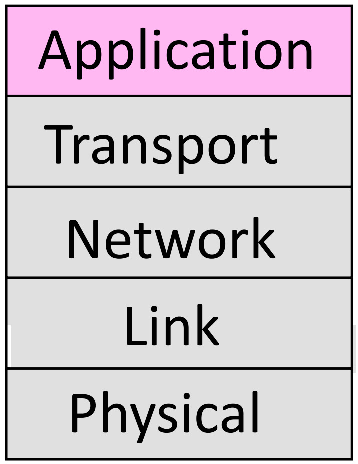
Operating Systems and Networks

Network Lecture 12: Application Layer

Adrian Perrig
Network Security Group
ETH Zürich

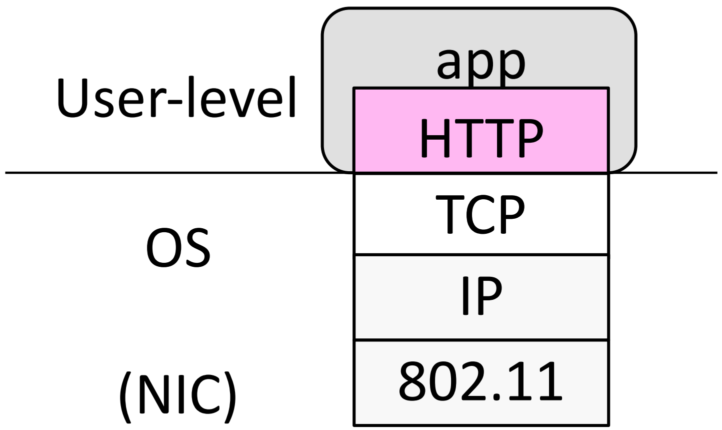
Where we are in the Course

- Starting the Application Layer!
  - Builds distributed "network services" (DNS, Web) on Transport services



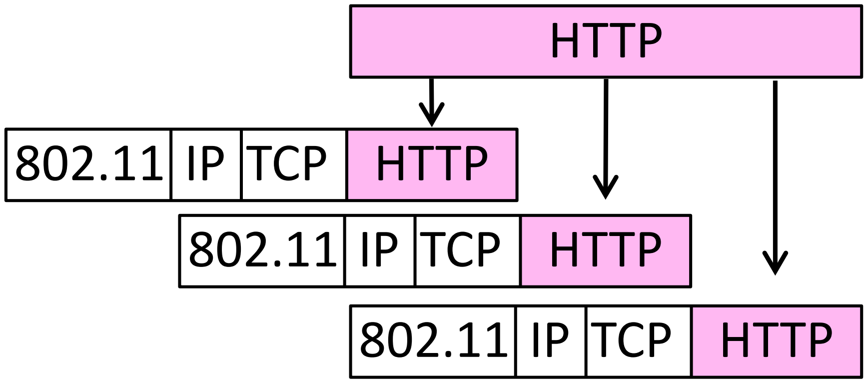
Recall

- Application layer protocols are often part of an "app"
  - But don't need a GUI, e.g., DNS



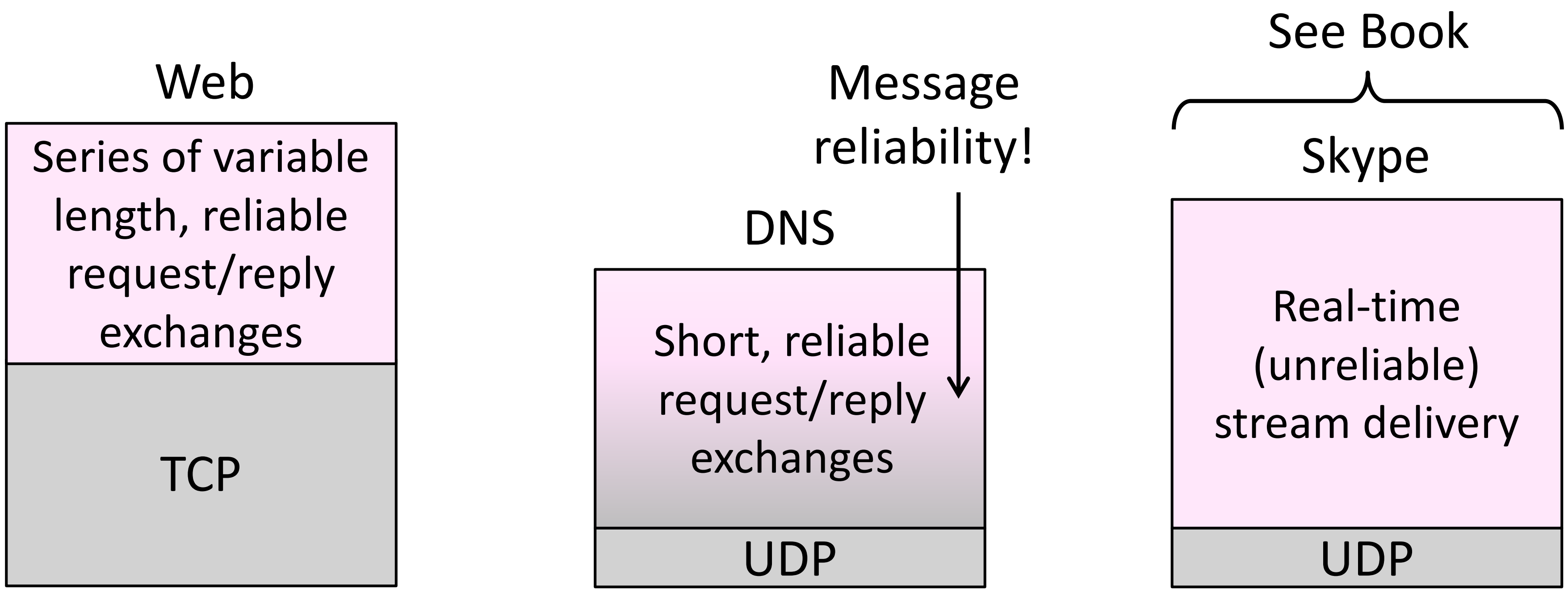
Recall (2)

- 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! (applies to layers 6 and 5)

7	Application	– Provides functions needed by users
6	Presentation	– Converts different data representations
5	Session	– Multiple sessions between same src-dst
4	Transport	– Provides end-to-end delivery
3	Network	– Sends packets over multiple links
2	Data link	– Sends frames of information
1	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 images
  - 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 content encoding
  - Application headers are often simple and readable versus packed for efficiency

Topics

- Evolving Internet applications
- DNS (Domain Name System)
- HTTP (HyperText Transfer Protocol)
- Web proxies and caching
- Content Distribution Networks
- Peer-to-peer (BitTorrent)

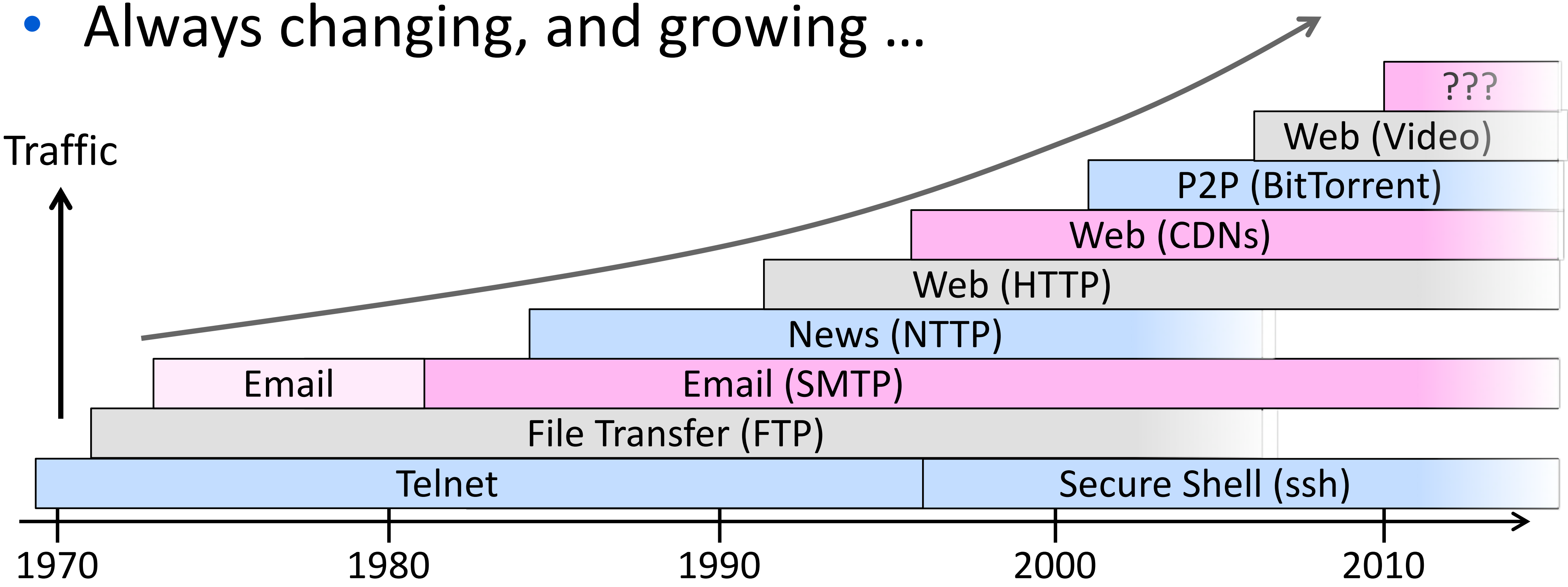
} This lecture

- Real-time applications (VoIP)

} See Book

Evolution of Internet Applications

- Always changing, and growing ...

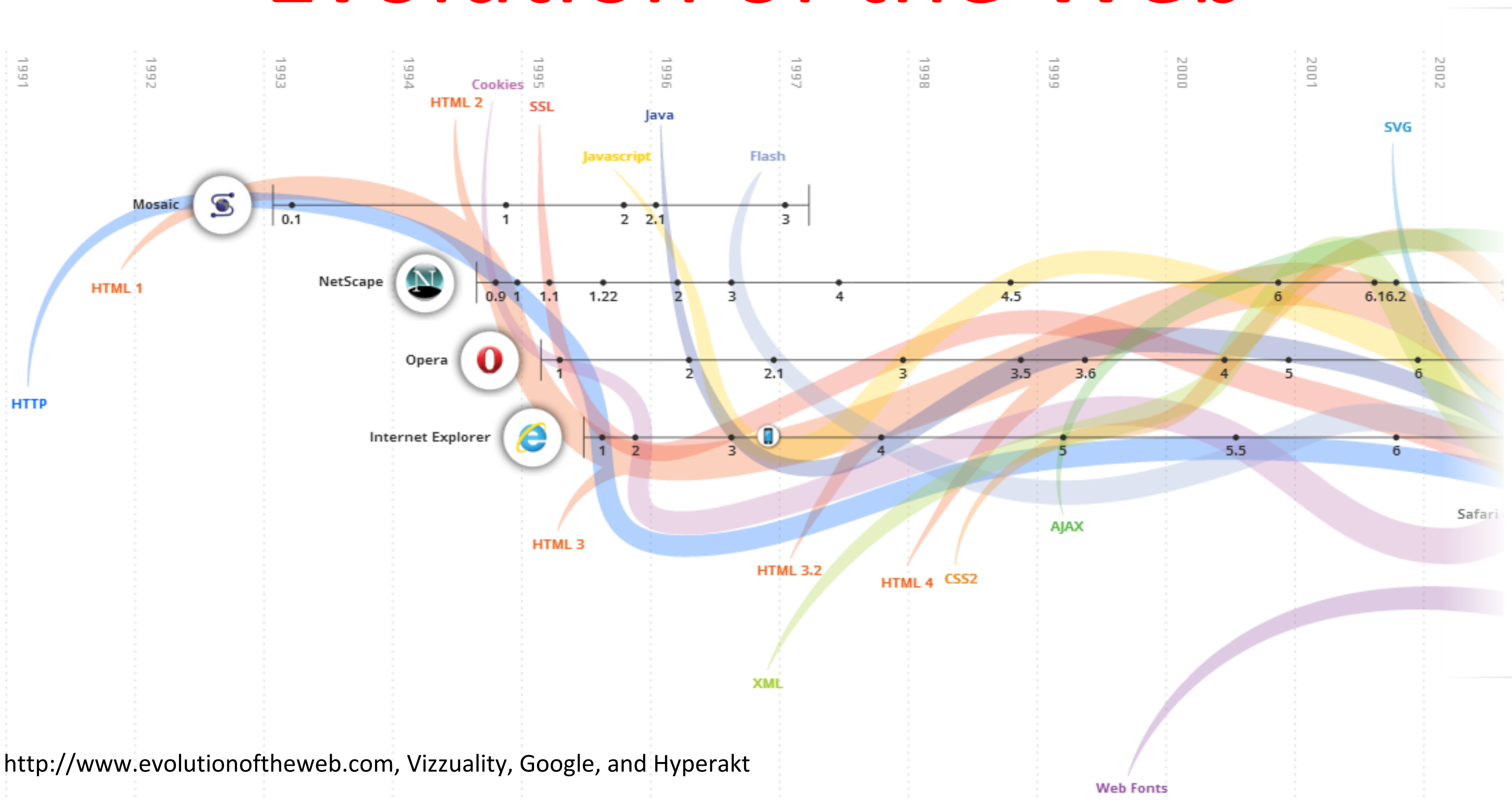


Evolution of Internet Applications (2)

