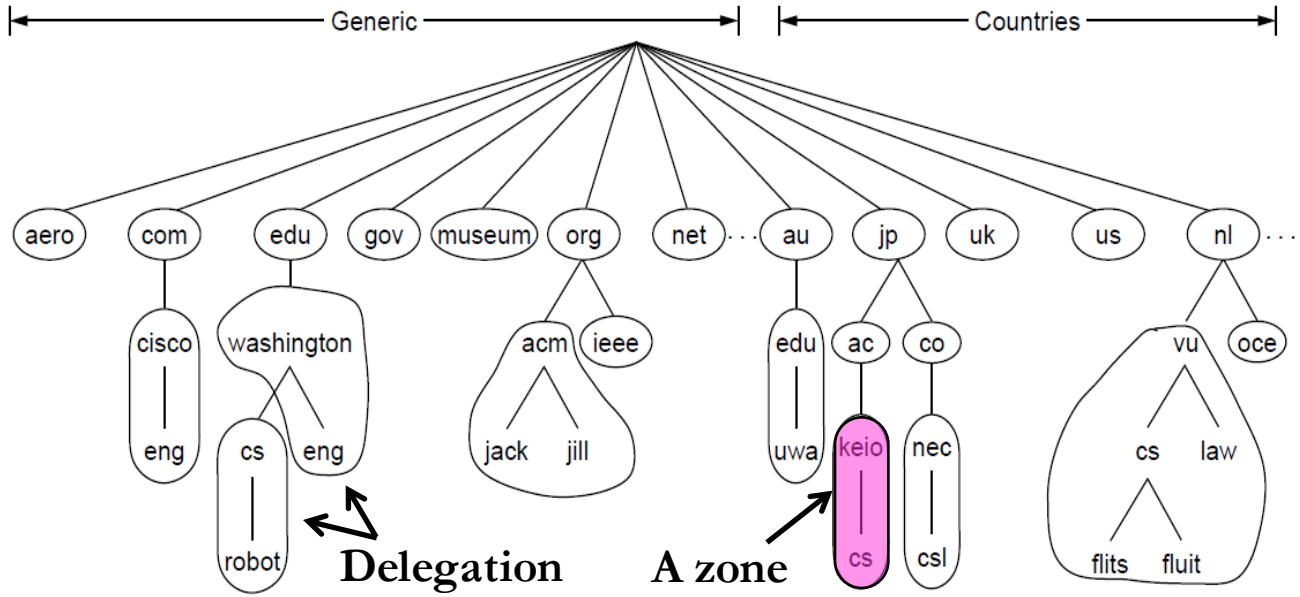


TLDs (Top-Level Domains)

- Run by ICANN (Internet Corp. for Assigned Names and Numbers)
  - Starting in '98; naming is financial, political, and international ☺
- 22+ generic TLDs
  - Initially .com, .edu , .gov., .mil, .org, .net
  - Added .aero, .museum, etc. from '01 through .xxx in '11
  - Different TLDs have different usage policies
- ~250 country code TLDs
  - Two letters, e.g., ".au", plus international characters since 2010
  - Widely commercialized, e.g., .tv (Tuvalu)
  - Many domain hacks, e.g., instagr.am (Armenia), goo.gl (Greenland)

DNS Zones

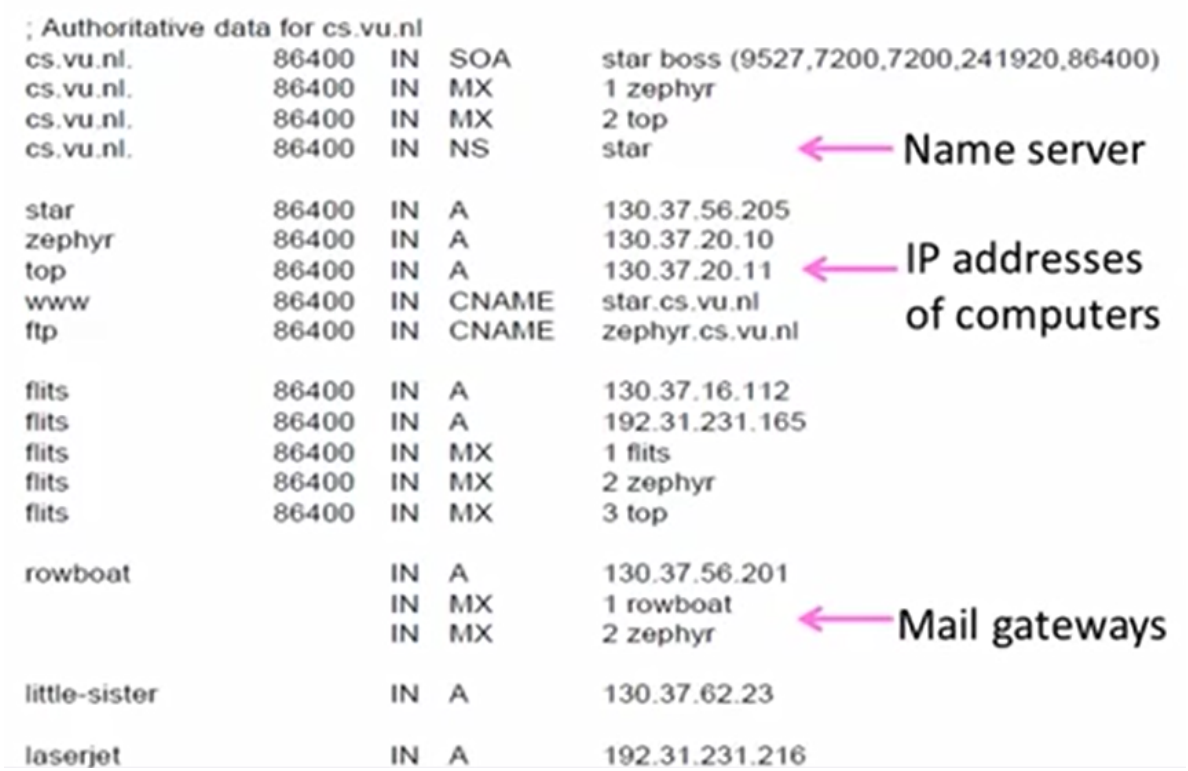
- A zone is a contiguous portion of the namespace
- Zones are the basis for distribution
  - EDU Registrar administers .edu
  - UW administers washington.edu
  - CS&E administers cs.washington.edu
- Each zone has a nameserver to contact for information about it
  - Zone must include contacts for delegations, e.g., .edu knows nameserver for washington.edu



DNS Resource Records

- A zone is comprised of DNS resource records that give information for its domain names

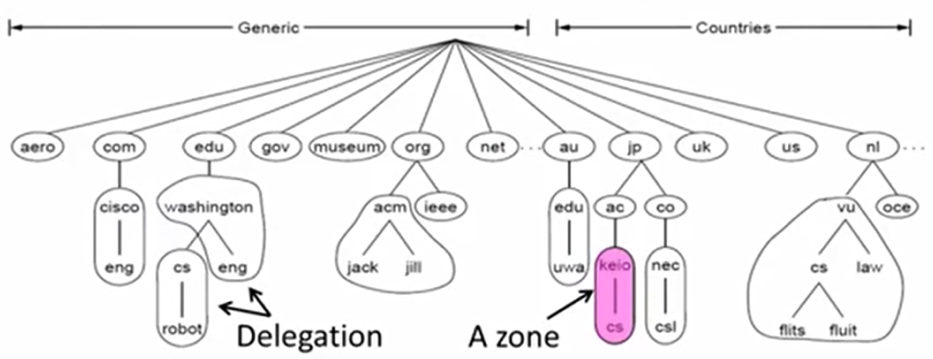
Type	Meaning
SOA	Start of authority, has key zone parameters
A	IPv4 address of a host
AAAA	IPv6 address of a host
CNAME	Canonical name for an alias
MX	Mail exchanger for the domain
NS	Nameserver of domain or delegated subdomain

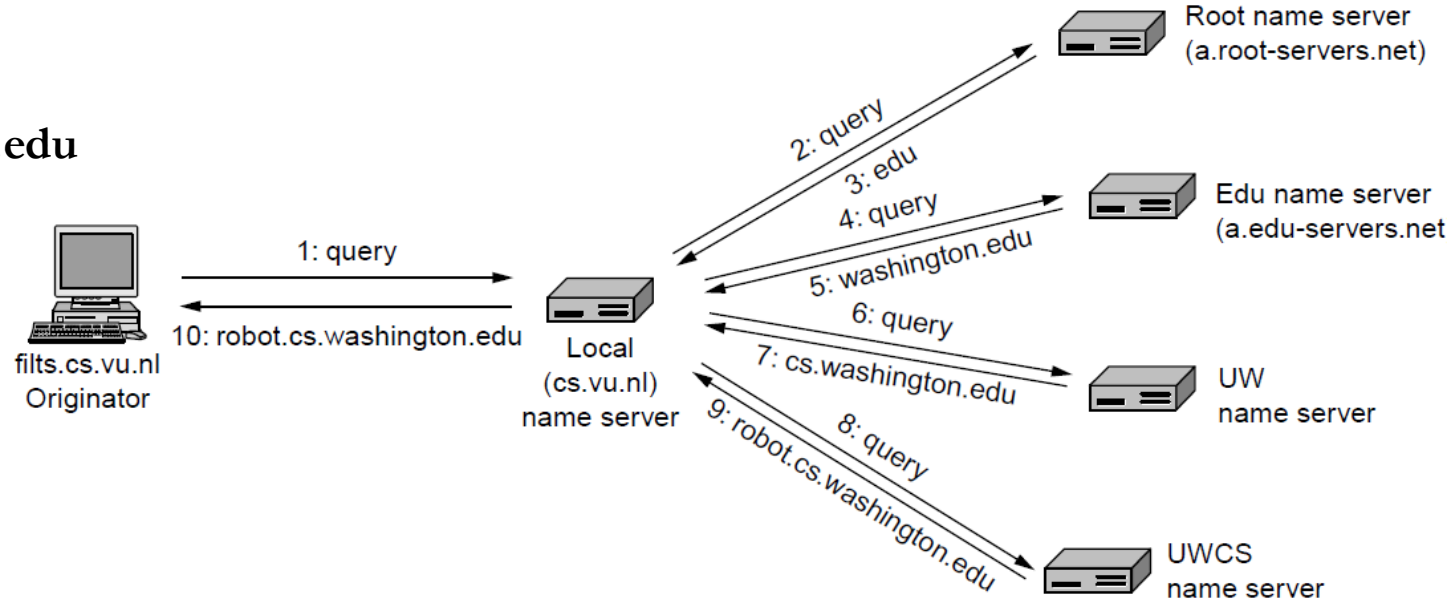


DNS Resolution

- A zone is a contiguous portion of the namespace
  - Each zone is managed by one or more nameservers
- DNS protocol lets a host resolve any host name (domain) to IP address
- If unknown, can start with the root nameserver and work down zones
- Let's see an example first …

