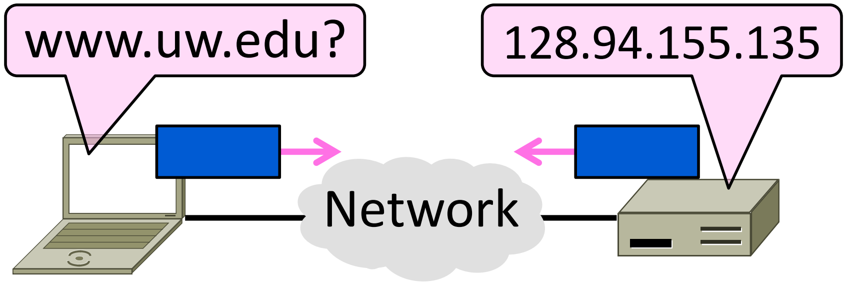
Computer Networks

Application Layer Protocols

Topic

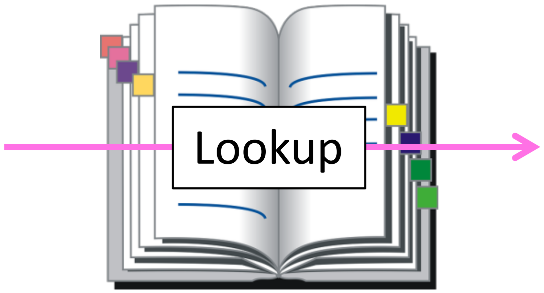
- The DNS (Domain Name System)
  - Human-readable host names, and more
  - Part 1: the distributed namespace



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

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



Lookup

Directory

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

Before the DNS – HOSTS.TXT

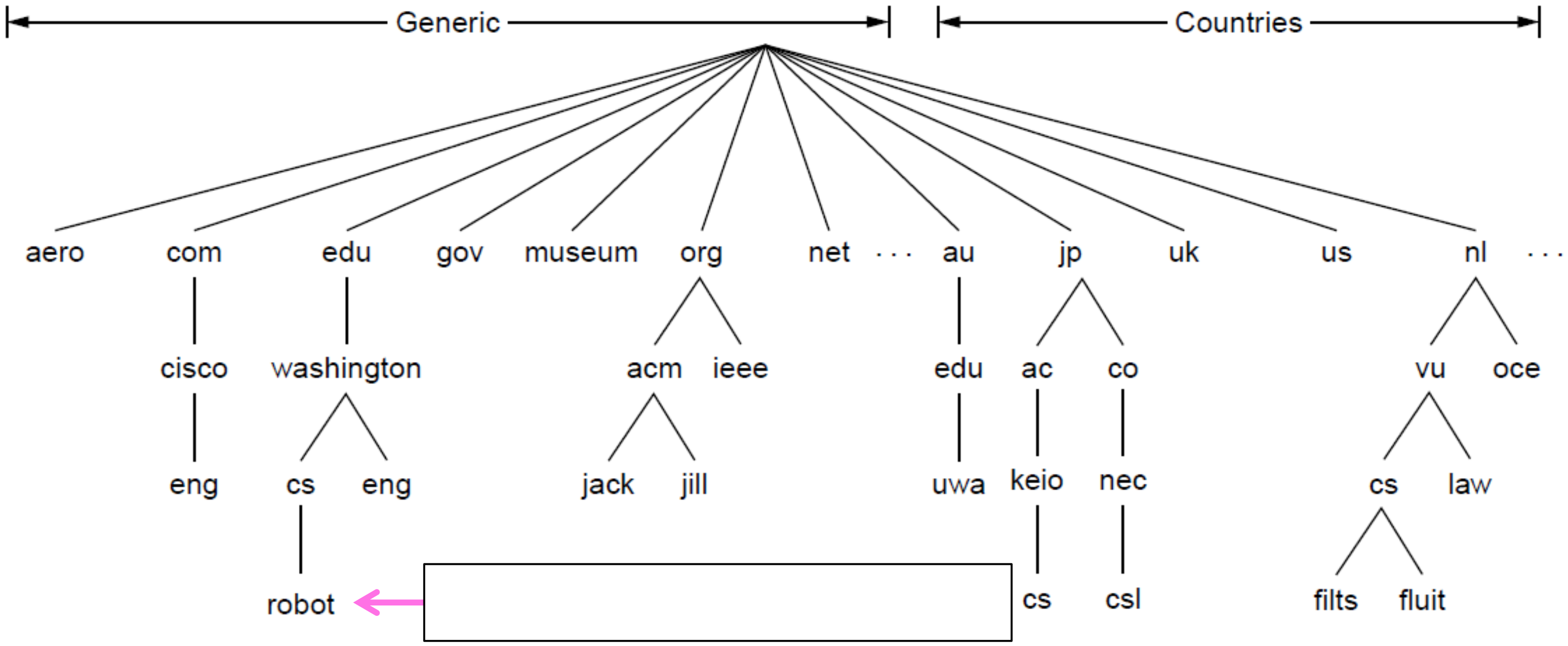
- Directory was a file HOSTS.TXT regularly retrieved for all hosts from a central machine at the NIC (Network Information Center)
- Names were initially flat, became hierarchical (e.g., lcs.mit.edu) ~85
- Neither manageable nor efficient as the ARPANET grew …