- 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

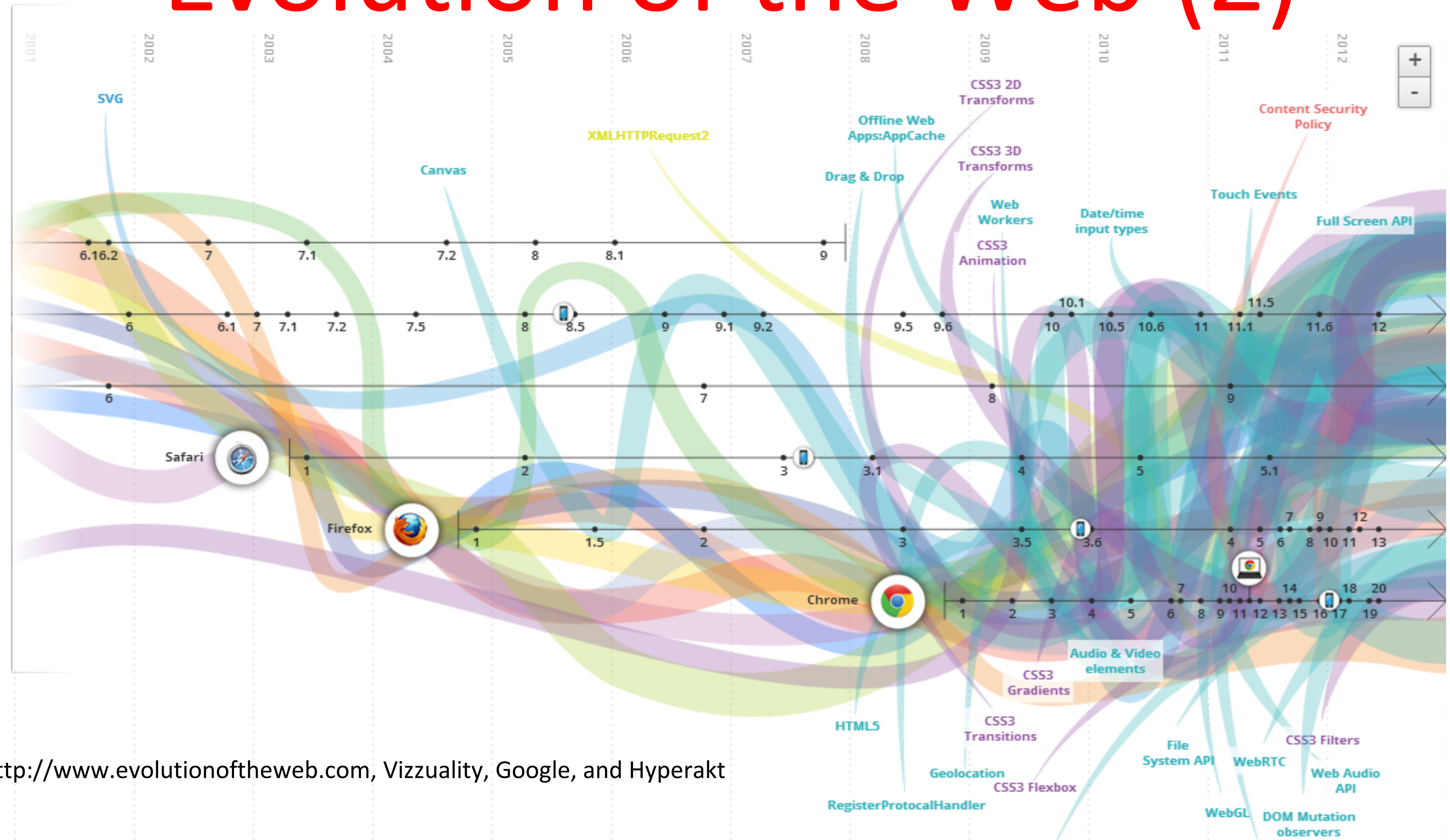
- Robust Internet growth, esp. video, wireless and mobile
  - Most traffic is video, will be 90% of Internet in a few years
  - Wireless traffic will soon overtake wired traffic
  - Mobile traffic is still a small portion (15%) of overall
  - Growing attack traffic from China, also U.S. and Russia

Evolution of the Web



Source: http://www.evolutionoftheweb.com, Vizzuality, Google, and Hyperakt

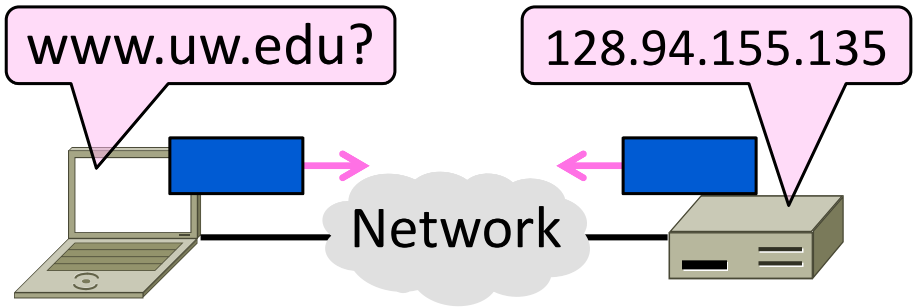
Evolution of the Web (2)



Source: http://www.evolutionoftheweb.com, Vizzuality, Google, and Hyperakt

Domain Name System (DNS) (§7.1.1-7.1.3)

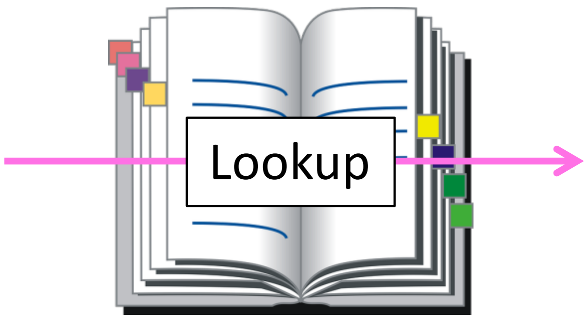
- The DNS (Domain Name System)
  - Human-readable host names, and more
  - Part 1: the distributed namespace



Names and Addresses

- Names: higher-level (user-understandable) resource identifiers
- Addresses: lower-level resource locators
  - Multiple levels, e.g., full name → email → IP address → Ethernet address
- Resolution (or lookup): mapping a name to an address

Name, e.g.
“Andy Tanenbaum,”
or “flits.cs.vu.nl”



Directory

Address, e.g.
“Vrije Universiteit, Amsterdam”
or IPv4 “130.30.27.38”

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) ~1985

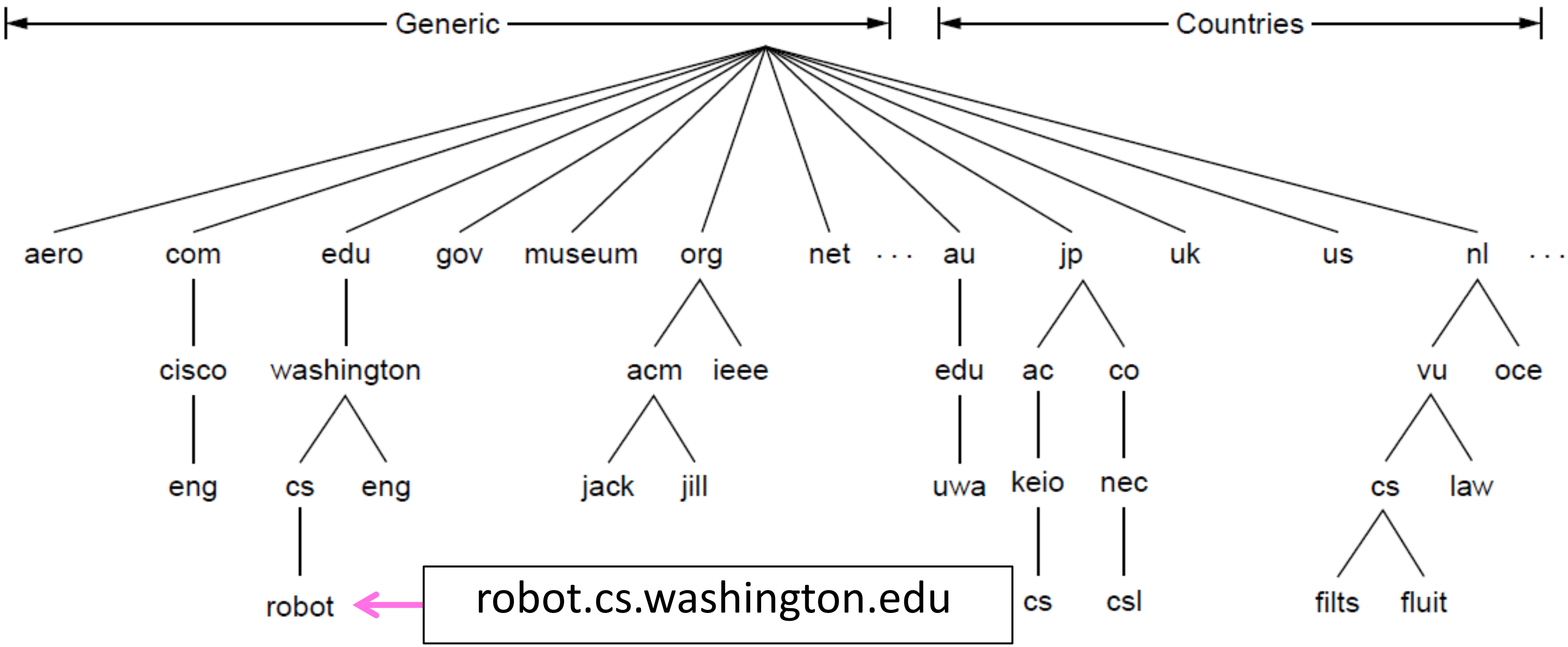
- Neither manageable nor efficient as the ARPANET grew …

DNS

- A naming service to map between host names and their IP addresses (and more)
  - www.uwa.edu.au → 130.95.128.140
- Goals
  - Easy to manage (especially 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)

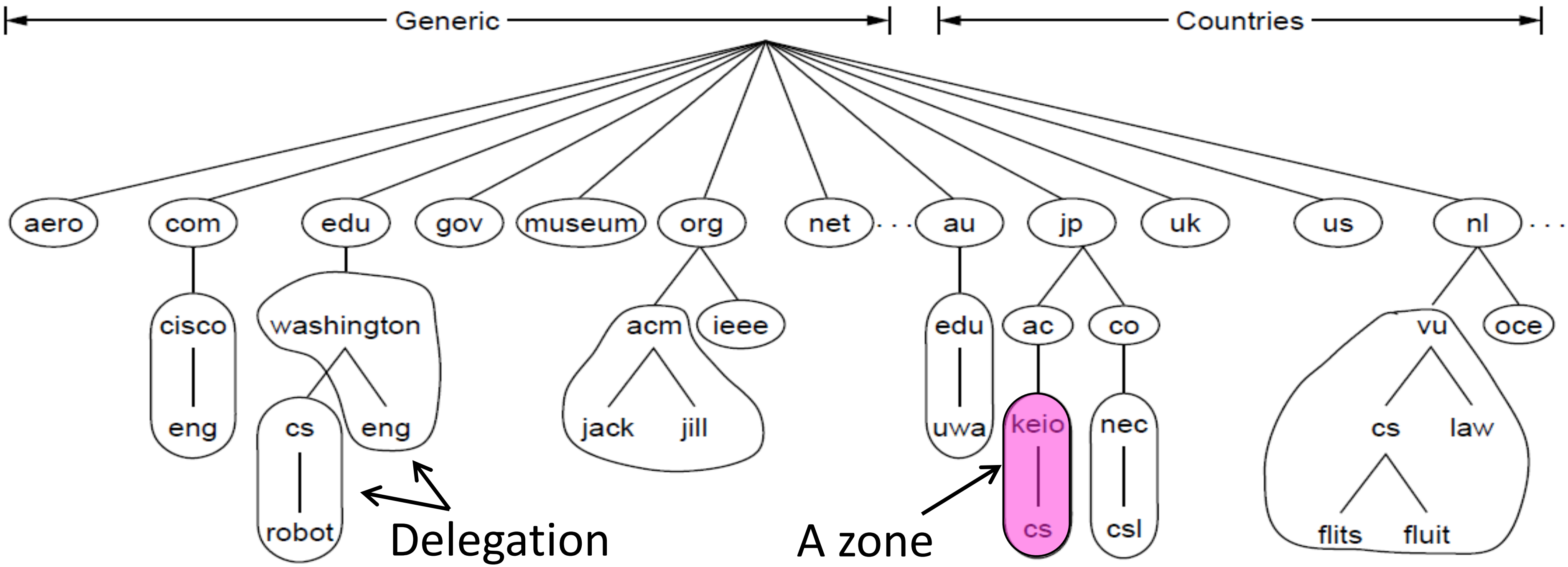


TLDs (Top-Level Domains)

- 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, .info, .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



DNS Zones (2)

- Zones are the basis for distribution
  - EDU Registrar administers .edu
  - UW administers washington.edu
  - CS&E administers cs.washington.edu
- Each zone has a <u>nameserver</u> 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 provide information about its domain names

Type	Meaning
SOA	Start of authority, has main zone parameters
A	IPv4 address of a host
AAAA (“quad A”)	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 Resource Records (2)

; Authoritative 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
star	86400	IN	A	130.37.56.205
zephyr	86400	IN	A	130.37.20.10
top	86400	IN	A	130.37.20.11
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
little-sister		IN	A	130.37.62.23
laserjet		IN	A	192.31.231.216

← Name server

← IP addresses of computers

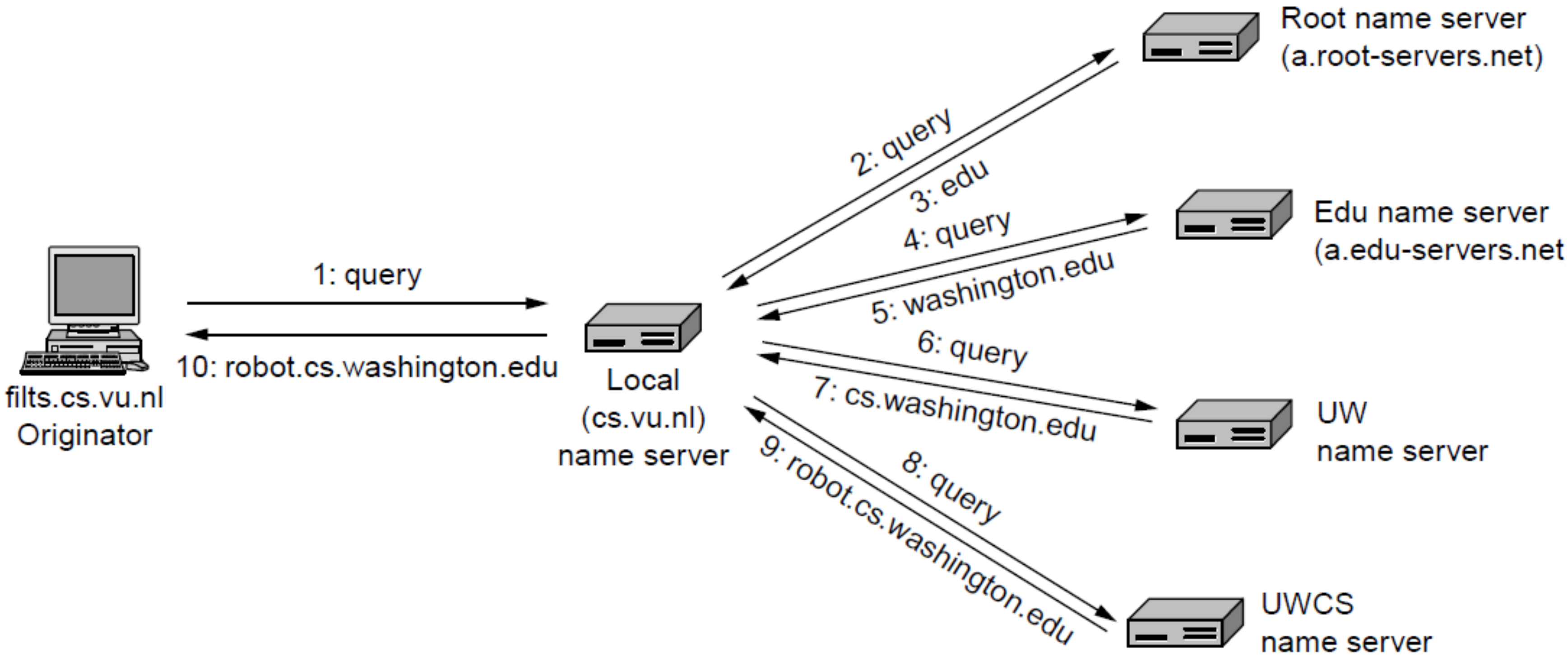
← Mail gateways

DNS Resolution

- 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 …

DNS Resolution (2)