flits.cs.vu.nl resolves robot.cs.washington.edu





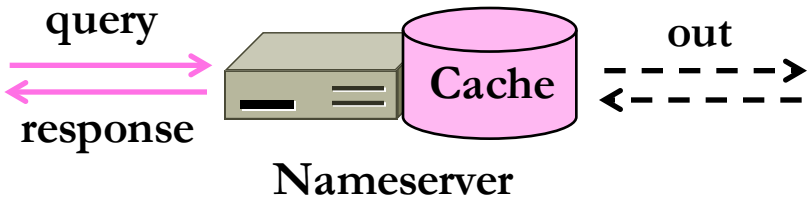
Iterative vs. Recursive Queries

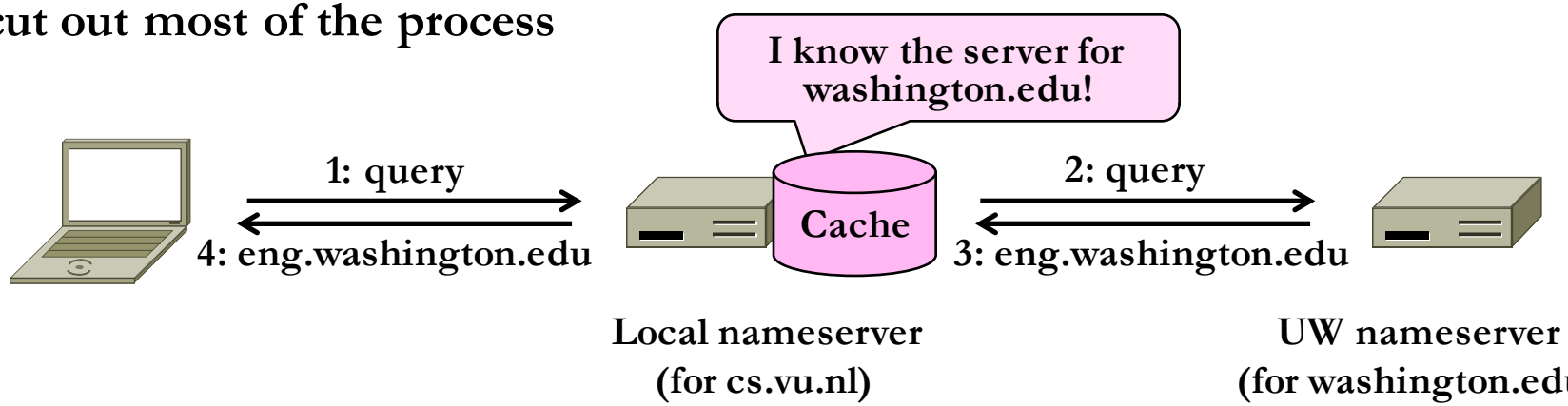
- Recursive query
  - Nameserver completes resolution and returns the final answer
  - E.g., flits local → nameserver
- Iterative query
  - Nameserver returns the answer or who to contact next for the answer
  - E.g., local nameserver → all others

- Recursive query
  - Lets server offload client burden (simple resolver) for manageability
  - Lets server cache over a pool of clients for better performance
- Iterative query
  - Lets server "file and forget"
  - Easy to build high load servers

Caching

- Resolution latency should be low
  - Adds delay to web browsing
- Cache query/responses to answer future queries immediately
  - Including partial (iterative) answers
  - Responses carry a TTL for caching
- flits.cs.vu.nl now resolves eng.washington.edu
  - And previous resolutions cut out most of the process





Local Nameservers

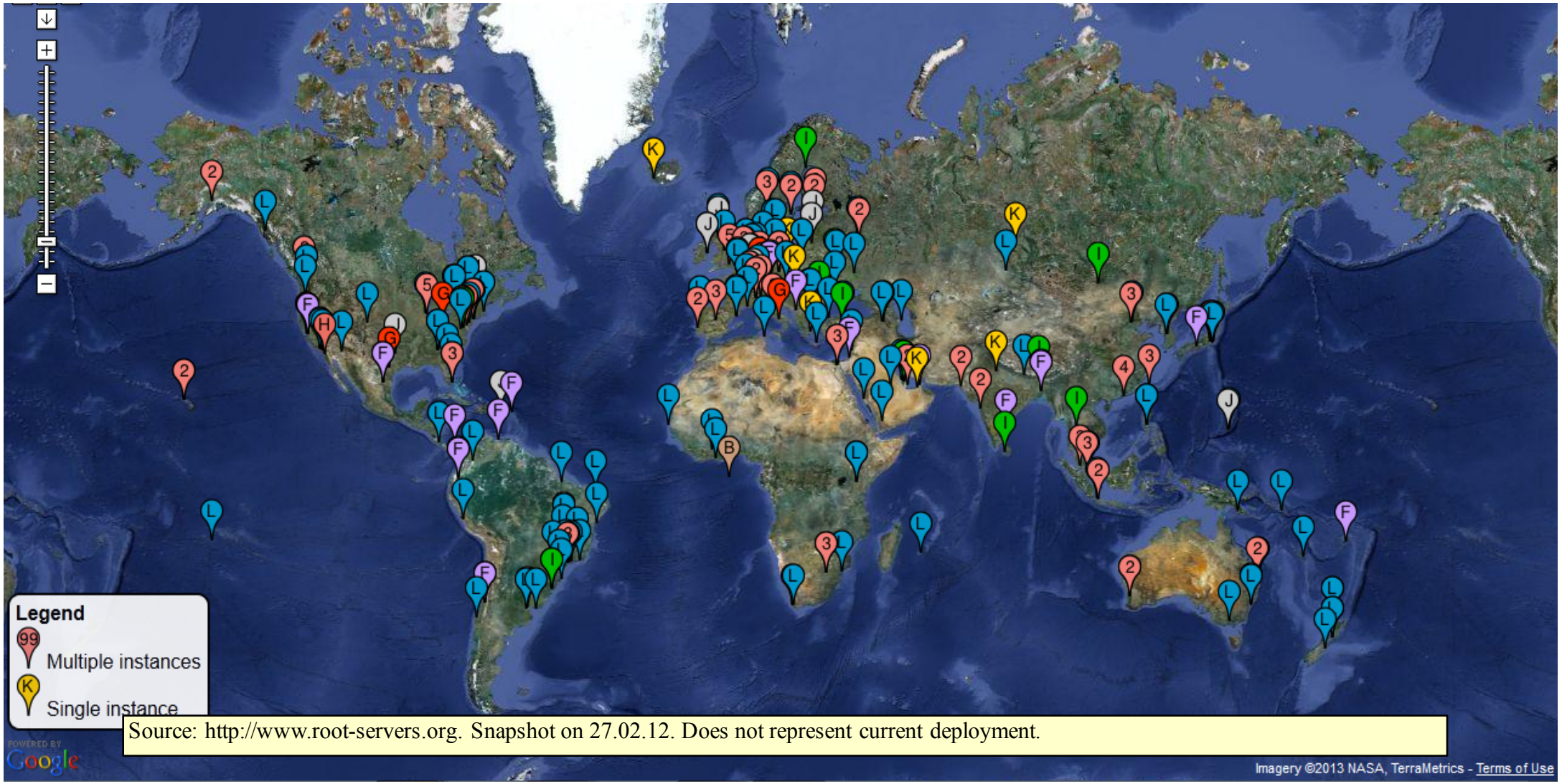
- Local nameservers typically run by IT (enterprise, ISP)
  - But may be your host or wireless router
  - Or alternatives e.g., Google public DNS (8.8.8.8, …)
- Clients need to be able to contact their local nameservers
  - Typically configured via DHCP (can piggyback more …)

Root Nameservers

- Root (dot) is served by 13 server names
  - a.root-servers.net to m.root-servers.net
  - All nameservers need root IP addresses
  - Handled via configuration file (named.ca)
- There are >250 distributed server instances
  - Highly reachable, reliable service
  - Most servers are reached by IP anycast (Multiple locations advertise same IP! Routes take client to the closest one.)
  - Servers are IPv4 and IPv6 reachable

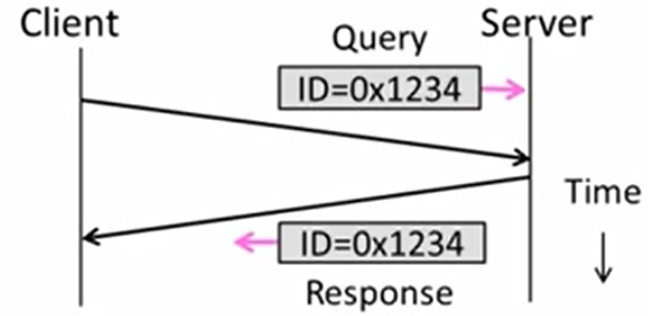
How to destroy the Internet?