DNS

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

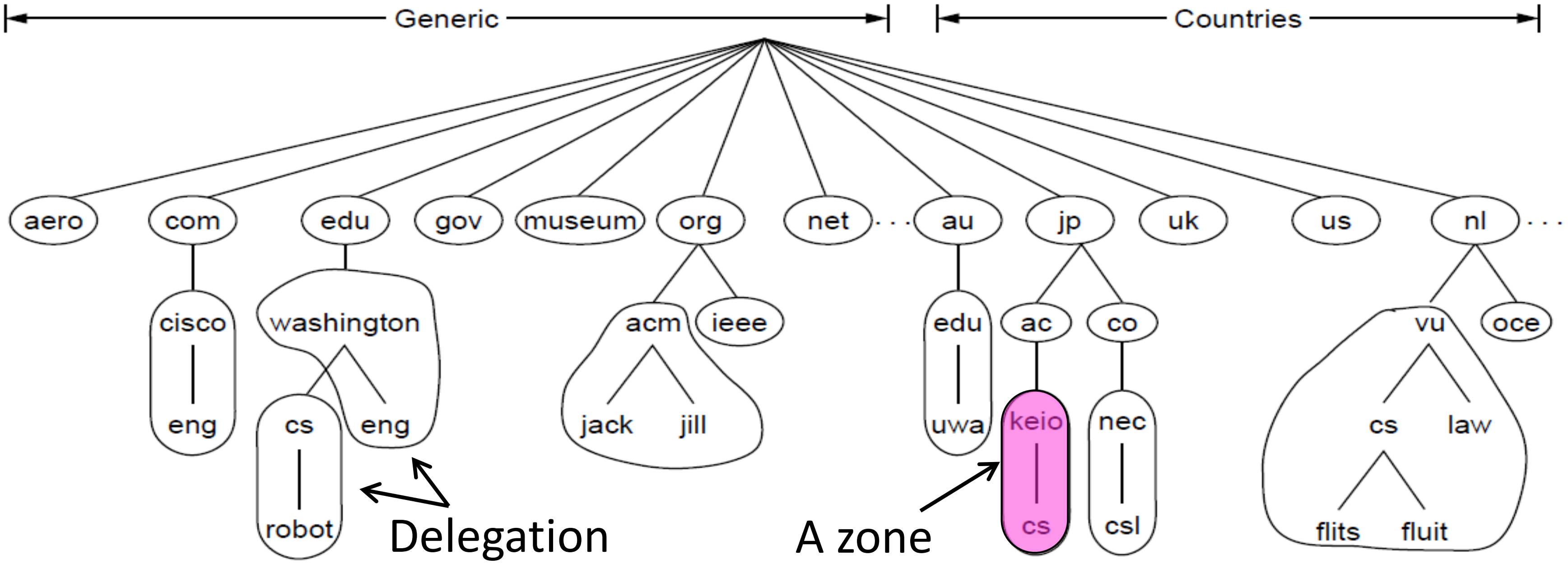


TLDs (Top-Level Domains)