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



Iterative vs. Recursive Queries

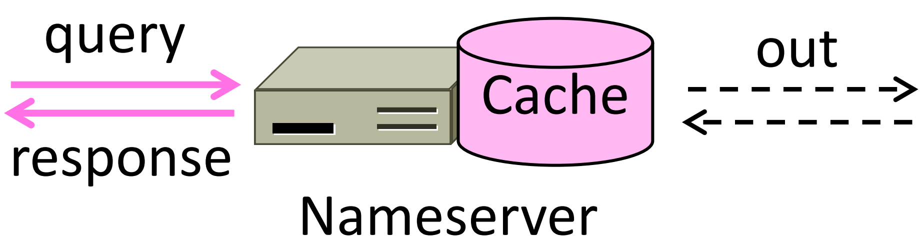
- Recursive query
  - Nameserver completes resolution and returns the final answer
  - E.g., flits → local nameserver
- Iterative query
  - Nameserver returns the answer or who to contact next for the answer
  - E.g., local nameserver → all others

Iterative vs. Recursive Queries (2)

- 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

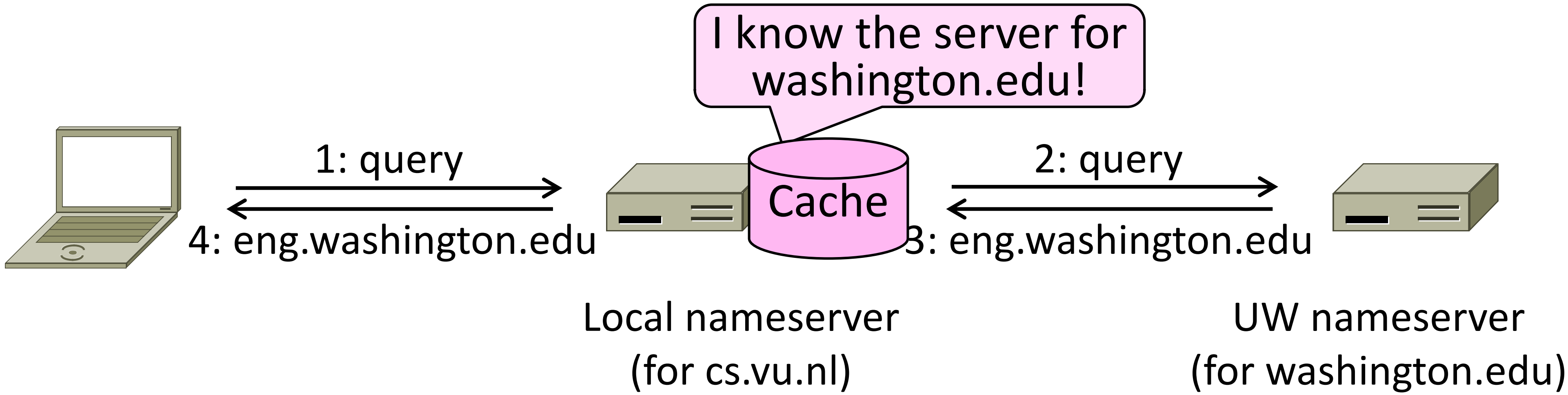
Caching

- 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



Caching (2)

- flits.cs.vu.nl now resolves eng.washington.edu
  - And previous resolutions cut out most of the process



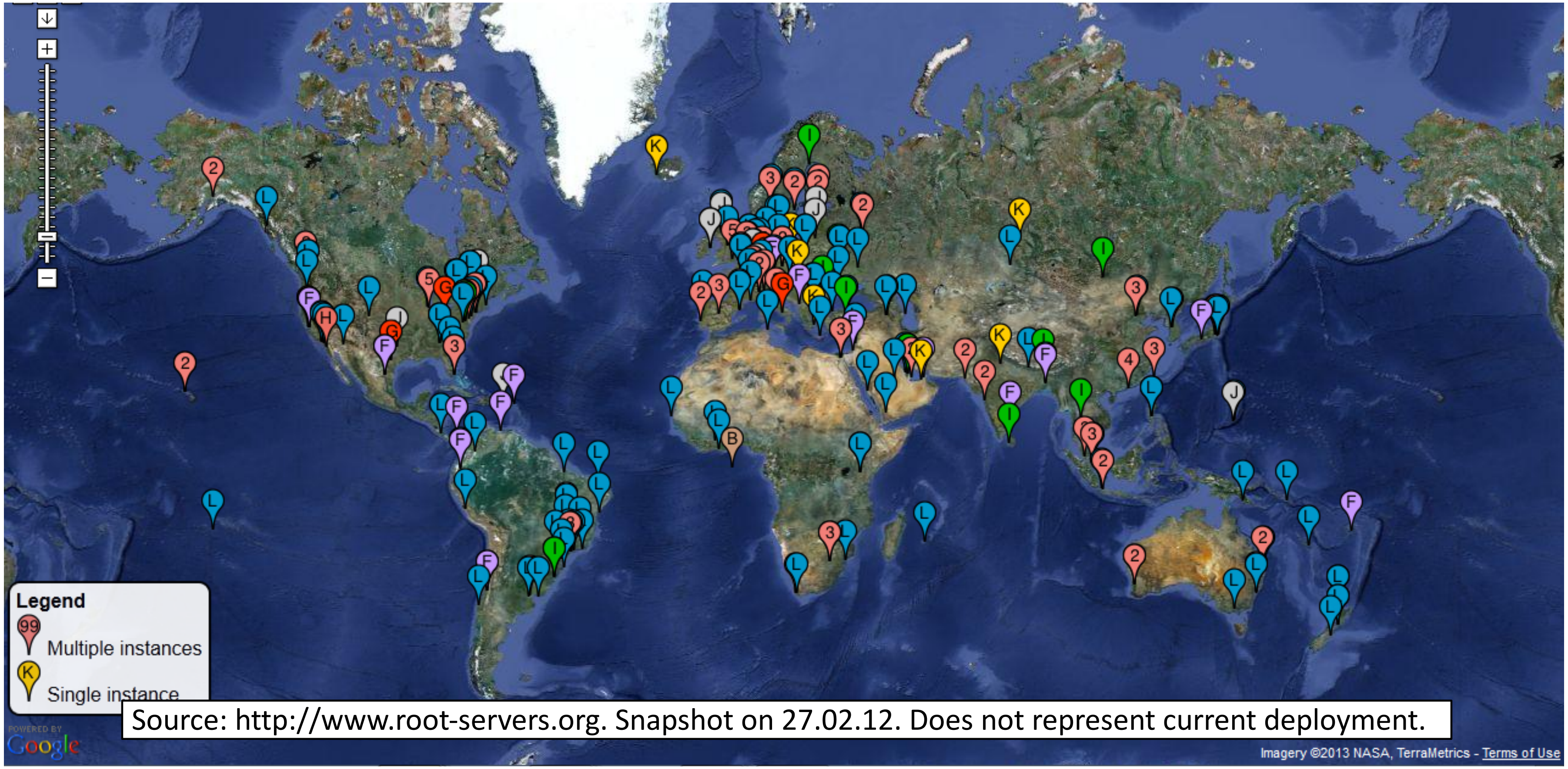
Local Nameservers

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

Root Nameservers

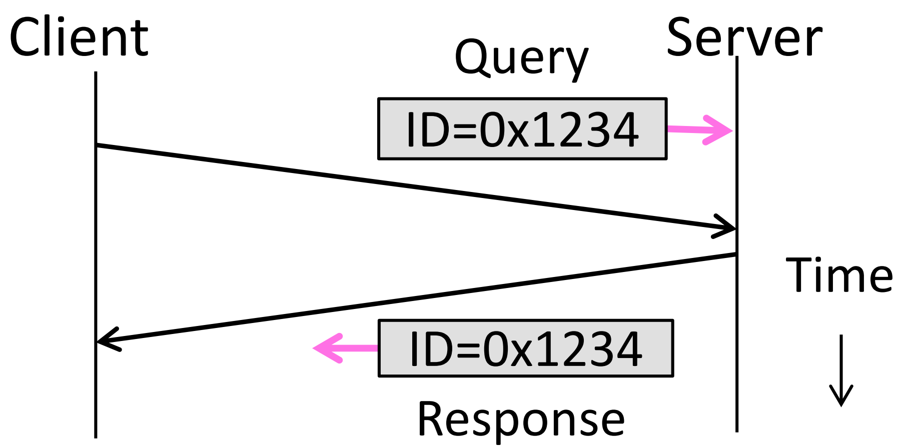
- Root (dot) is served by 13 server names
  - a.root-servers.net to m.root-servers.net
  - All nameservers need root IP addresses
  - Handled via configuration file (named.ca)
- There are >250 distributed server instances
  - Highly reachable, reliable service
  - Most servers are reached by IP anycast (Multiple locations advertise same IP! Routes take client to the closest one. See §5.2.9)
  - Servers are IPv4 and IPv6 reachable

Root Server Deployment



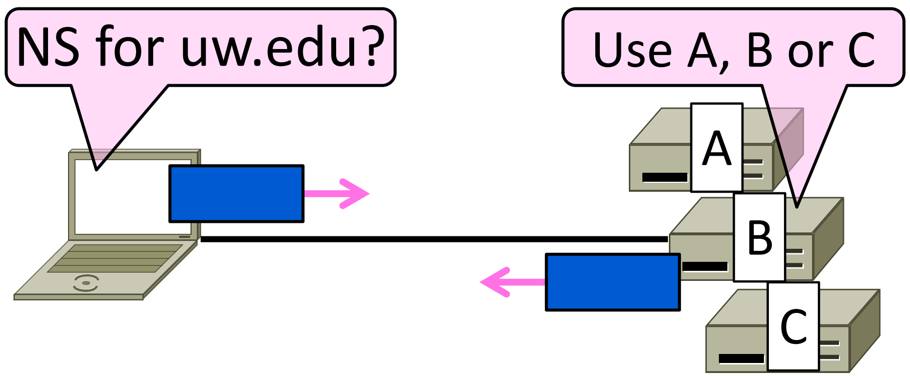
DNS Protocol

- Query and response messages
  - Built on UDP messages, port 53
  - ARQ for reliability; server is stateless!
  - Messages linked by a 16-bit ID field



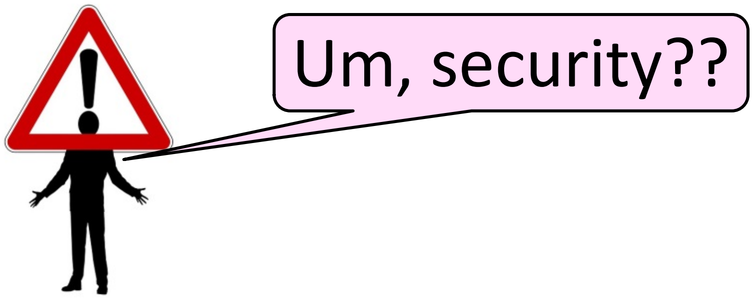
DNS Protocol (2)

- Service reliability via replicas
  - Run multiple nameservers for domain
  - Return the list; clients use one answer
  - Helps distribute load too



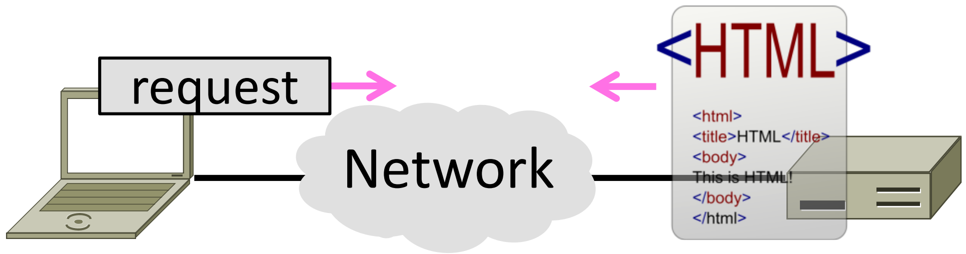
DNS Protocol (3)

- Security is a major issue
  - Compromise redirects to wrong site!
  - Not part of initial protocols ..
- DNSSEC (DNS Security Extensions)
  - Long under development, now partially deployed



HTTP, the HyperText Transfer Protocol (§7.3.1-7.3.4)