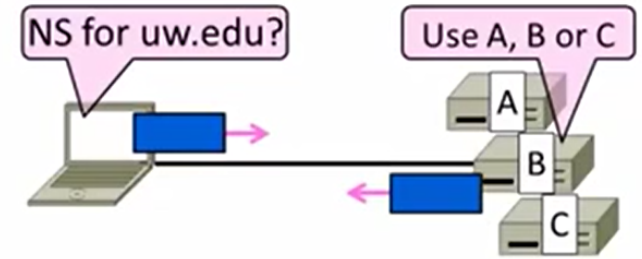
Root Server Deployment

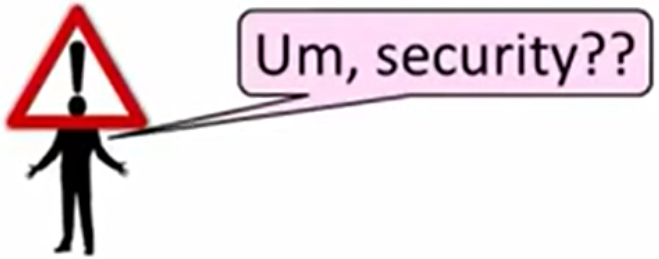


DNS Protocol

- 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
  - Return the list; clients use one answer
  - Helps distribute load too
- Security is a major issue
  - Compromise redirects to wrong site!
- DNSSEC (DNS Security Extensions)
  - Long under development, now partially deployed. We'll look at it later







Outline

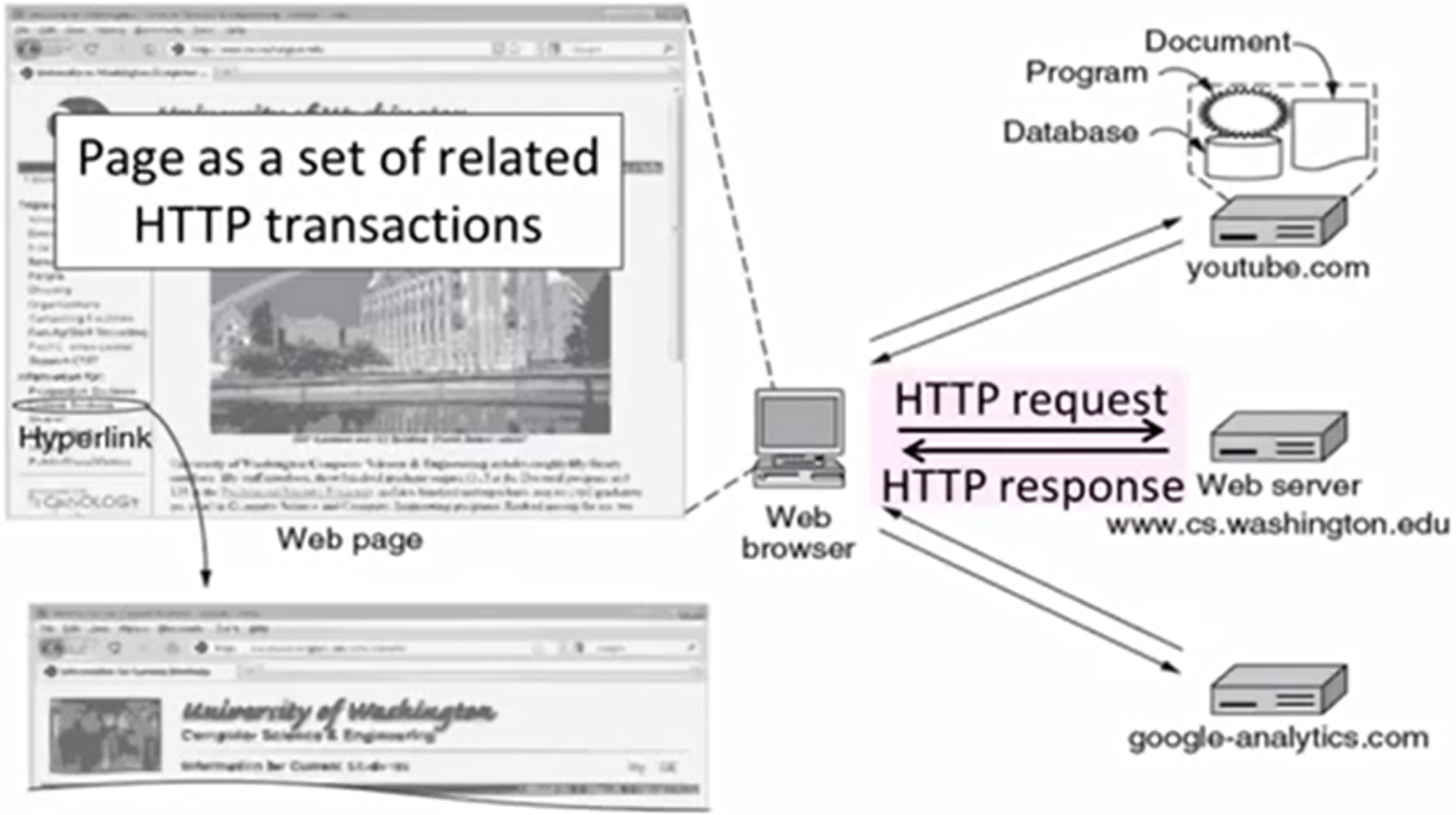
- Application Layer Overview
- Domain Name System
- HTTP, the HyperText Transfer Protocol
  - Basis for fetching Web pages
- HTTP Performance
- HTTP Caching and Proxies
- CDNs (content Delivery Networks)
- The Future of HTTP
- Peer-to-Peer Content Delivery (BitTorrent)

Sir Tim Berners-Lee (1955-)

- Inventor of the Web
  - Dominant Internet app since mid 90s
  - He now directs the W3C
- Developed Web at CERN in '89
  - Browser, server and first HTTP
  - Popularized via Mosaic ('93), Netscape
  - First WWW conference in '94 …

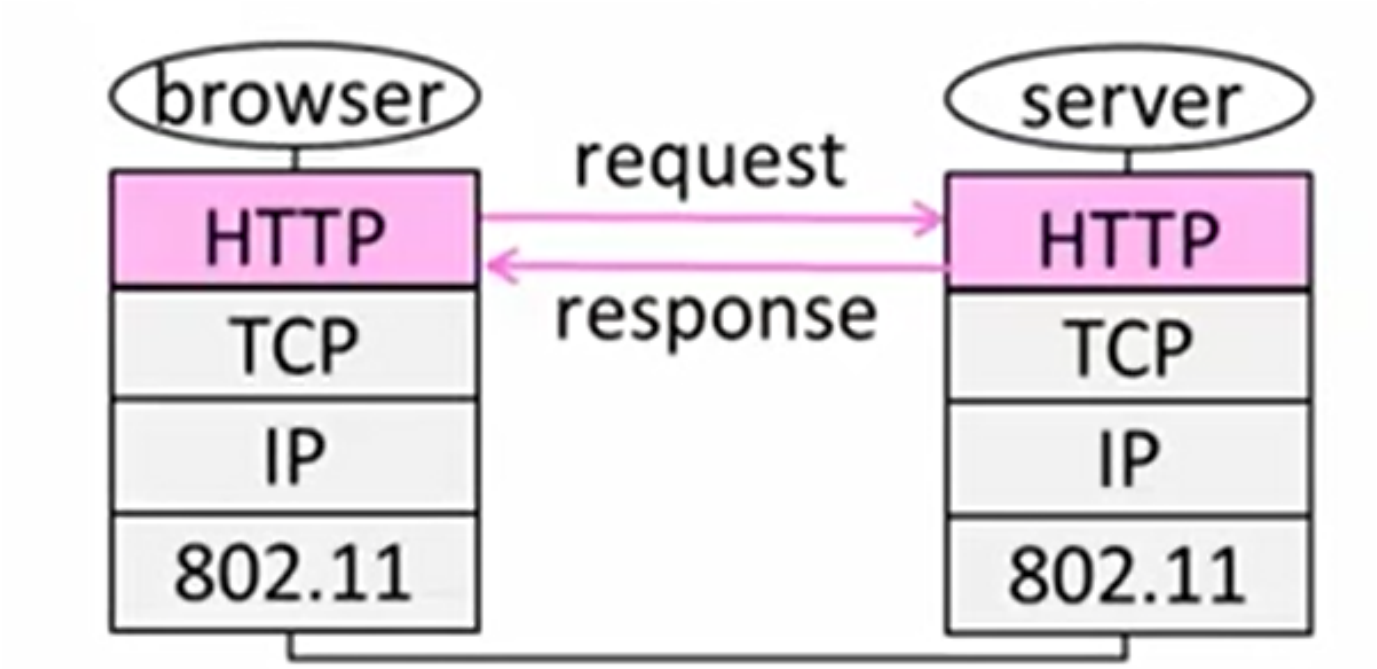
Source: By Paul Clarke, CC-BY-2.0, via Wikimedia Commons

Web Context


Page as a set of related HTTP transactions
Hyperlink
Web page
Document
Program
Database
youtube.com
HTTP request
HTTP response
Web server
www.cs.washington.edu
Web browser
google-analytics.com

HTTP Context

- HTTP is a request/response protocol for fetching Web resources
  - Runs on TCP, typically port 80
  - Part of browser/server app



Fetching a Web page with HTTP

- Start with the page URL:

http://en.wikioedia.org/wiki/Vegemite

Protocol Server Page on server

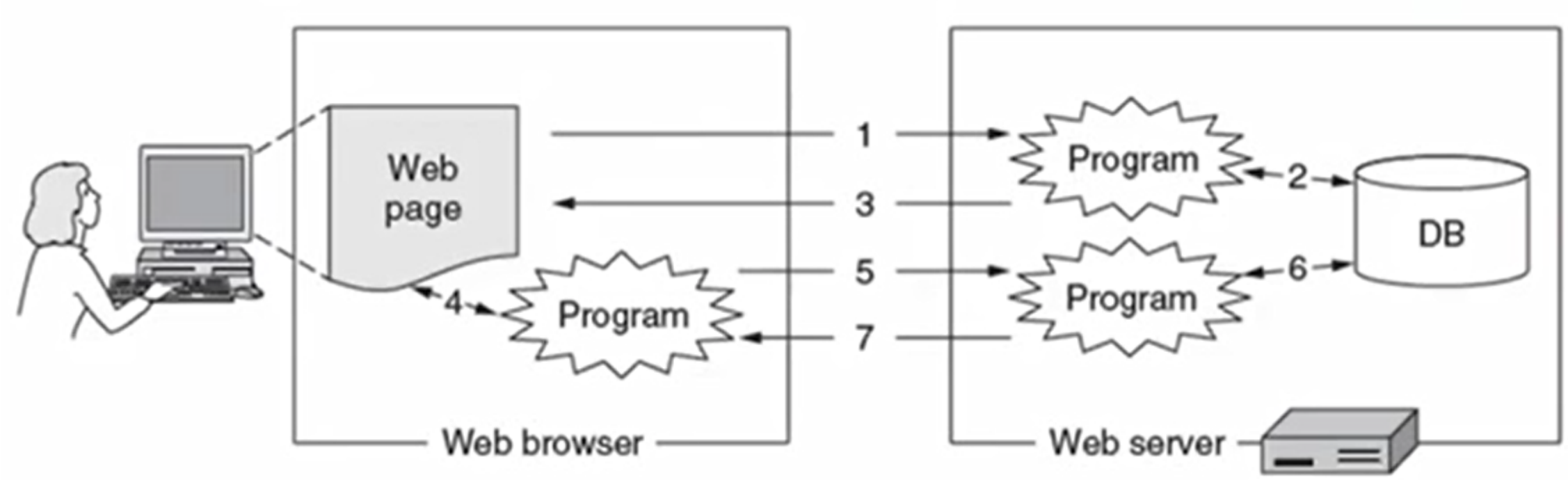
- Steps:
  - Resolve the server to IP address (DNS)
  - Set up TCP connection to the server
  - Send HTTP request for the page
  - (Await HTTP response for the page)