- HTTP, (HyperText Transfer Protocol)
  - Basis for fetching Web pages

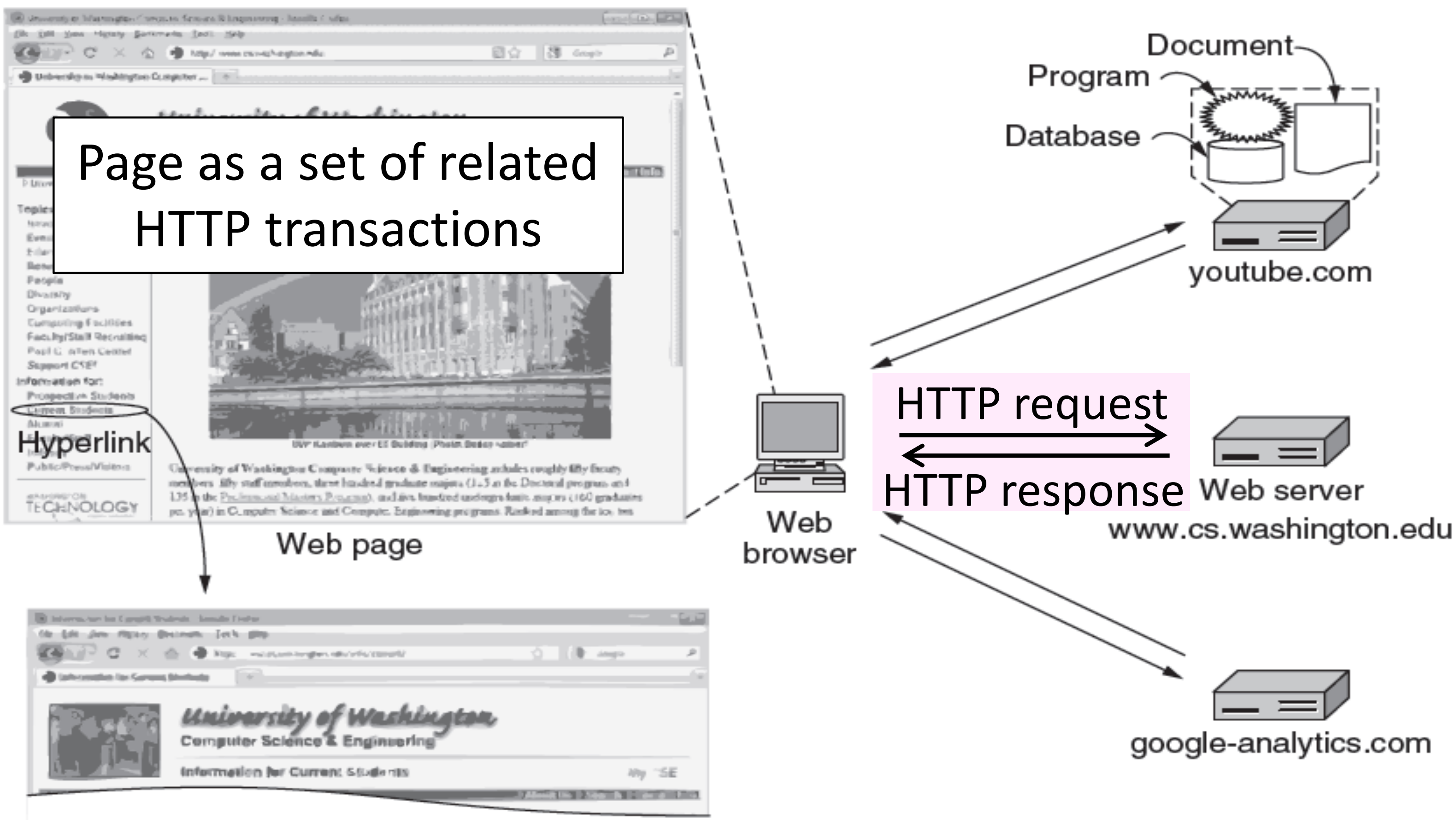


Sir Tim Berners-Lee (1955–)

- 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

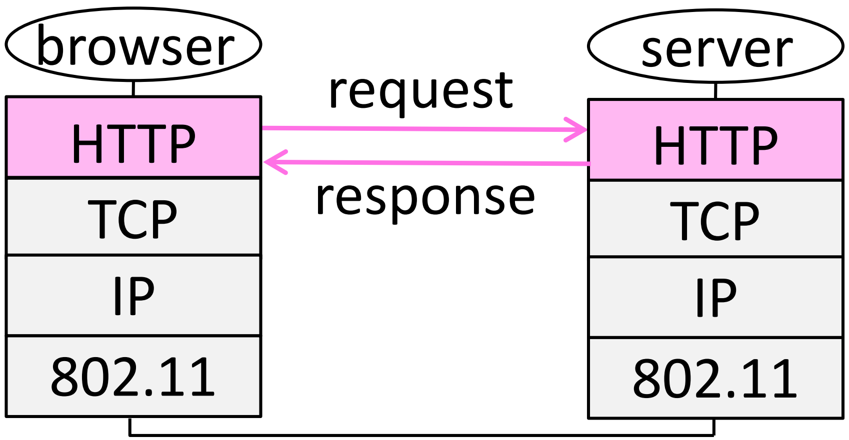
Web Context



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

Web Protocol 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:

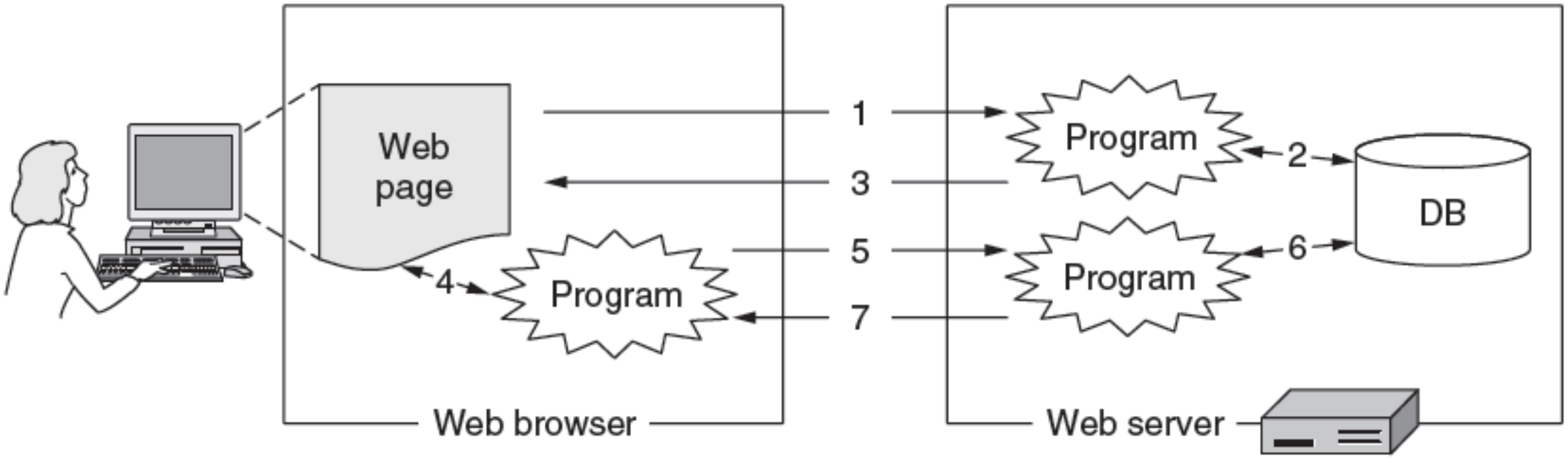
  http://de.wikipedia.org/wiki/Chuchichäschtli

  Protocol: http — Server: de.wikipedia.org — Page on server: /wiki/Chuchichäschtli

- 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

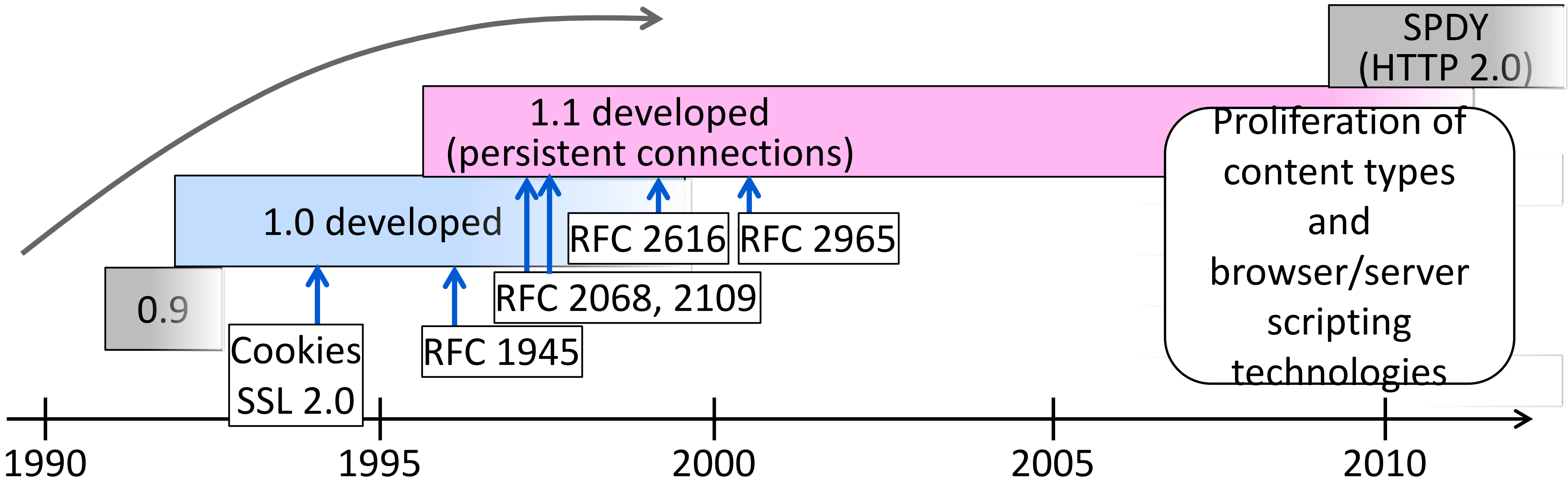
Static vs Dynamic Web pages

- Static web page is content of a file, 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 www.scion-architecture.net 80”
  - Type “GET / HTTP/1.1” followed by “Host: www.scion-architecture.net” followed by a blank line
  - Server will return HTTP response with the page contents (or other info)

HTTP Get Result

```
$ telnet www.scion-architecture.net 80
Trying 129.132.85.42...
Connected to scion-architecture.net.
Escape character is '^]'.
GET / HTTP/1.1
Host: www.scion-architecture.net

HTTP/1.1 200 OK
Date: Wed, 01 Jun 2016 21:04:24 GMT
Server: Apache/2.2.15 (Red Hat)
X-Powered-By: PHP/5.3.3
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

3565

<!DOCTYPE HTML>

<html>
...
```

HTTP Protocol (2)

- Commands used in the request

Fetch page → GET

Upload data → POST

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

HTTP Protocol (3)

- Codes returned with the 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

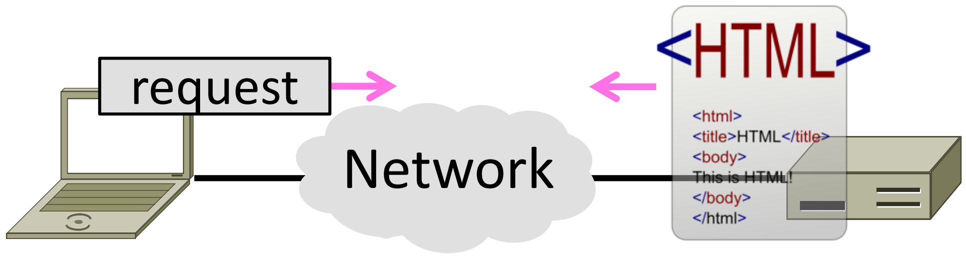
HTTP Protocol (4)

- Many header fields specify capabilities and content
  - E.g., Content-Type: text/html, Cookie: lect=12-1-http

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

HTTP Performance (§7.3.4)

- Performance of HTTP
  - Parallel and persistent connections



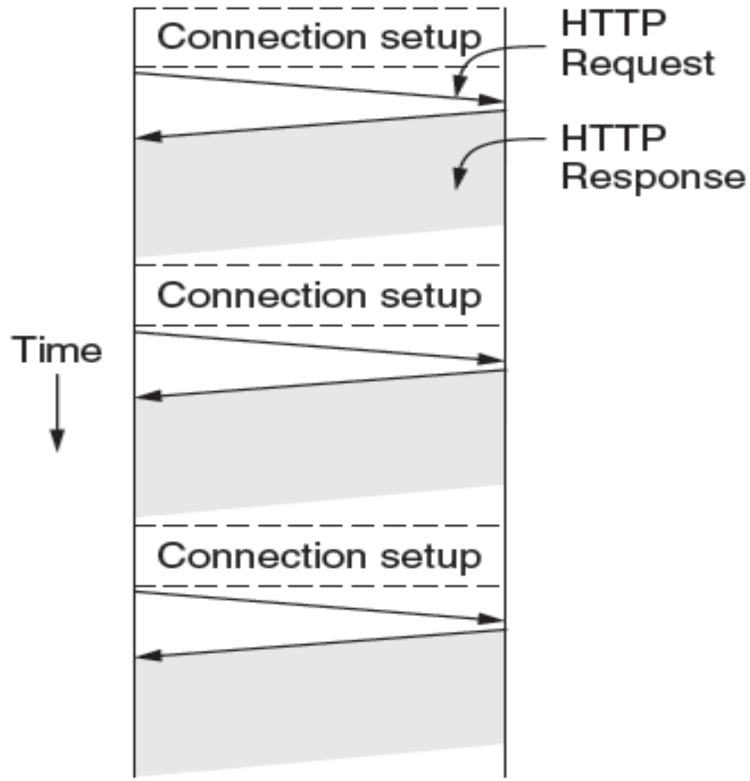
PLT (Page Load Time)

- PLT is the key measure of web performance
  - From click until user sees page
  - Small increases in PLT decrease sales

- PLT depends on many factors
  - Structure of page/content
  - HTTP (and TCP!) protocol
  - Network RTT and bandwidth

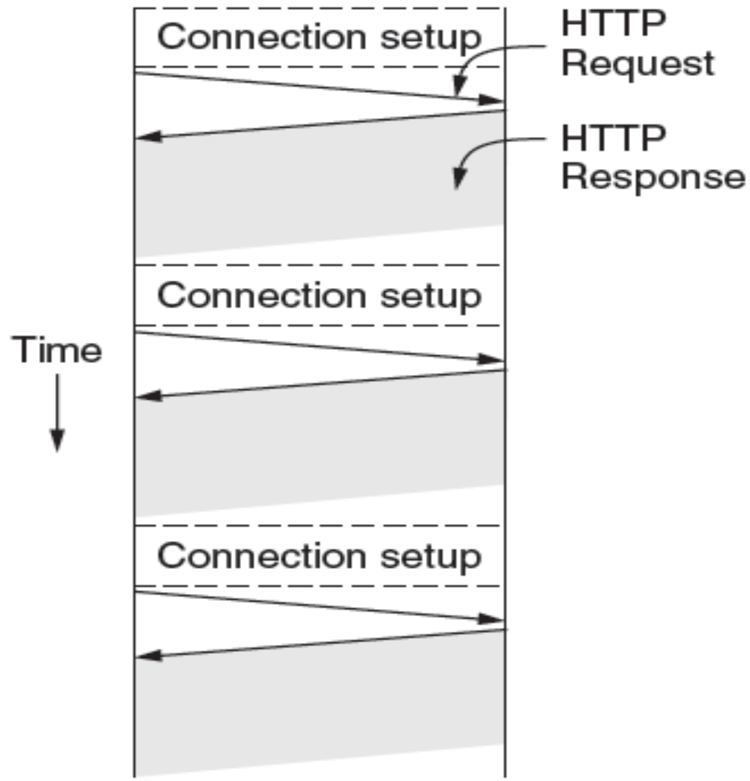
Early Performance (1)

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



Early Performance (2)

- 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
- Network is not used effectively
  - Worse with many small resources / page



Ways to Decrease PLT

1. Reduce content size for transfer
   - Smaller images, gzip
2. Change HTTP to make better use of available bandwidth (e.g., avoid TCP slow start)
3. Change HTTP to avoid repeated transfers of the same content
   - Caching, and proxies
4. Move content closer to client
   - CDNs [later]