** Execute/fetch embedded resources/render

  - Clean up any idle TCP connections

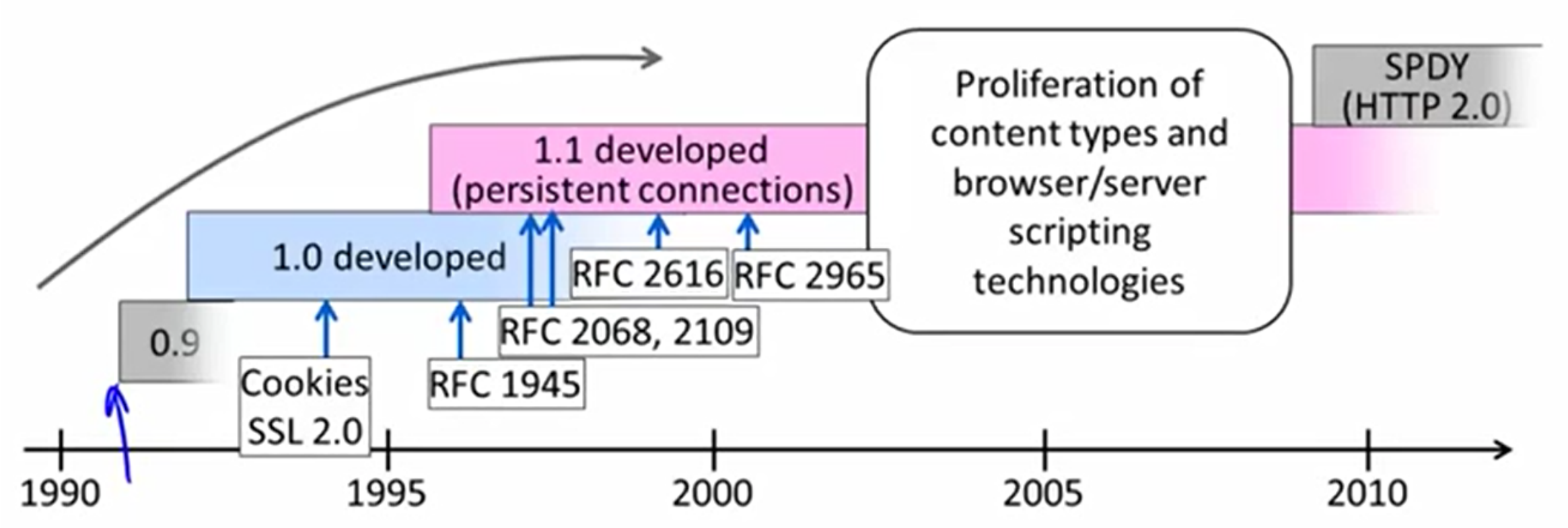
Static vs Dynamic Web pages

- Static web page is a file contents, e.g., image
- Dynamic web page is the result of program execution
  - Javascript on client, PHP on server, or both



Evolution of HTTP

- Consider security (SSL/TLS for HTTPS) later



HTTP Protocol

- Originally a simple protocol, with many options added over time
  - Text-based commands, headers
- Try it yourself
  - As a "browser" fetching a URL
  - Run "telnet en.wikipedia.org 80"
  - Type "GET /wiki/Vegemite HTTP/1.0" to server followed by a blank line
  - Server will return HTTP response with the page contents (or other info)

Commands used in the request

	Method	Description
Fetch page →	GET	Read a Web page
	HEAD	Read a Web page's header
Upload data →	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

Code returned with response

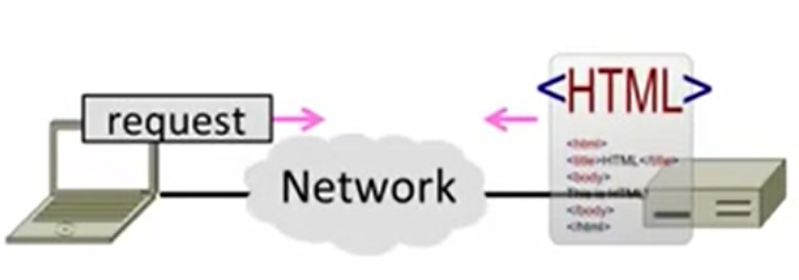
Code	Meaning	Examples
1xx	Information	100 = server agrees to handle client's request
Yes! → 2xx	Success	200 = request succeeded; 204 = no content present
3xx	Redirection	301 = page moved; 304 = cached page still valid
4xx	Client error	403 = forbidden page; 404 = page not found
5xx	Server error	500 =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

Many header fields specify capabilities and content

Function	Example Headers
Browser capabilities (client → server)	User-Agent, Accept, Accept-Charset, Accept-Encoding, Accept-Language
Caching related (mixed directions)	If-Modified-Since, If-None-Match, Date, Last-Modified, Expires, Cache-Control, ETag
Browser context (client → server)	Cookie, Referer, Authorization, Host
Content delivery (server → client)	Content-Encoding, Content-Length, Content-Type, Content-Language, Content-Range, Set-Cookie

Outline

- Application Layer Overview
- Domain Name System
- HTTP, the HyperText Transfer Protocol
- HTTP Performance
  - Parallel and persistent connections
- HTTP Caching and Proxies
- CDNs (content Delivery Networks)
- The Future of HTTP
- Peer-to-Peer Content Delivery (BitTorrent)



PLT (Page Load Time)

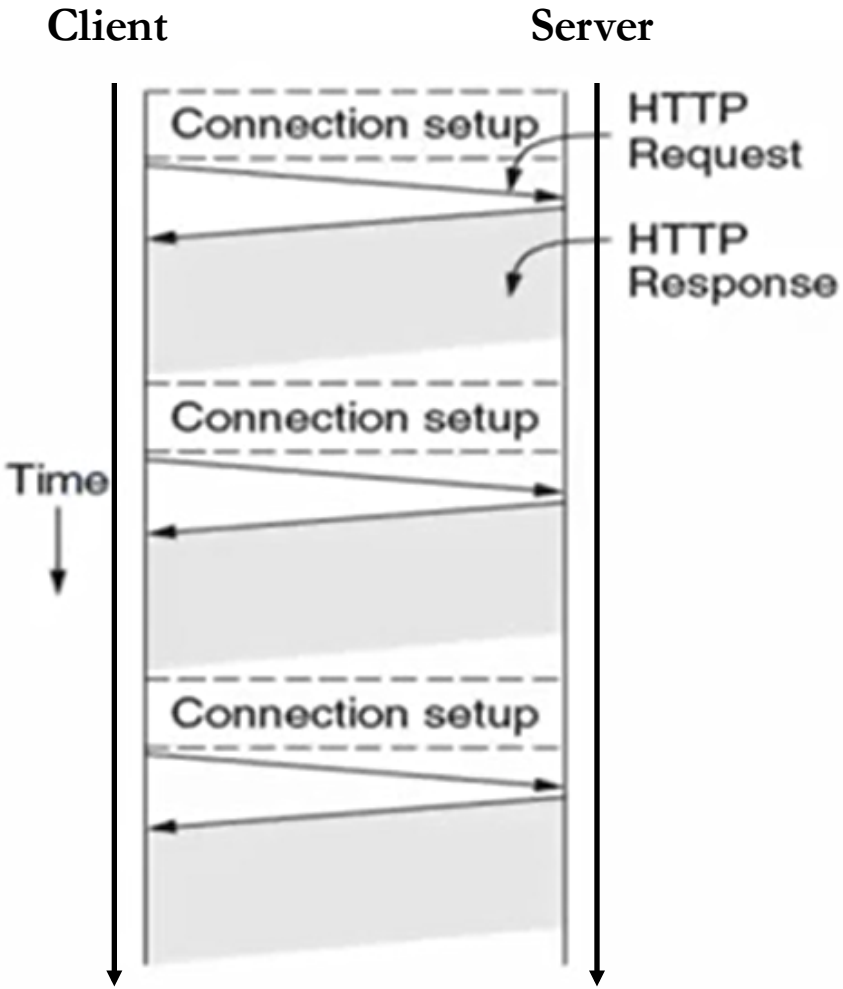
- PLT is the key measure of web performance
  - From click until user sees page
  - Small increases in PLT decrease sales
    - https://www.nngroup.com/articles/response-times-3-important-limits/
- PLT depends on many factors
  - Structure of page/content
  - HTTP (and TCP!) protocol
  - Network RTT and bandwidth

Early Performance

- HTTP/1.0 uses one TCP connection to fetch one web resource
  - Made HTTP very easy to build
  - But gave fairly poor PLT …

TCP Connection overhead:
3-way handshake +
4-way termination

- Many reasons why PLT is larger than necessary
  - Sequential request/responses, even when to different servers
  - Multiple TCP connection setups to the same server
  - Multiple TCP slow-start phases – congestion control purpose
- Network is not used effectively
  - Worse with many small resources / page