Parallel Connections

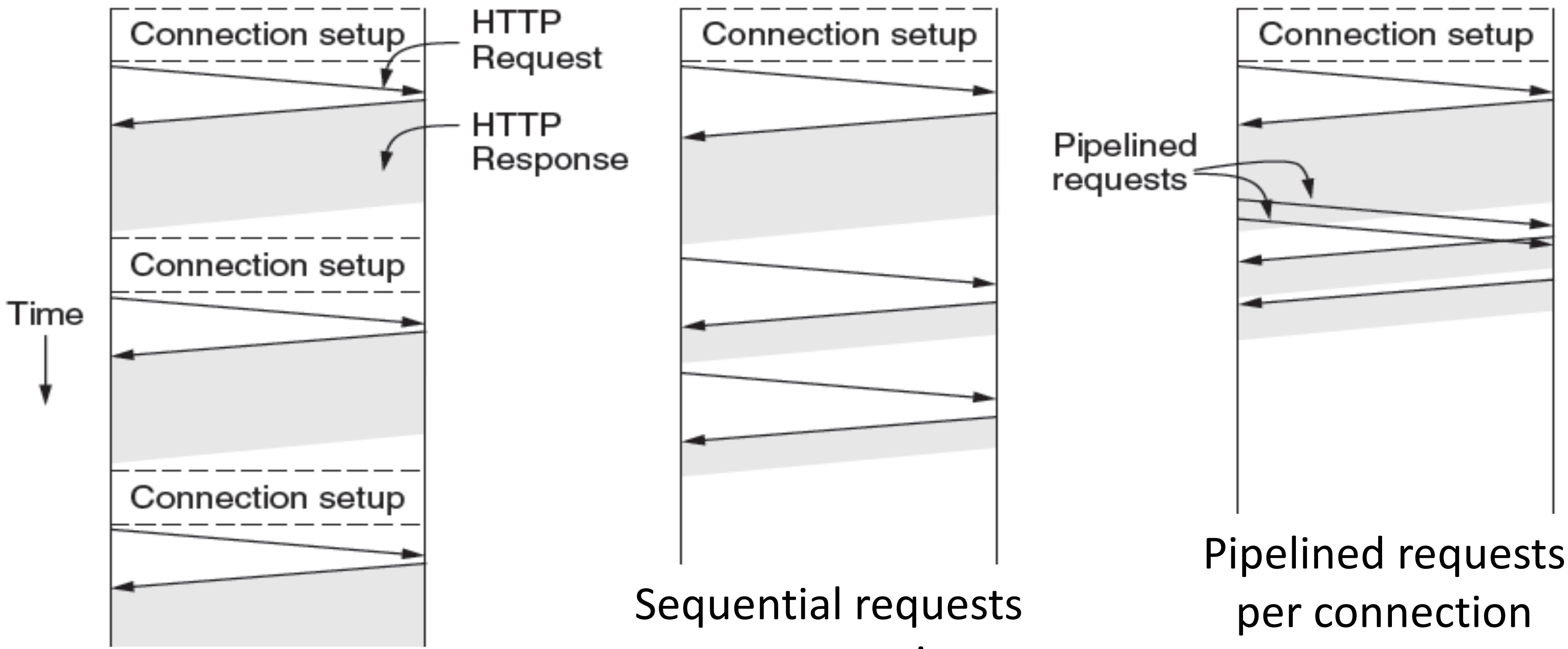
- One simple way to reduce PLT
  - Browser runs multiple (8, say) HTTP instances in parallel
  - Server is unchanged; already handles concurrent requests for many clients
- How does this help?
  - Single HTTP wasn’t using network much …
  - So parallel connections aren’t slowed much
  - Pulls in completion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

Persistent Connections (2)



One request per connection

Sequential requests per connection

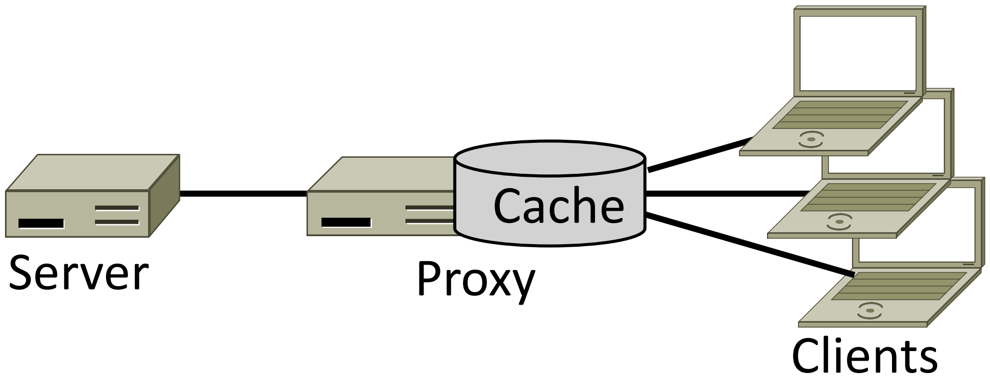
Pipelined requests per connection

Persistent Connections (3)

- 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?)

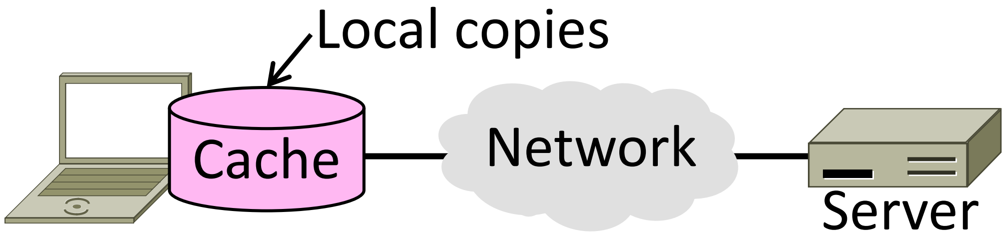
HTTP Caching and Proxies (§7.3.4, §7.5.2)

- HTTP caching and proxies
  - Enabling content reuse



Web Caching

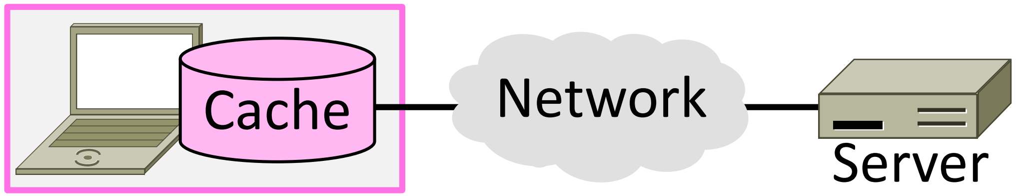
- Users often revisit web pages
  - Big win from reusing local copy!
  - This is caching



- Key question:
  - When is it OK to reuse local copy?

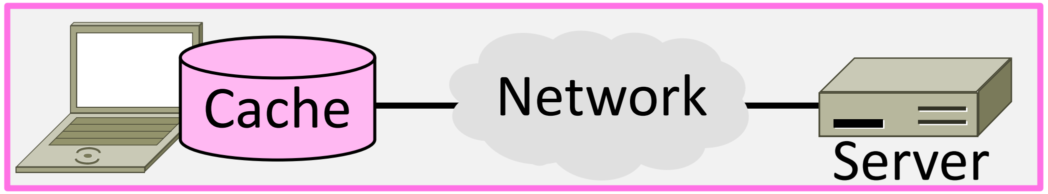
Web Caching (2)

- Locally determine if copy is still valid
  - Based on expiry information such as “Expires” header from server
  - Or use a heuristic to guess (cacheable, freshly valid, not modified recently)
  - Content is then available right away



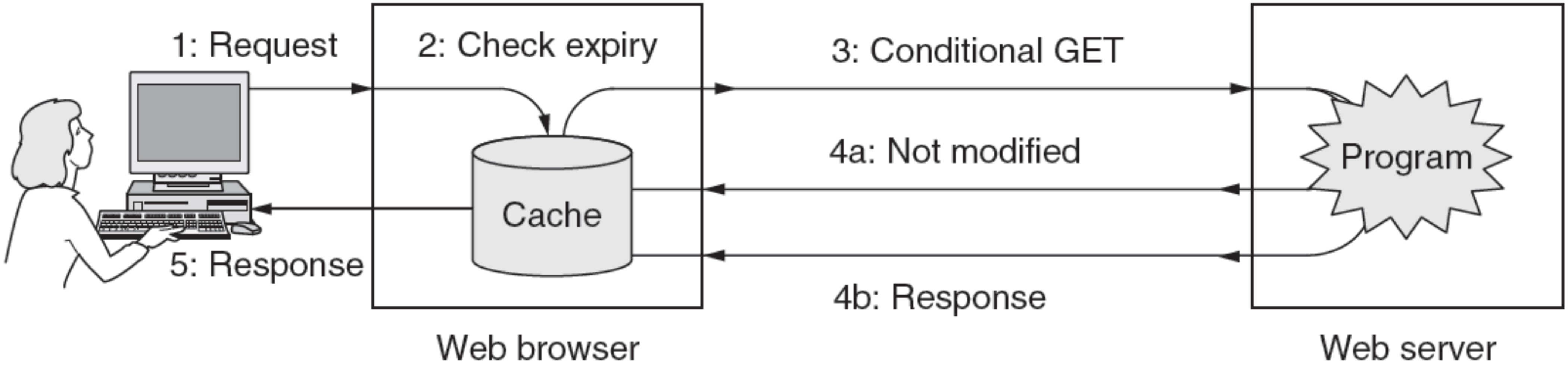
Web Caching (3)

- Revalidate copy with remote server
  - Based on timestamp of copy such as “Last-Modified” header from server
  - Or based on content such as “ETag” header from server: Entity Tag, computed by server as a unique object identifier
  - Content is available after 1 RTT



Web Caching (4)

- Putting the pieces together:

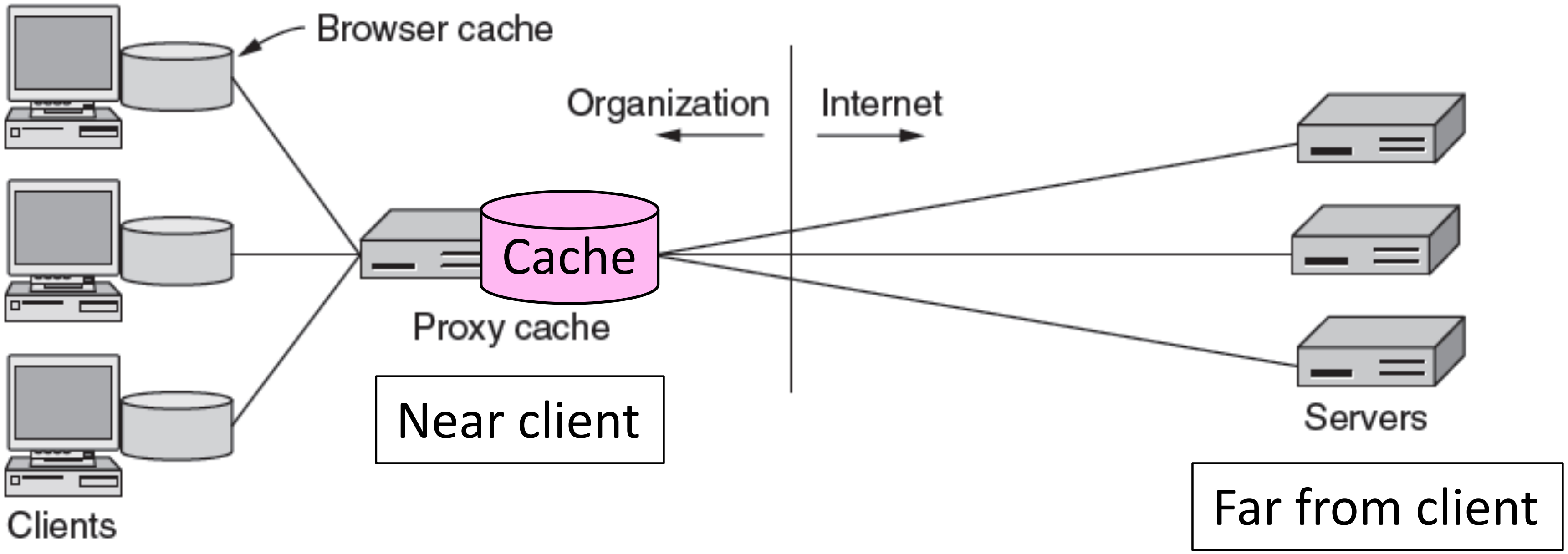


Web Proxies

- Place intermediary between pool of clients and external web servers
- Proxy caching
  - Clients benefit from larger, shared cache (other clients may have already accessed content)
  - Benefits limited by secure / dynamic per-client content, as well as "long tail" data access pattern
  - Enables application of organizational access policies

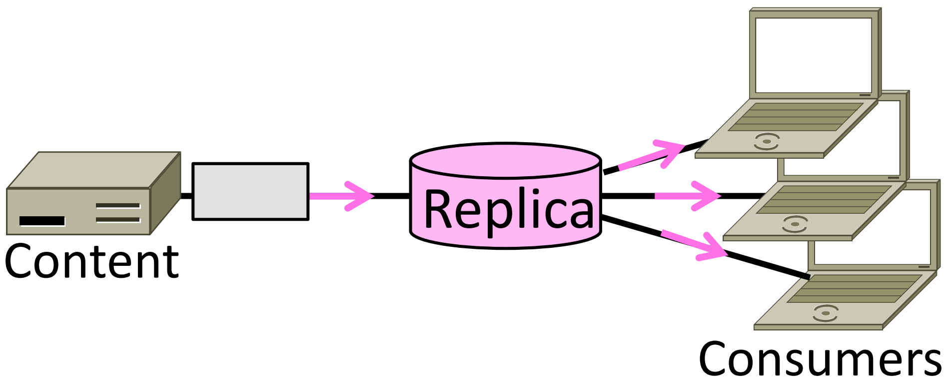
Web Proxies (2)

- Clients contact proxy; proxy contacts server



CDNs (Content Delivery Networks) (§7.5.3)

- CDNs (Content Delivery Networks)
  - Efficient distribution of popular content; faster delivery for clients



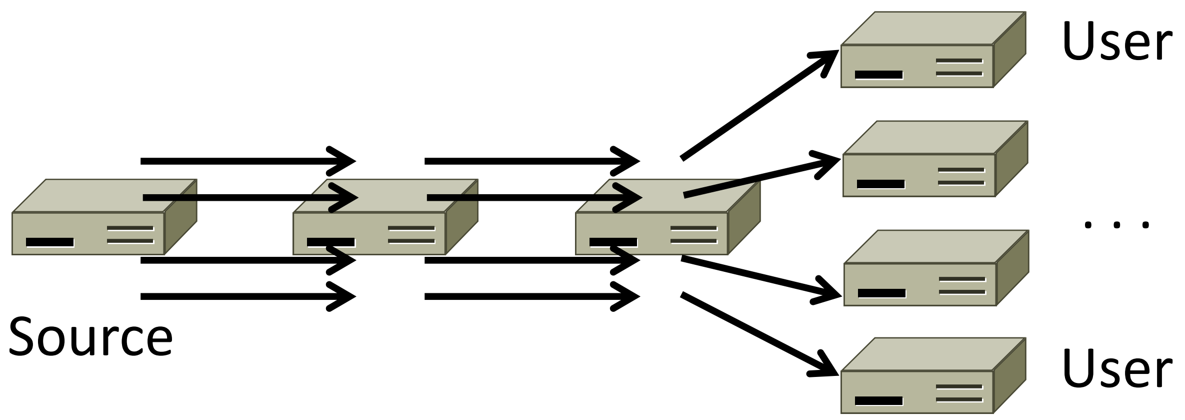
Context

- As the web took off in the 90s, traffic volumes grew and grew. This:
  1. Concentrated load on popular servers
  2. Led to congested networks and need to provision more bandwidth
  3. Gave a poor user experience

- Idea:
  - Place popular content near clients
  - Helps with all three issues above

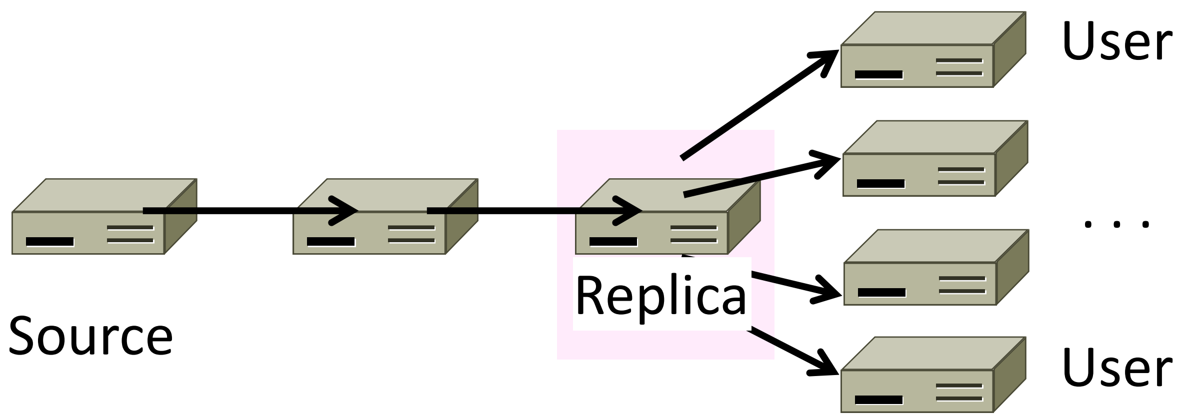
Before CDNs

- Sending content from the source to 4 users takes 4 x 3 = 12 “network hops” in the example



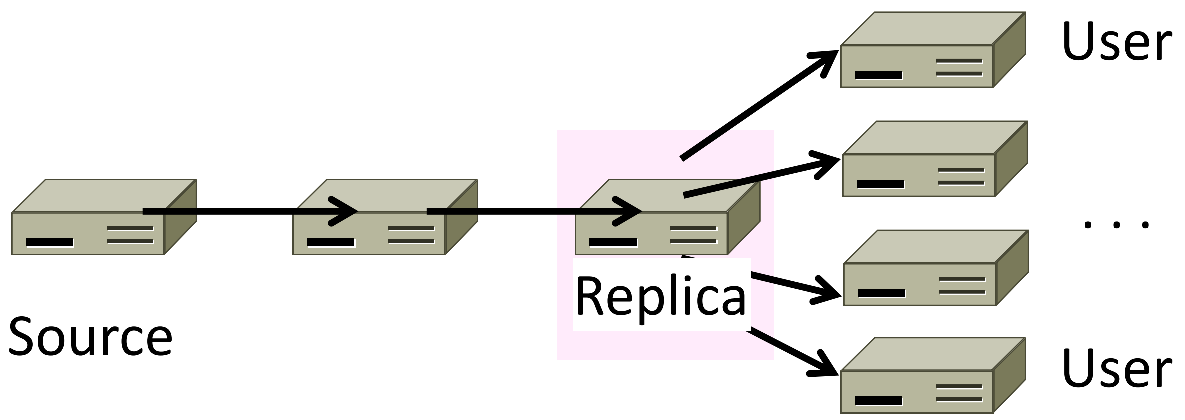
After CDNs

- Sending content via replicas takes only 4 + 2 = 6 "network hops"



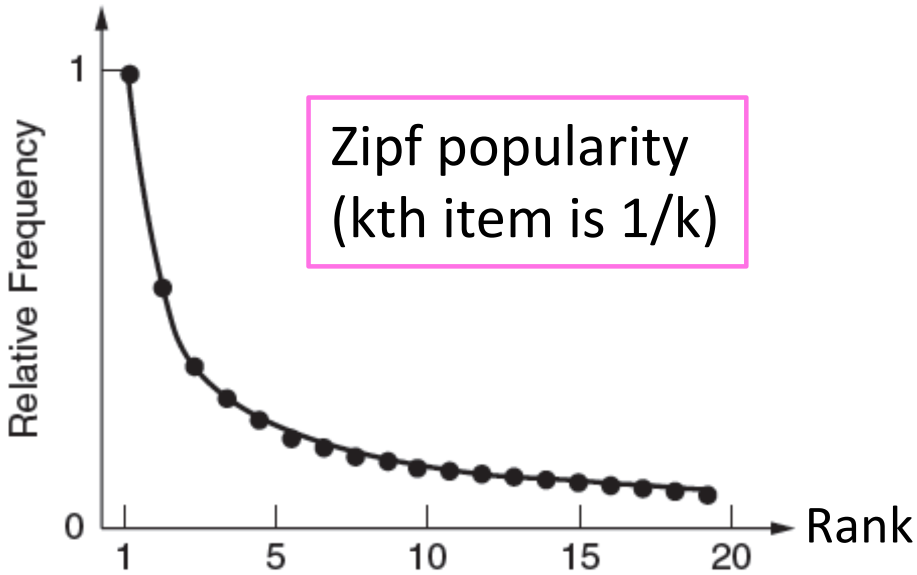
After CDNs (2)

- Benefits assuming popular content:
  - Reduces server, network load
  - Improves user experience (PLT)



Popularity of Content

- Zipf's Law: few popular items, many unpopular ones ("heavy tail" of probability distribution); both matter



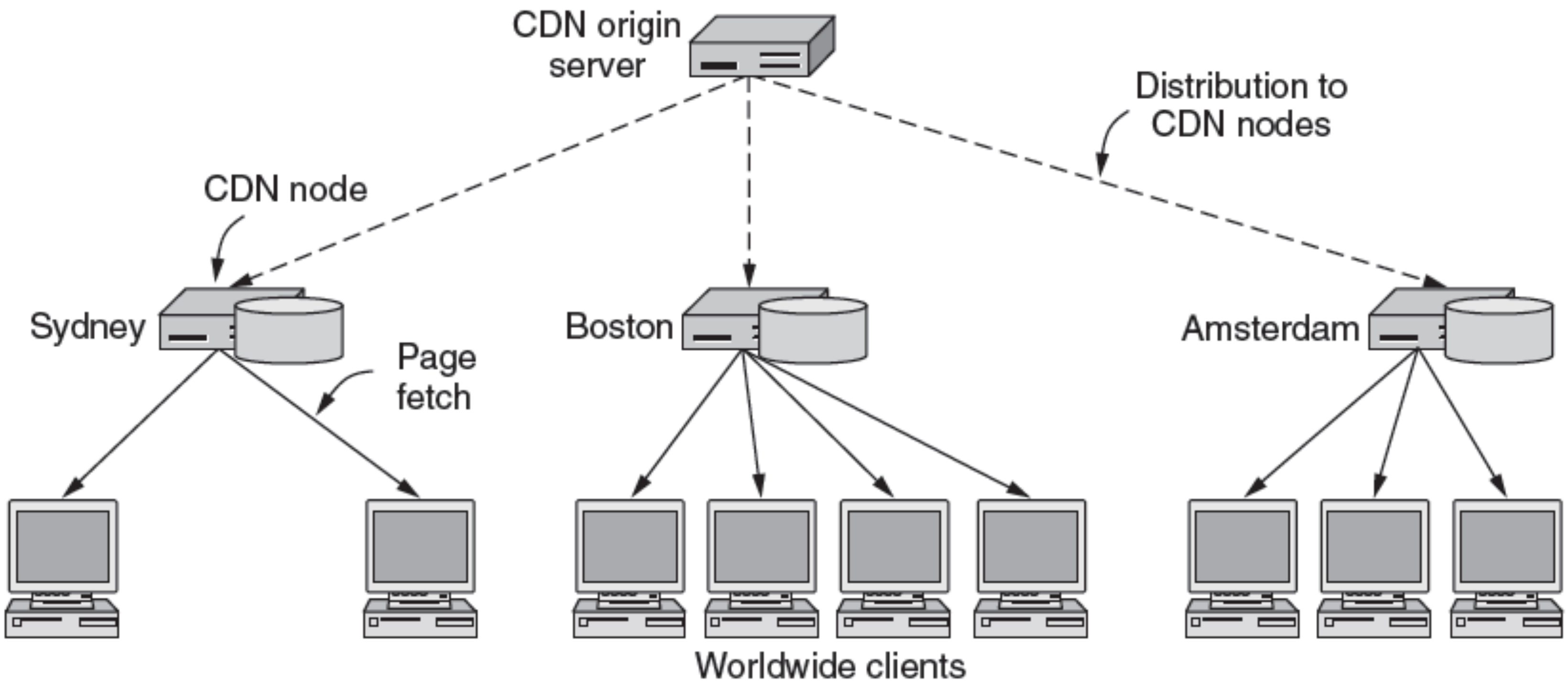
George Zipf (1902-1950)

Source: Wikipedia

How to place content near clients?

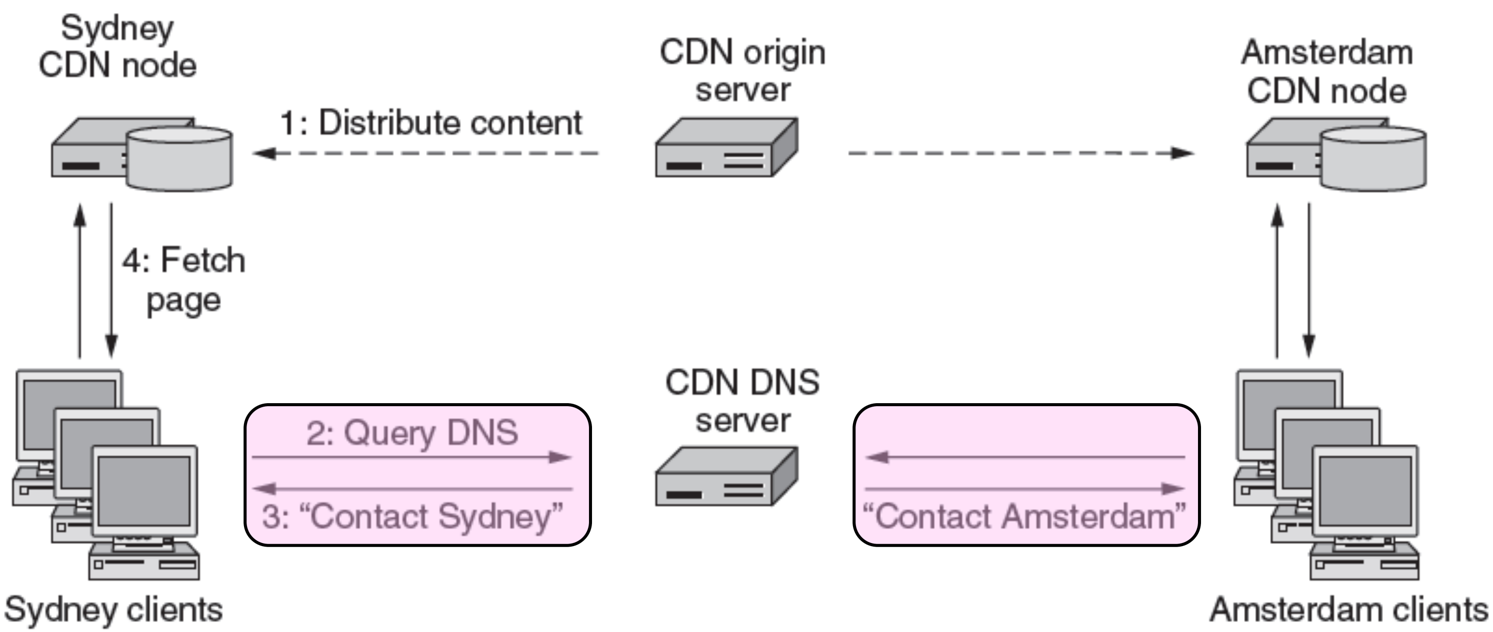
- Use browser and proxy caches
  - Helps, but limited to one client or clients in one organization
- Want to place replicas across the Internet for use by all nearby clients
  - Done by clever use of DNS

Content Delivery Network



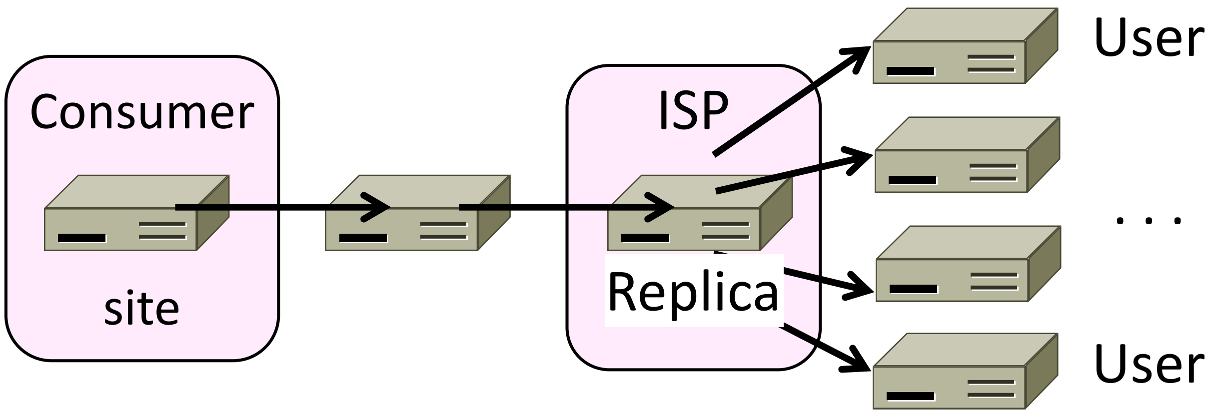
Content Delivery Network (2)

- DNS resolution of site gives answer depending on client
  - Direct each client to the nearest replica (using IP geolocation)