Ways to Decrease PLT

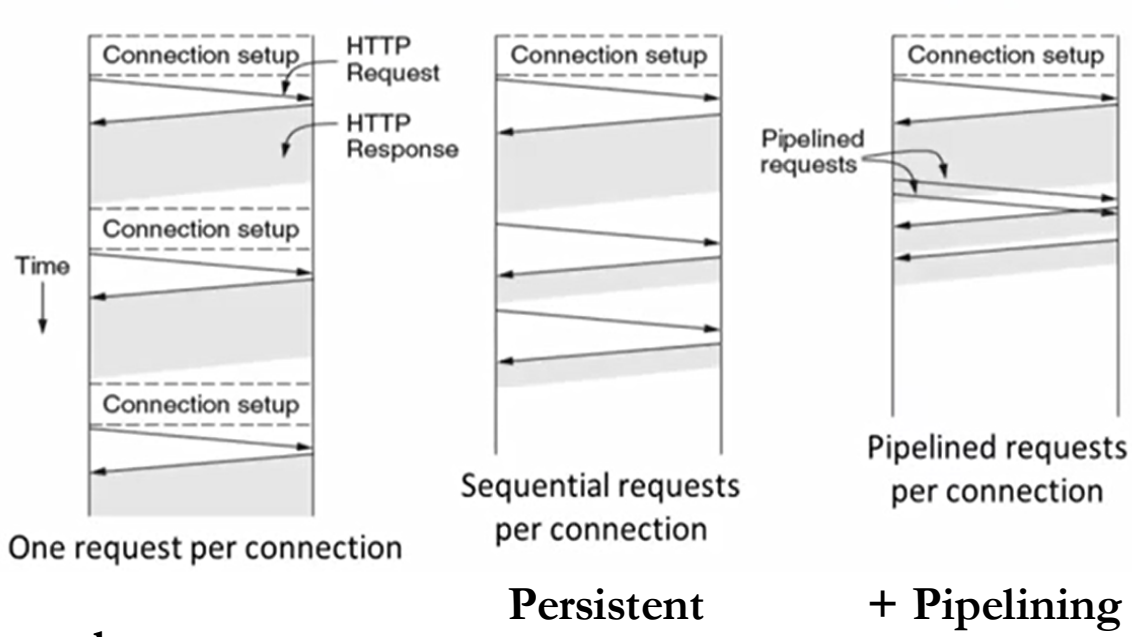
- Reduce content size for transfer
  - Smaller images, gzip
- Change HTTP to make better use of available bandwidth
- Change HTTP to avoid repeated transfers of the same content
  - Caching, and proxies
- Move content closer to client
  - CDNs

Parallel Connections

- One simple way to reduce PLT
  - Browser runs multiple (8, say) HTTP instances in parallel
  - Server is unchanged; already handled concurrent requests for many clients
- How does this help?
  - Single HTTP wasn't using network much …
  - So parallel connections aren't slowed much
  - Pulls in completion time of last fetch

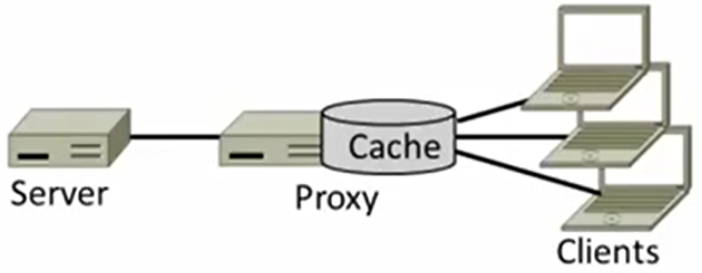
Persistent Connections

- Parallel connections compete with each other for network resources
  - 1 parallel client ≈ 8 sequential clients?
  - Exacerbates network bursts, and loss
- Persistent connection alternative
  - Make 1 TCP connection to 1 server
  - Use it for multiple HTTP requests
- Widely used as part of HTTP/1.1
  - Supports optional pipelining
  - PLT benefits depending on page structure, but easy on network
- Issues with persistent connections
  - How long to keep TCP connection?
  - Can it be slower? (Yes. But why?)



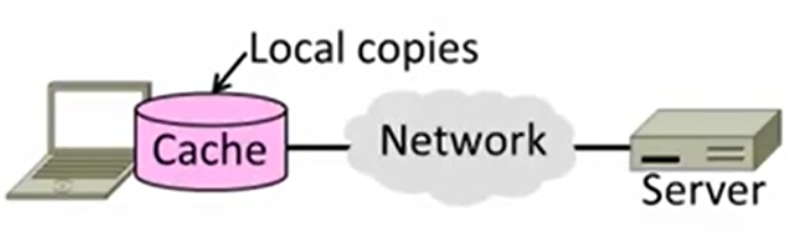
Outline

- Application Layer Overview
- Domain Name System
- HTTP, the HyperText Transfer Protocol
- HTTP Performance
- HTTP Caching and Proxies
  - Enabling content reuse
- CDNs (content Delivery Networks)
- The Future of HTTP
- Peer-to-Peer Content Delivery (BitTorrent)

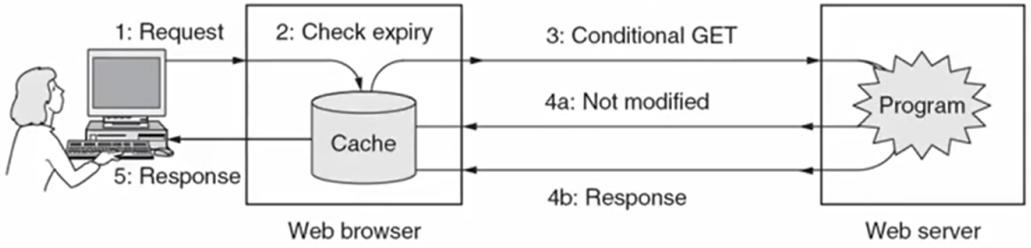


Web Caching

- Users often revisit web pages
  - Big win from reusing local copy/caching!
- Locally determine copy is still valid
  - Based on expiry information such as "Expires" header form server
  - Or use a heuristic to guess (cacheable, freshly valid, not modified recently)
  - Benefits: Content is then available right away
- Revalidate copy with remote server
  - Based on timestamp of copy such as "Last-Modified" header from server
  - Or based on content of copy such as "Etag" header from server
  - Content is available after 1 RTT

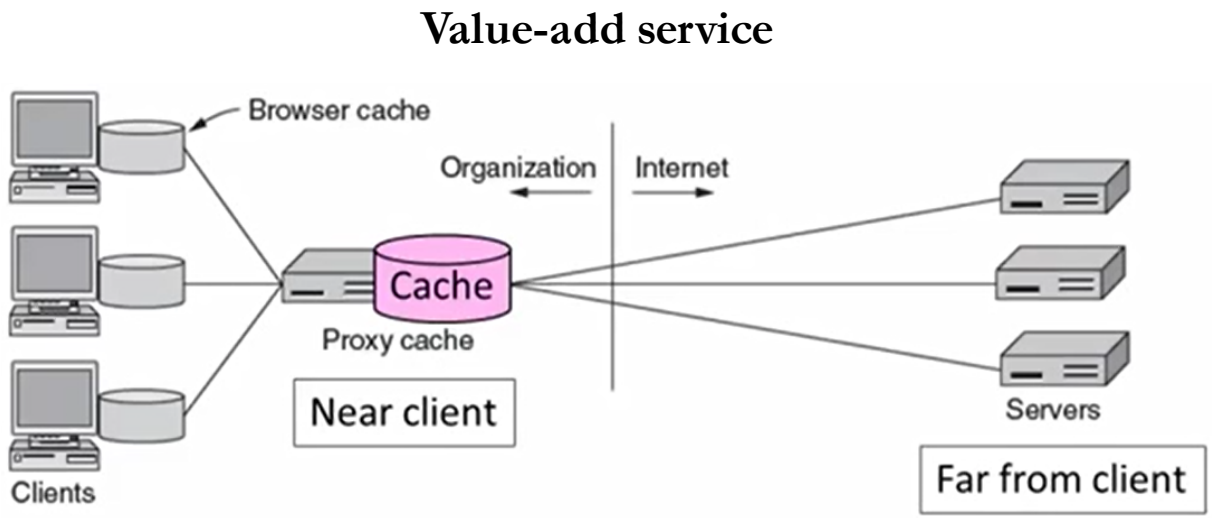


When is it OK to reuse local copy?



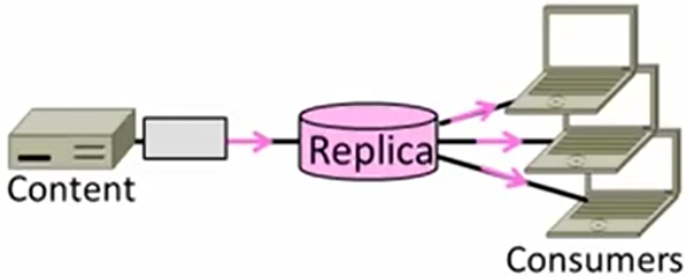
Web Proxies

- Place intermediary between pool of clients and external web servers
  - Benefits for clients include greater caching and security checking
  - Organizational access policies too!
- Proxy caching
  - Clients benefit from larger, shared cache
  - Benefits limited by secure / dynamic content, as well as "long tail"
- Clients contact proxy; proxy contacts server



Outline

- Application Layer Overview
- Domain Name System
- HTTP, the HyperText Transfer Protocol
- HTTP Performance
- HTTP Caching and Proxies
- CDNs (Content Delivery Networks)
  - Efficient distribution of popular content; faster delivery for clients
- The Future of HTTP
- Peer-to-Peer Content Delivery (BitTorrent)

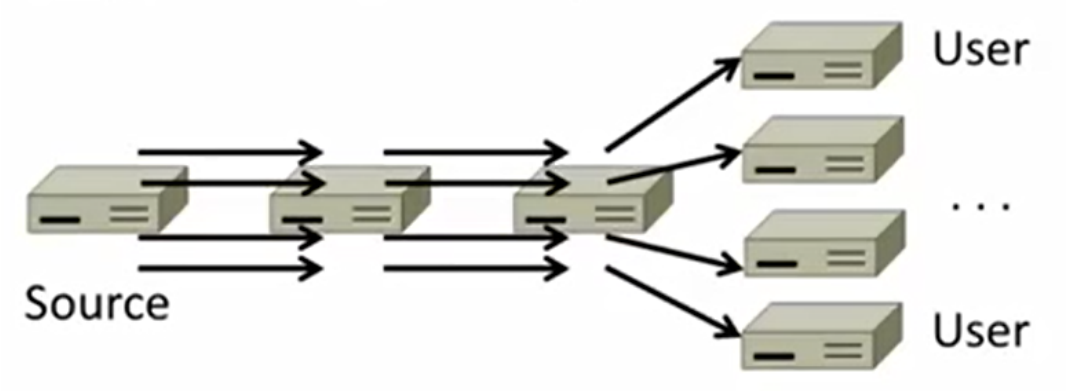


Context

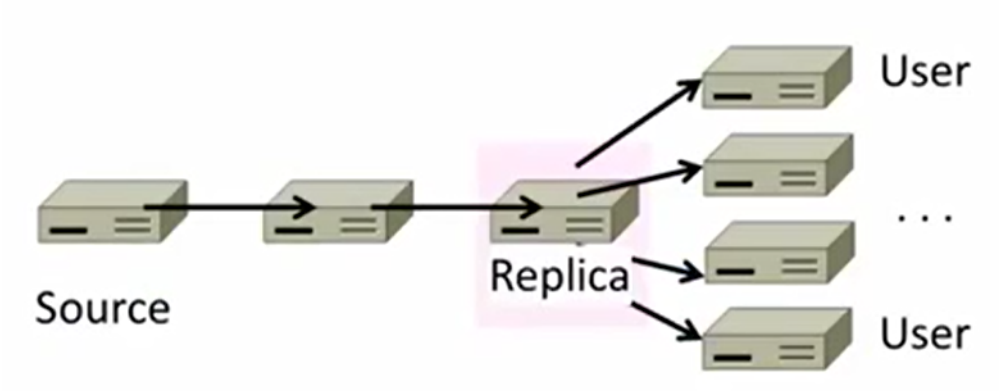
- As the web took off in the 90s, traffic volumes grew and grew. This:
  - Concentrated load on popular servers
  - Led to congested networks and need to provision more bandwidth
  - Gave a poor user experience
- Idea:
  - Place popular content near clients
  - Helps with all three issues above

Benefits of CDNs

- Before CDNs
  - Sending content from the source to 4 users takes 4 X 3 = 12 "network hops" in the example

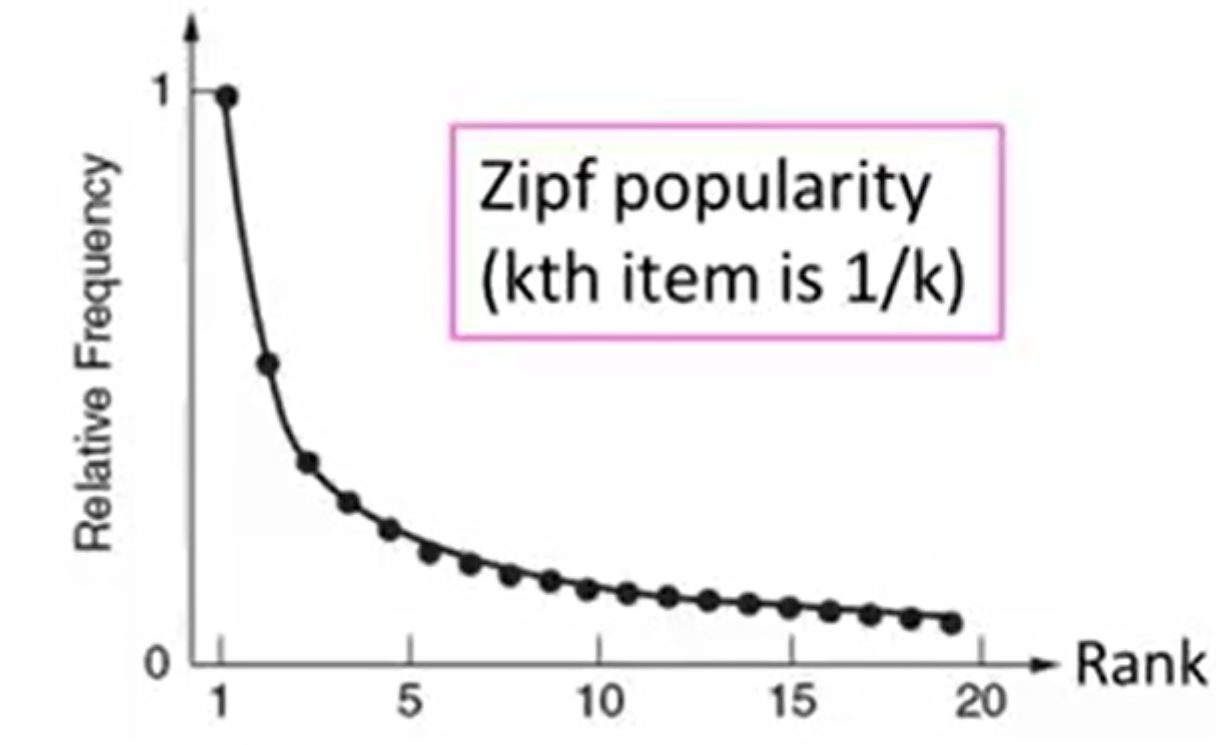


- After CDNs
  - Sending content via replicas takes only 4 + 2 = 6 "network hops"
  - Benefits assuming popular content:
    - Reduces server, network load
    - Improves user experience (PLT)



Popularity of Content

- **Zipf's Law: few popular items, many unpopular ones; both matter**

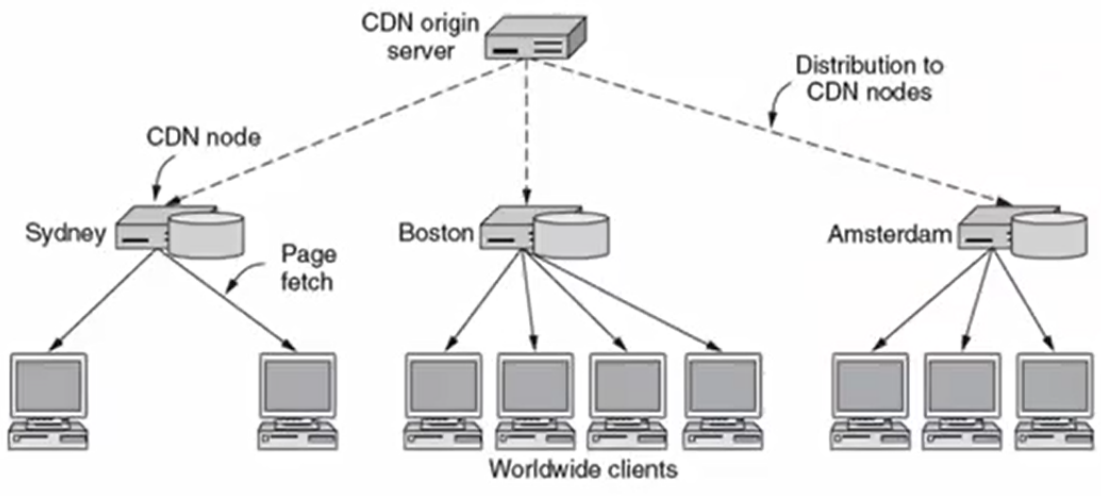


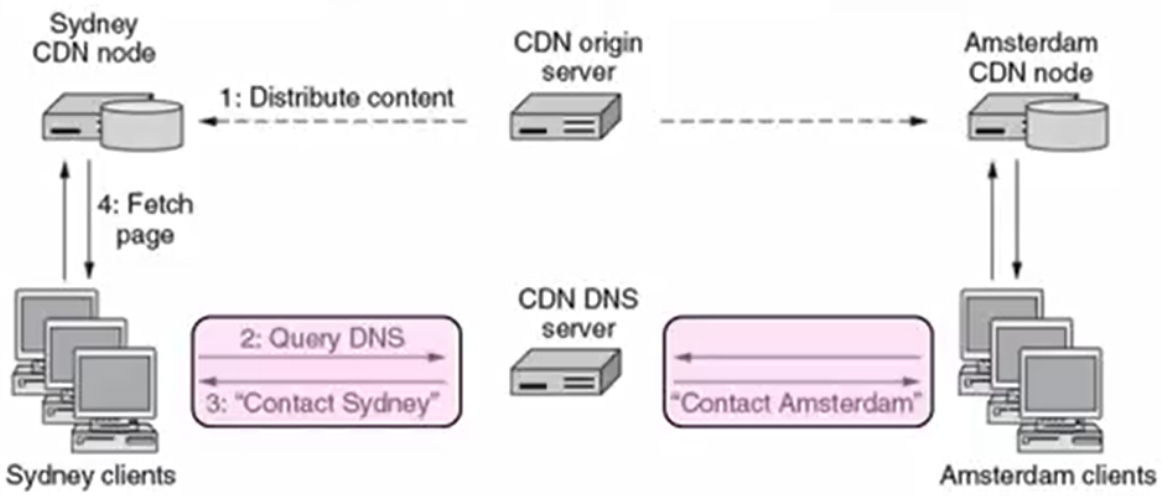
George Zipf (1902-1950)

Source: Wikipedia

How to place content near clients?