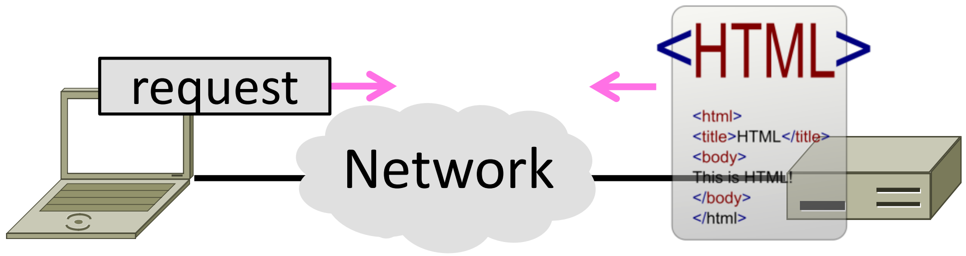
Business Model

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



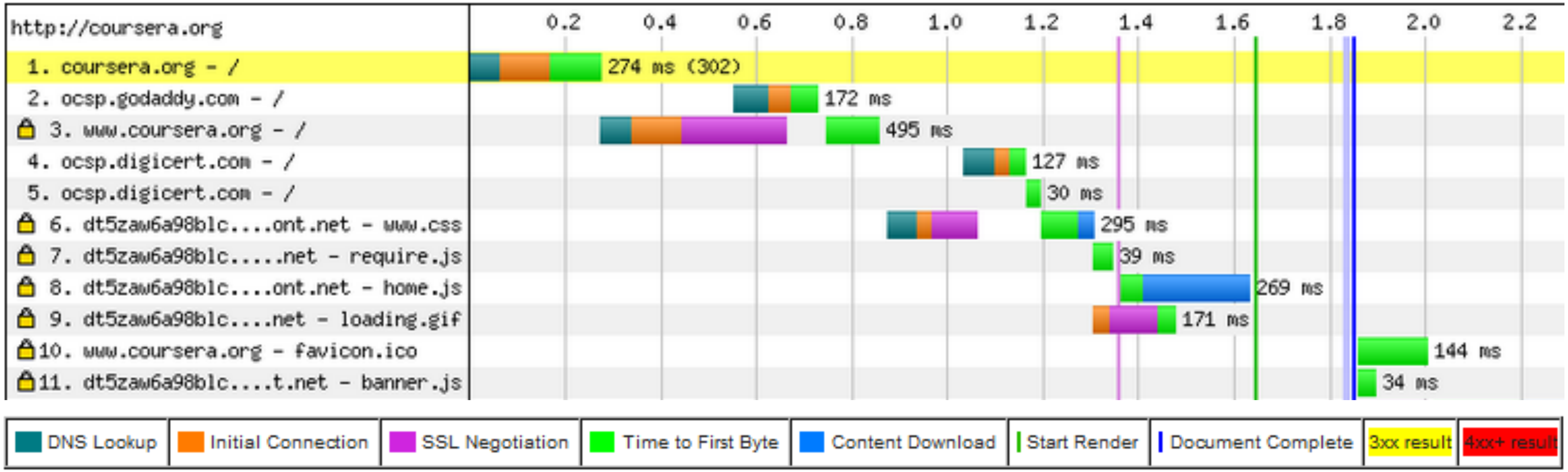
The Future of HTTP

- The Future of HTTP
  - How will we make the web faster?
  - A brief look at some approaches



Modern Web Pages

- Waterfall diagram shows progression of page load

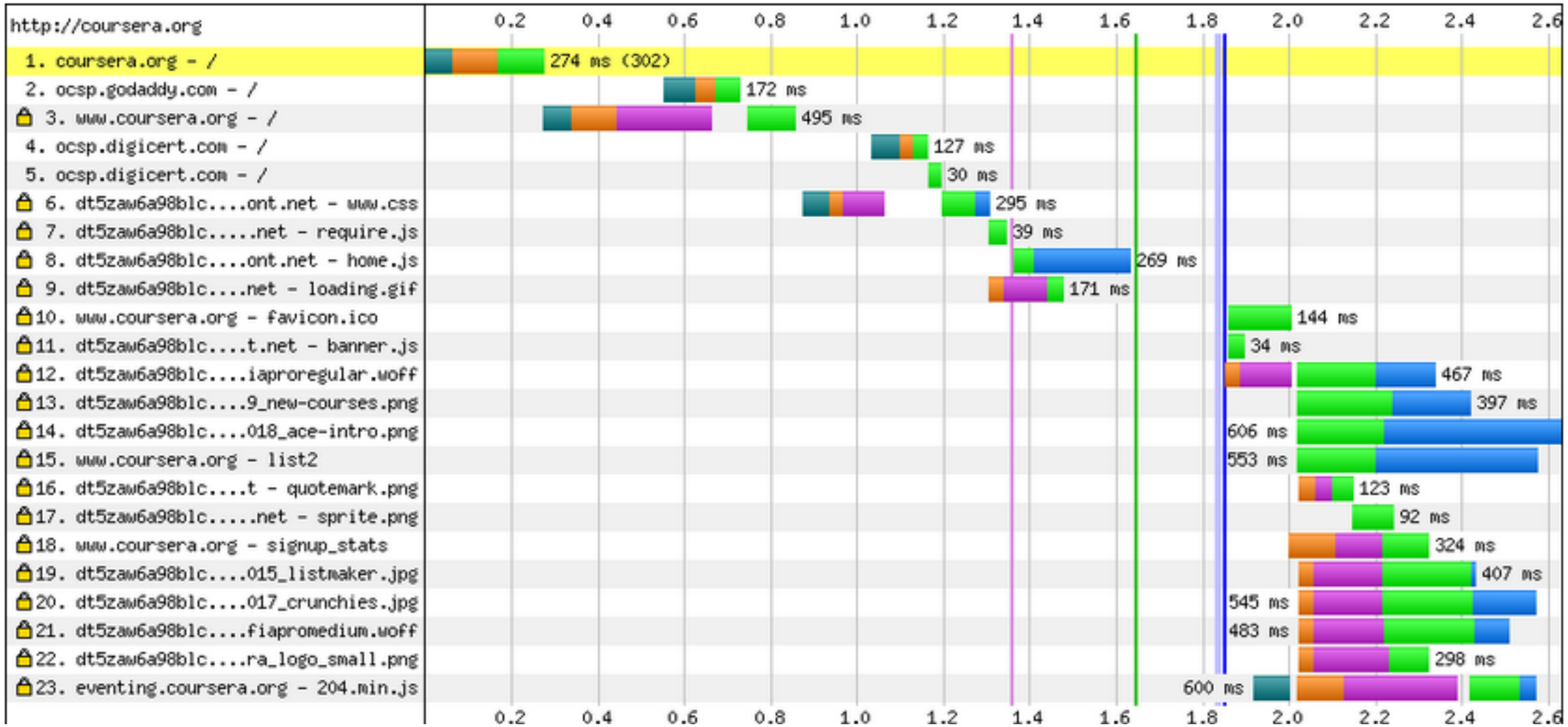


webpagetest tool for http://coursera.org (Firefox, 5/1 Mbps, from VA, 3/1/13)

Modern Web Pages (2)

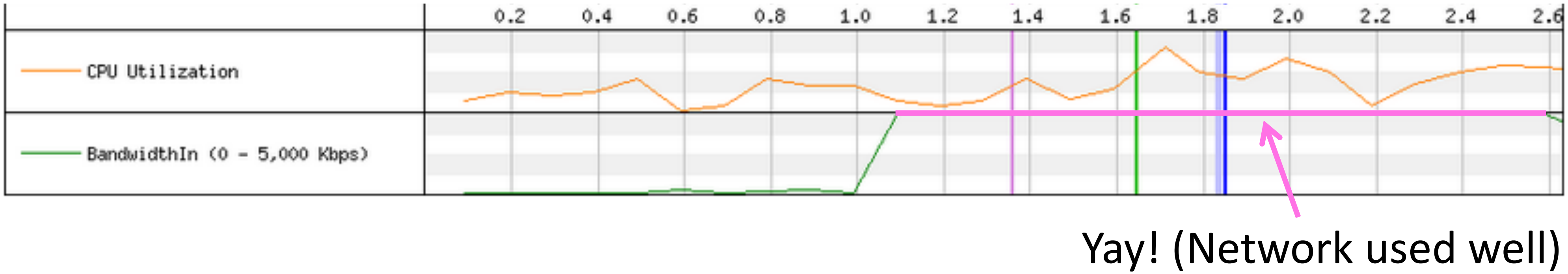
Yikes!

- -23 requests
- -1 Mb data
- -2.6 secs



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

Modern Web Pages (3)



- 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

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 HTTP/2

mod_pagespeed

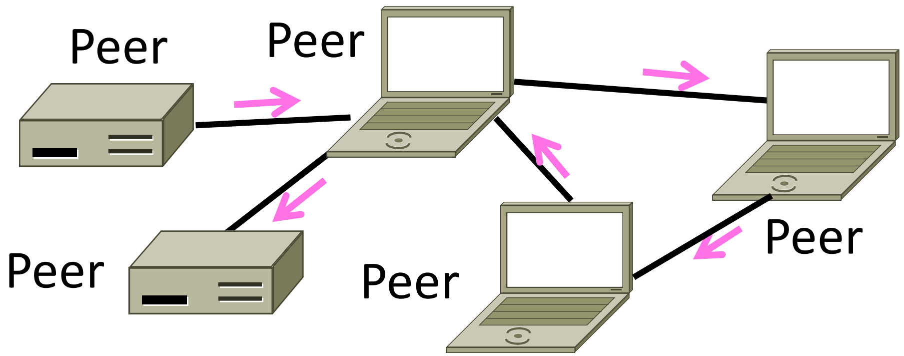
- Observation:
  - The way pages are written affects how quickly they load
  - Many books on best practices for page authors and developers
- Key idea:
  - Have server re-write (compile) pages to help them load quickly!
  - Apache mod_pagespeed is an example

mod_pagespeed (2)

- 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)

Peer-to-Peer Content Delivery (BitTorrent) (§7.5.4)

- Peer-to-peer content delivery
  - 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

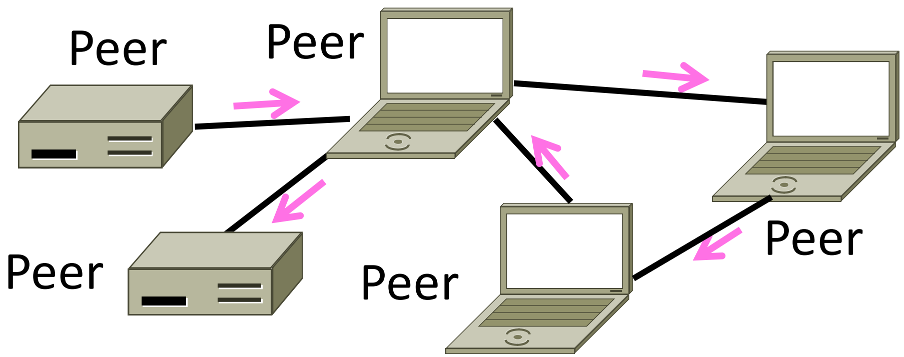
P2P (Peer-to-Peer)

- Goal is delivery *without* dedicated infrastructure or centralized control
  - Still efficient at scale, and reliable

- Key idea is to have participants (or peers) help each other
  - Initially Napster '99 for music (gone)
  - Now BitTorrent '01 onwards (popular!)

P2P Challenges

- No servers on which to rely on
  - Communication must be <u>peer-to-peer</u> and self-organizing, not client-server
  - Leads to several issues at scale …

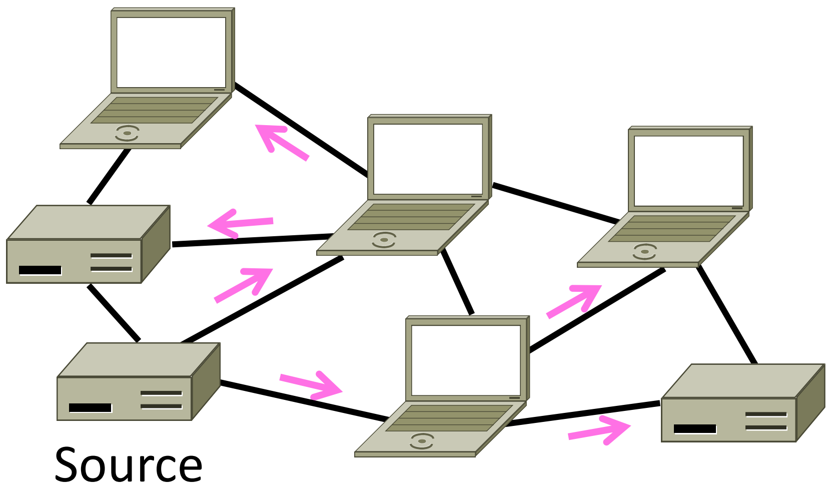


P2P Challenges (2)

1. Limited capabilities
   – How can one peer deliver content to all other peers?
2. Participation incentives
   – Why would peers help each other?
3. Decentralization
   – How will peers find content?