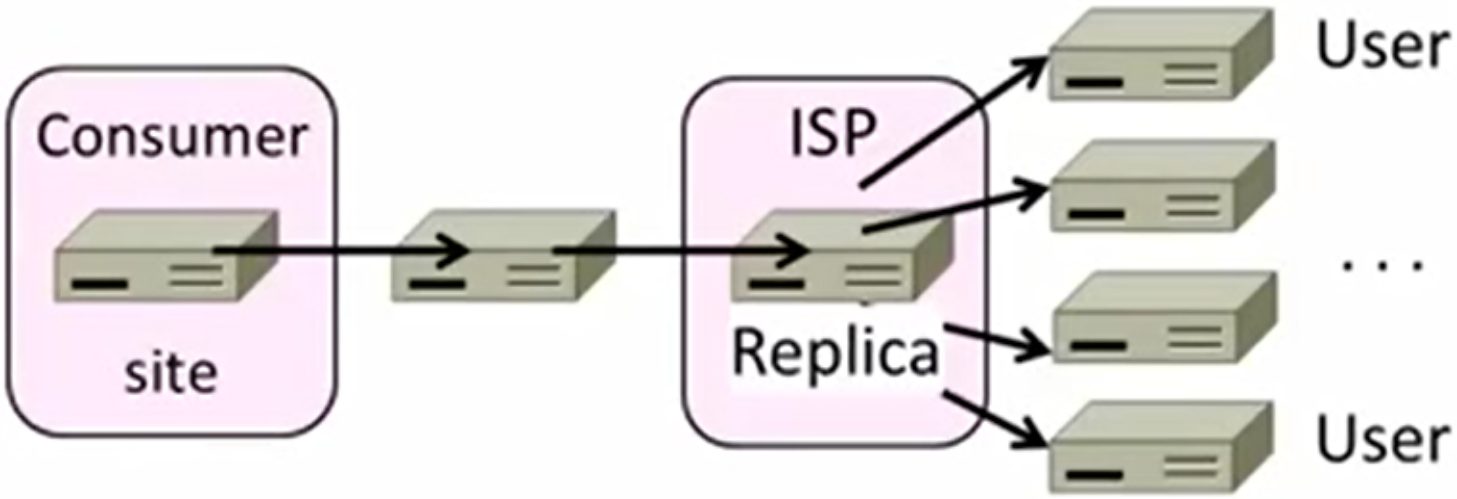
- Use browser and proxy caches
  - Helps, but limited to one client or clients in one organization
- Want to place replicas across the Internet for use by all nearby clients
  - Done by clever use of DNS – give different answers to clients based on their IPs





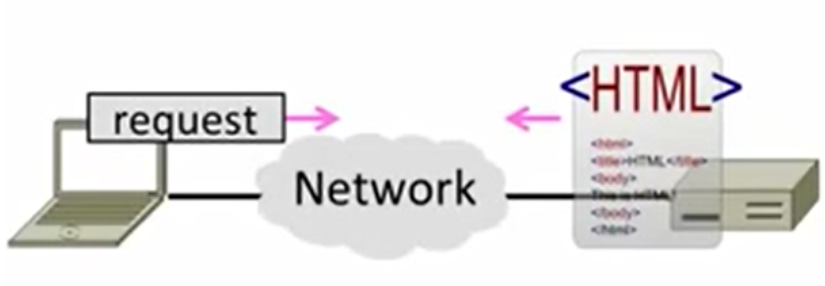
Business Model

- Clever model pioneered by Akamai
  - Placing site replica at an ISP is win-win
  - Improves site experience and reduces bandwidth usage of ISP



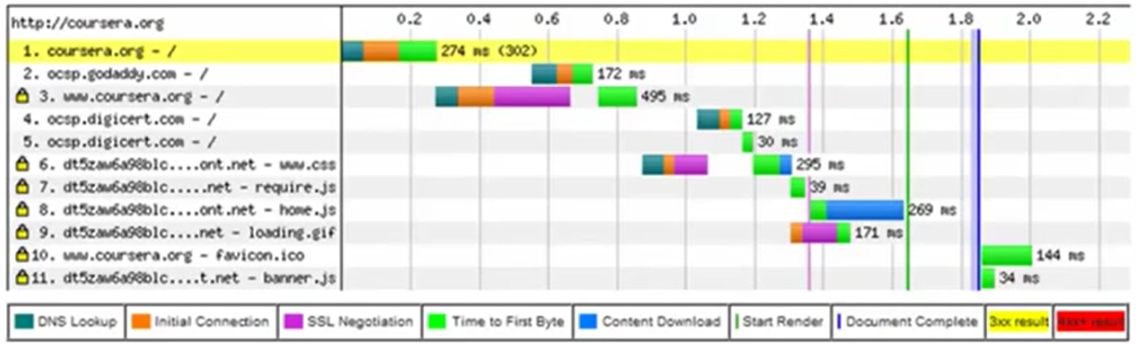
Outline

- Application Layer Overview
- Domain Name System
- HTTP, the HyperText Transfer Protocol
- HTTP Performance
- HTTP Caching and Proxies
- CDNs (Content Delivery Networks)
- The Future of HTTP
  - How will we make the web faster?
- Peer-to-Peer Content Delivery (BitTorrent)



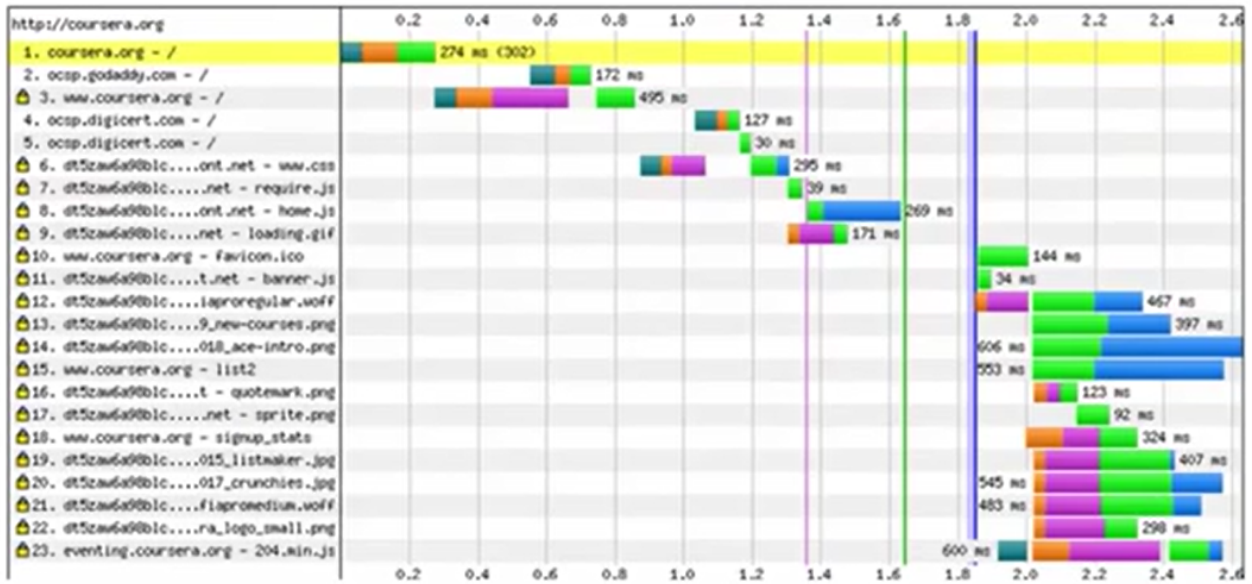
Modern Web Pages

- Waterfall diagram shows progression of page load



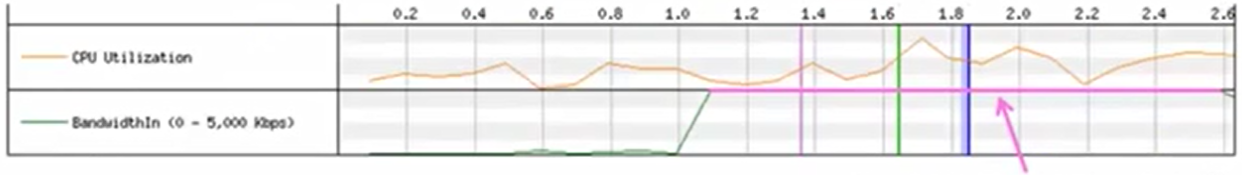
webpagetest tool for http://coursera.org (Firefox, 5/1 Mbps, from VA, 3/1/13)

23 requests; ~1 Mb data; ~2.6 secs



webpagetest tool for http://coursera.org (Firefox, 5/1 Mbps, from VA, 3/1/13)

- Waterfall and PLT depends on many factors
  - Very different for different browsers
  - Very different for repeat page views
  - Depends on local computation as well as network



Yay! (Network used well)

Recent work to reduce PLT

- Pages grow ever more complex!
  - Larger, more dynamic, and secure
  - How will we reduce PLT?

1. Better use of the network
   - HTTP/2 effort based on SPDY
2. Better content structures
   - mod_pagespeed server extension

SPDY ("speedy")

- A set of HTTP improvements
  - Multiplexed (parallel) HTTP requests on one TCP connection
  - Client priorities for parallel requests
  - Compressed HTTP headers
  - Server push of resources
- Now being tested and improved
  - Default in Chrome, Firefox
  - Basis for an HTTP/2 effort

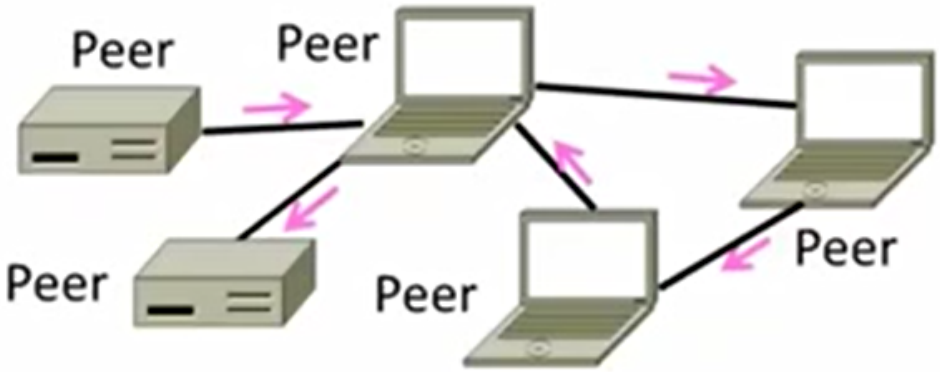
mod_pagespeed

- Observation:
  - The way pages are written affects how quickly they load
  - Many books on best practices for page authors and developers
- Key idea:
  - Have server re-write (compile) pages to help them load quickly!
  - mod_pagespeed is an example
- Apache server extension
  - Software installed with web server
  - Rewrites pages "on the fly" with rules based on best practices
- Example rewrite rules:
  - Minify Javascript
  - Flatten multi-level CSS files
  - Resize images for client
  - And much more (100s of specific rules)

https://github.com/pagespeed/mod_pagespeed

Outline

- Application Layer Overview
- Domain Name System
- HTTP, the HyperText Transfer Protocol
- HTTP Performance
- HTTP Caching and Proxies
- CDNs (content Delivery Networks)
- The Future of HTTP
- Peer-to-Peer Content Delivery (BitTorrent)
  - Runs without dedicated infrastructure
  - BitTorrent as an example