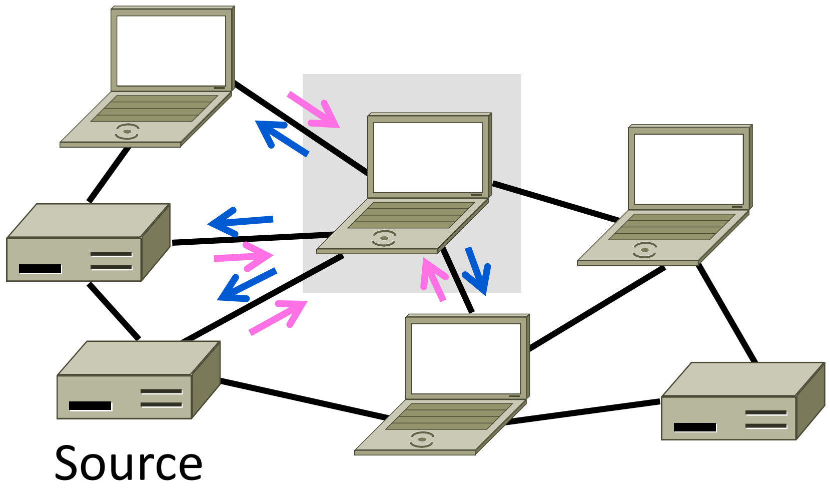
Overcoming Limited Capabilities

- Peer can send content to all other peers using a distribution tree
  - Typically done with replicas over time
  - Self-scaling capacity (more nodes → more capacity)



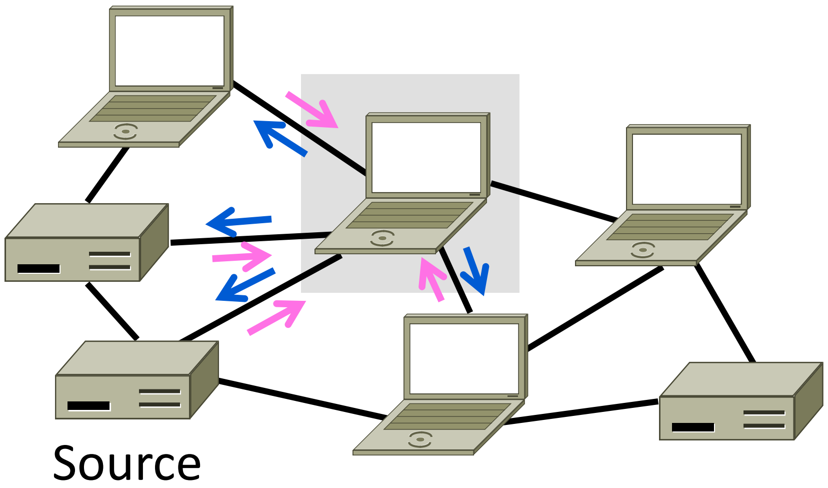
Providing Participation Incentives

- Peers play two roles:
  - Download (→) to help themselves, and upload (←) to help others



Providing Participation Incentives (2)

- 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

BitTorrent

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

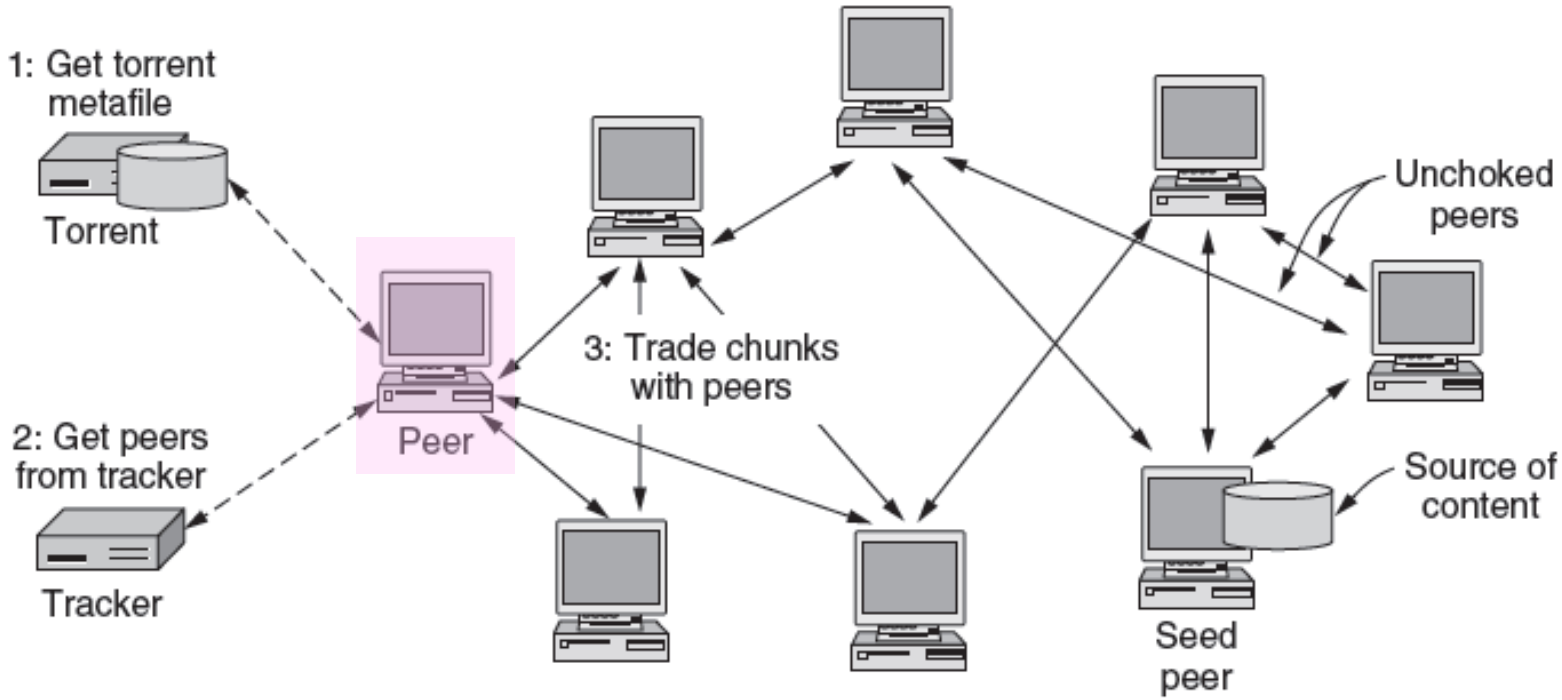
Bram Cohen (1975—)

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)
  2. Or, use DHT index for peers
  3. Trade pieces with different peers
  4. Favor peers that upload to you rapidly; “choke” peers that don’t by slowing your upload to them

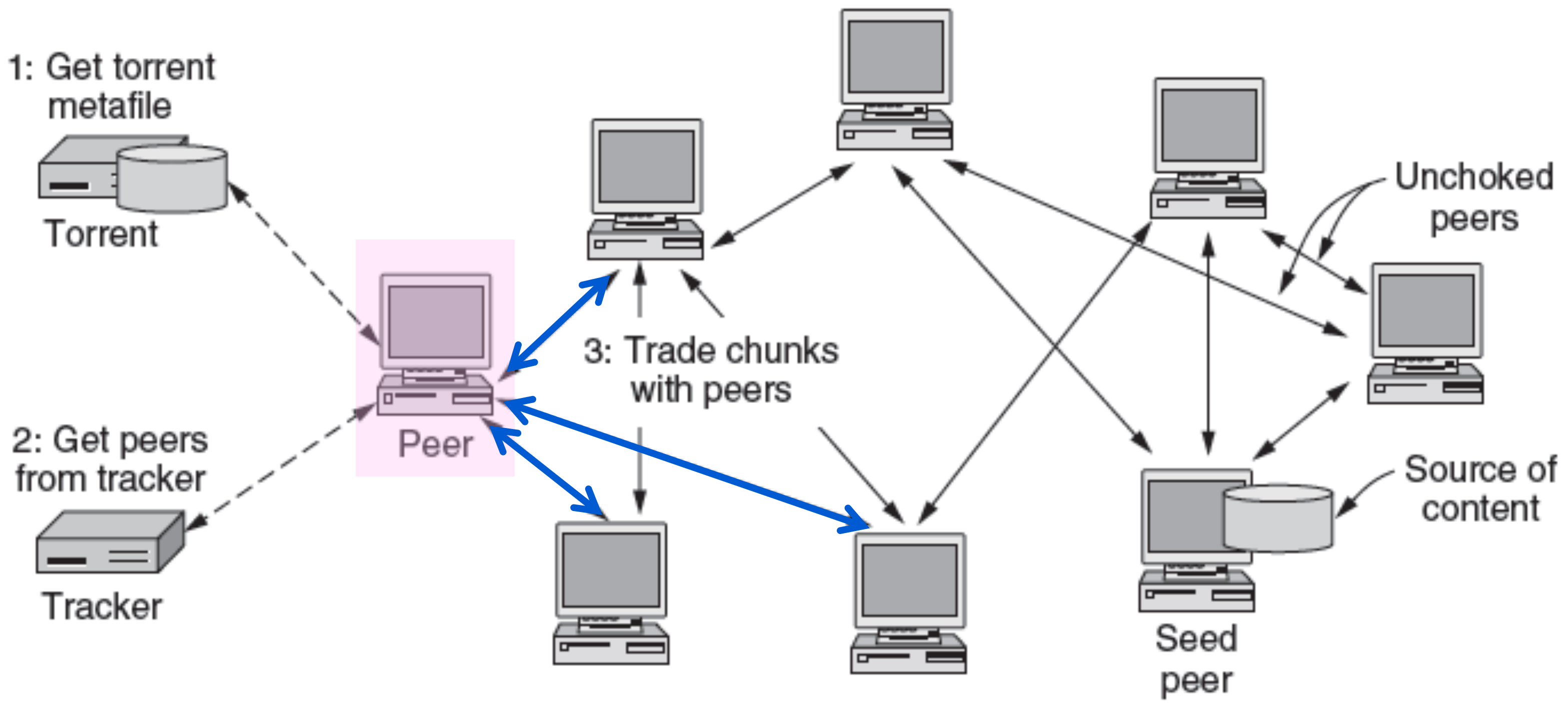
BitTorrent Protocol (2)

- All peers (except seed) retrieve torrent at the same time



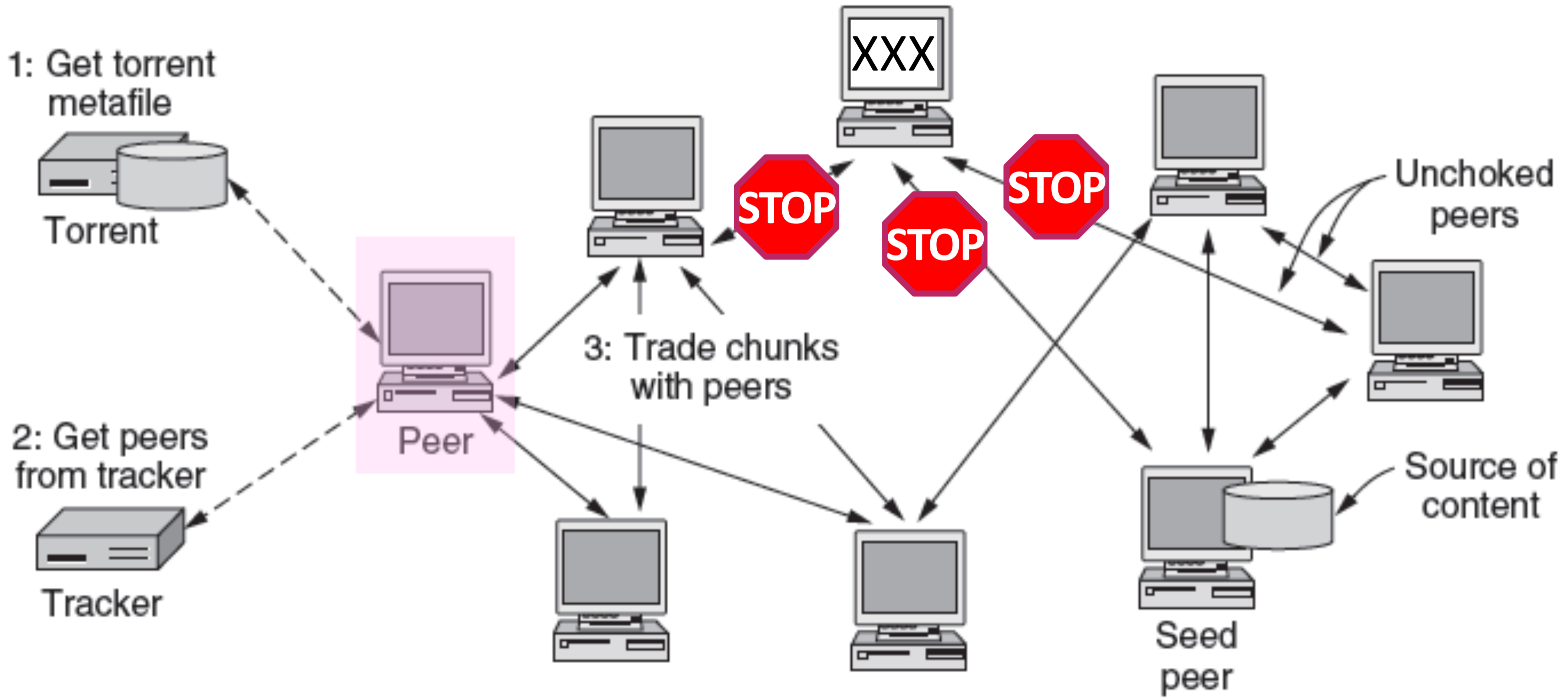
BitTorrent Protocol (3)

- Dividing file into pieces gives parallelism for speed



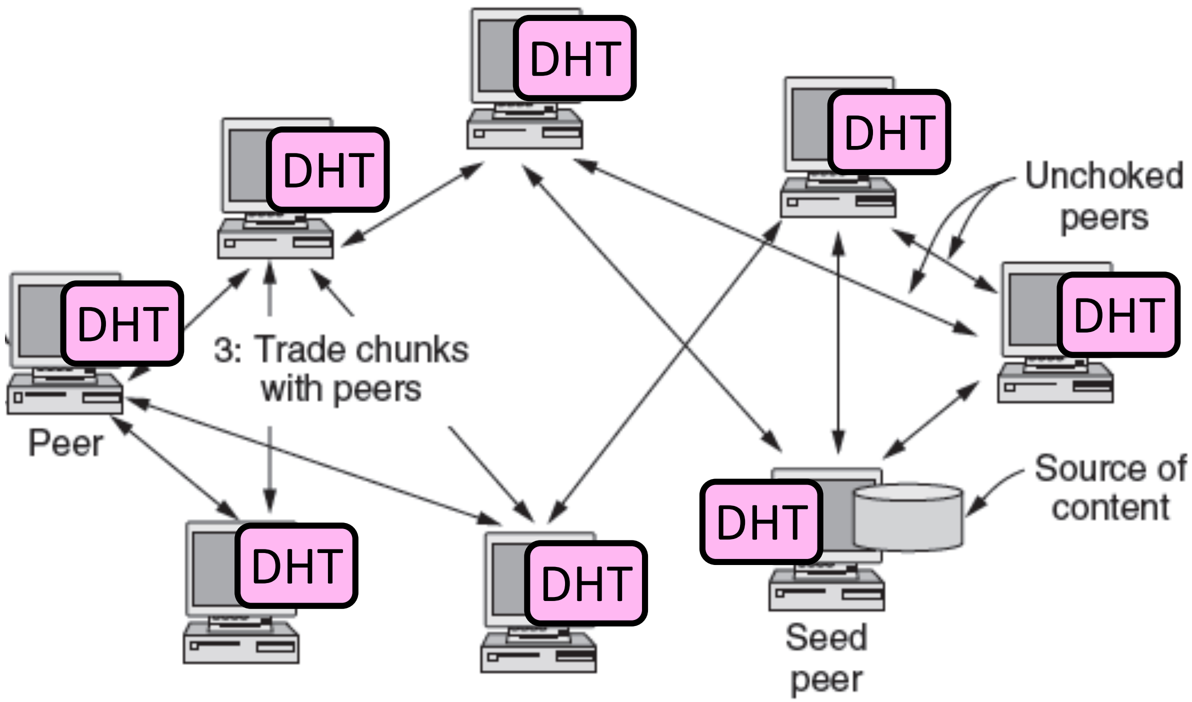
BitTorrent Protocol (4)

- Choking unhelpful peers encourages participation



BitTorrent Protocol (5)

- 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 cloud computing
  - Expect hybrid systems in the future