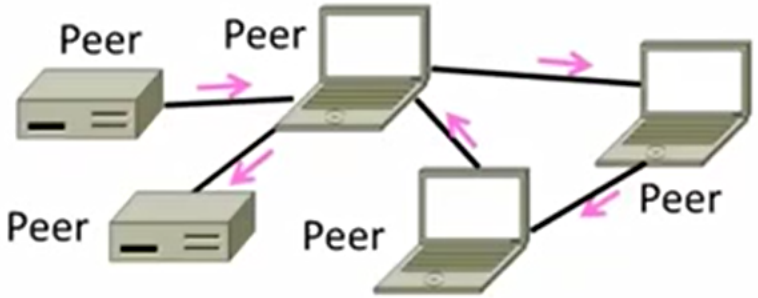


Context

- **Delivery with client/server CDNs:**
  - **Efficient, scales up for popular content**
  - **Reliable, managed for good service**
- **… but some disadvantages too:**
  - **Need for dedicated infrastructure**
  - **Centralized control/oversight**

P2P (Peer-to-Peer)

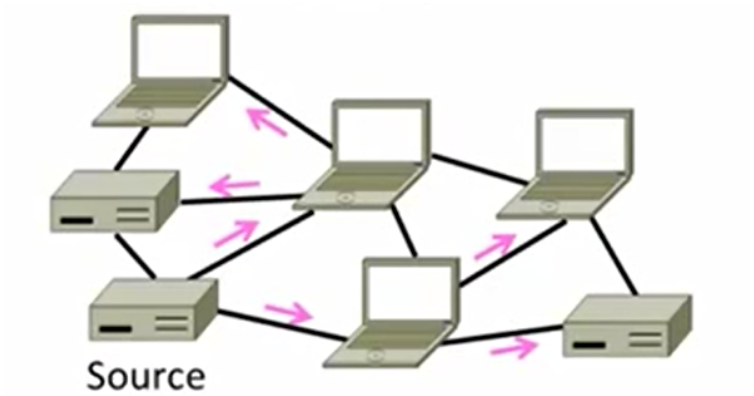
- Goal: delivery *without* dedicated infrastructure or centralized control
  - Still efficient at scale, and reliable
- Key idea: have participants (or peers) help themselves
  - Initially Napster '99 for music (gone)
  - Now BitTorrent '01 onwards (popular!)



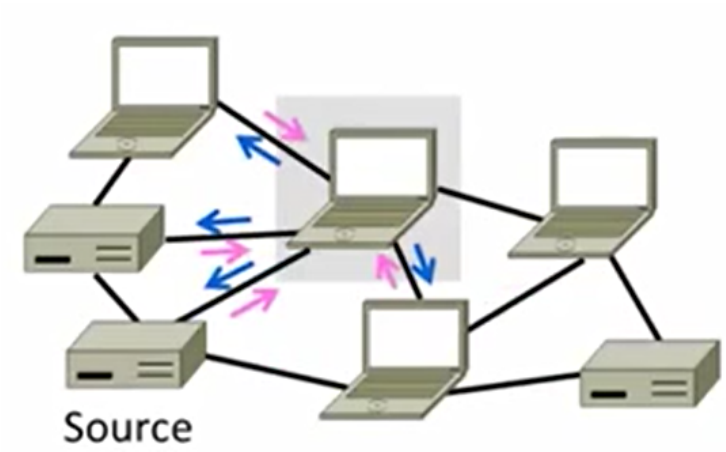
- Challenges
  - No servers on which to rely
    - Communication must be peer-to-peer and self-organizing, not client-server
    - Leads to several issues at scale …
  - Limited capabilities
    - How can one peer deliver content to all other peers?
  - Participation incentives
    - Why will peers help each other?
  - Decentralization
    - How will peers find content?

Solutions

- **Overcoming Limited Capabilities**
  - Peer can send content to all other peers using a distribution tree
    - Typically done with replicas over time
    - Self-scaling capacity



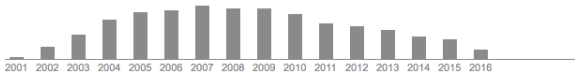
- **Providing Participation Incentives**
  - Peer play two roles:
    - Download (→) to help themselves, and upload (←) to help others
  - Couple the two roles:
    - I'll upload for you if you upload for me
    - Encourages cooperation



- **Enabling Decentralization**
  - Peer must learn where to get content
    - Use DHTs (Distributed Hash Tables)
  - DHTs are fully-decentralized, efficient algorithms for a distributed index
    - Index is spread across all peers
    - Index lists peers to contact for content
    - Any peer can lookup the index
    - Started as academic work in 2001

Chord: A scalable peer-to-peer lookup service for internet applications

Authors	Ion Stoica, Robert Morris, David Karger, M Frans Kaashoek, Hari Balakrishnan
Publication date	2001/10/1
Journal	ACM SIGCOMM Computer Communication Review
Volume	31
Issue	4
Pages	149-160
Publisher	ACM
Description	Abstract A fundamental problem that confronts peer-to-peer applications is to efficiently locate the node that stores a particular data item. This paper presents Chord, a distributed lookup protocol that addresses this problem. Chord provides support for just one operation: given a key, it maps the key onto a node. Data location can be easily implemented on top of Chord by associating a key with each data item, and storing the key/data item pair at the node to which the key maps. Chord adapts efficiently as nodes join and leave the system, ...
Total citations	Cited by 15654



Scholar articles
Chord: A scalable peer-to-peer lookup service for internet applications
I Stoica, R Morris, D Karger, MF Kaashoek… - ACM SIGCOMM Computer Communication Review, 2001
Cited by 12465 - Related articles - All 415 versions

Chord: a scalable peer-to-peer lookup protocol for internet applications *
I Stoica, R Morris, D Liben-Nowell, DR Karger… - IEEE/ACM Transactions on Networking (TON), 2003
Cited by 3545 - Related articles - All 53 versions

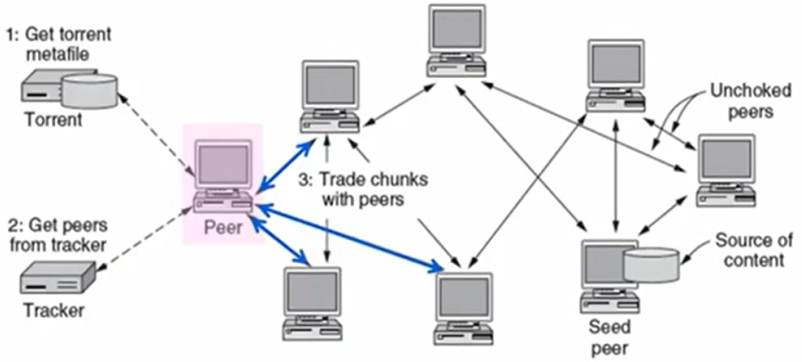
BitTorrent

- Main P2P system in use today
  - Developed by Conhen in '01
  - Very rapid growth, large transfers
  - Much of the Internet traffic today!
  - Used for legal and illegal content
- Delivers data using "torrents":
  - Transfers files in pieces for parallelism
  - Notable for treatment of incentives
  - Tracker or decentralized index (DHT)

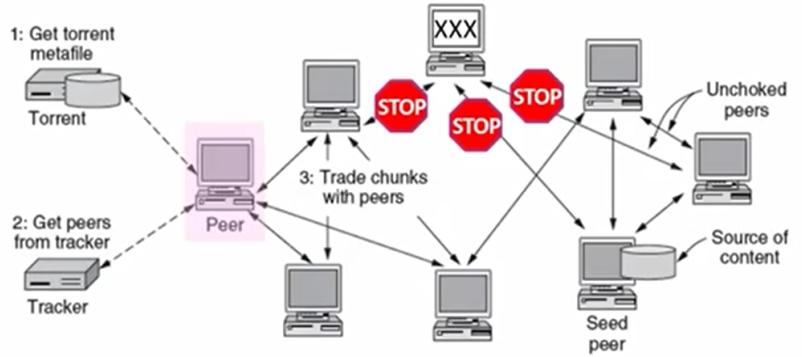
BitTorrent Protocol

- Steps to download a torrent:
  1. Start with torrent description
  2. Contact tracker to join and get list of peers (with at least seed peer)
  3. Or, use DHT index for peers
  4. Trade pieces with different peers
  5. Favor peers that upload to you rapidly; "choke" peers that don't by slowing your upload to them

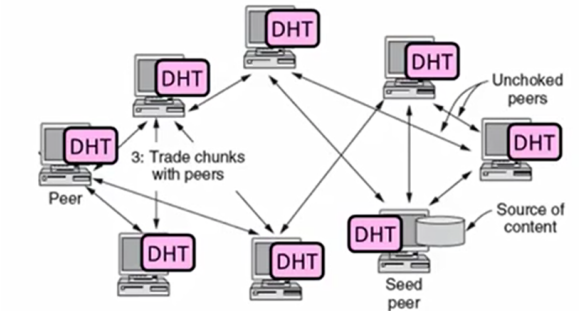
All peers (except seed) retrieve torrent at the same time, dividing file into pieces gives parallelism for speed



Choking unhelpful peers encourages participation



DHT index (spread over peers) is fully decentralized



P2P Outlook

- Alternative to CDN-style client-server content distribution
  - With potential advantages
- P2P and DHT technologies finding more widespread use over time
  - E.g., part of skype, Amazon
  - Expect hybrid systems in the future
- What's the problem with P2P?
  - Check P4P out: http://codex.cs.yale.edu/avi/home-page/p4p-dir/p4p.html