- 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



DNS Zones (2)

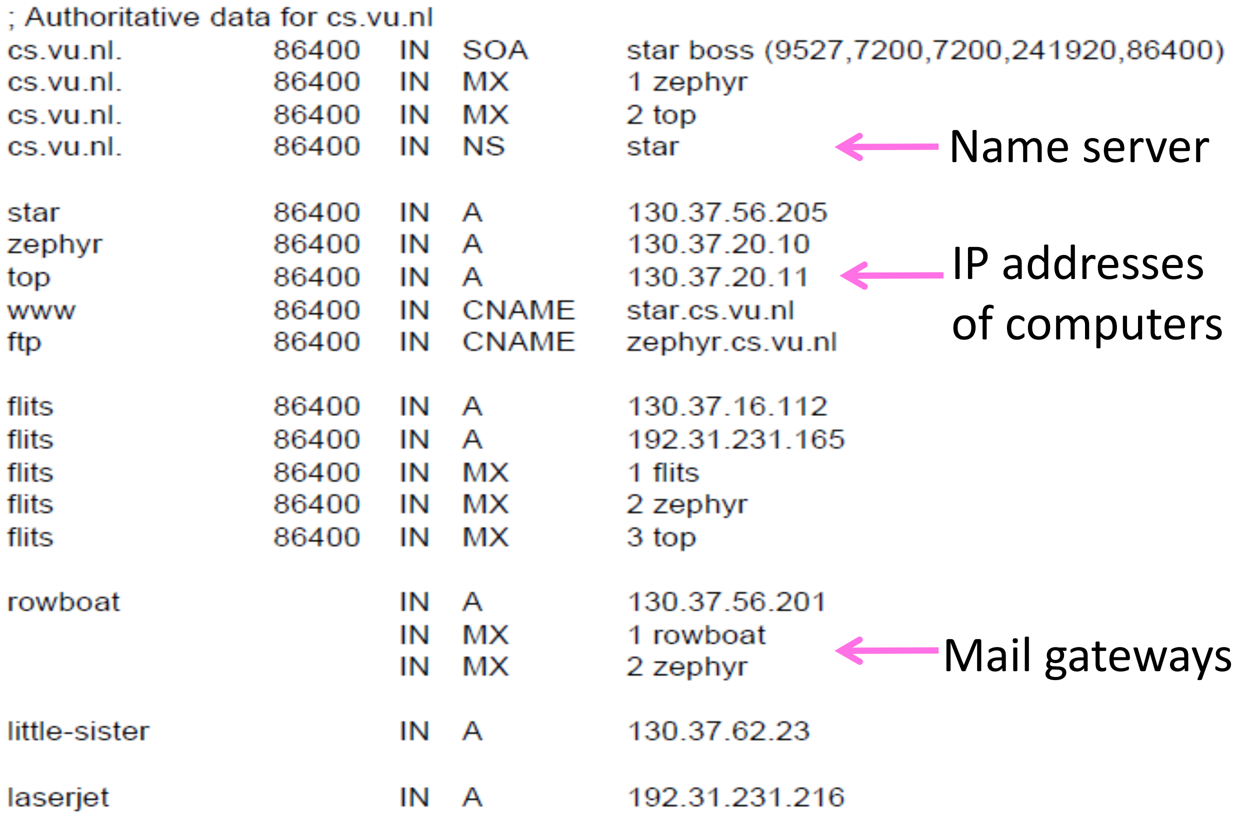
- Zones are the basis for distribution
  - EDU Registrar administers .edu
  - UW administers washington.edu
  - CS&E administers cs.washington.edu
- Each zone has a <u>nameserver</u> to contact for information about it
  - Zone must include contacts for delegations, e.g., .edu knows nameserver for washington.edu

DNS Resource Records

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

Type	Meaning
SOA	Start of authority, has key 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

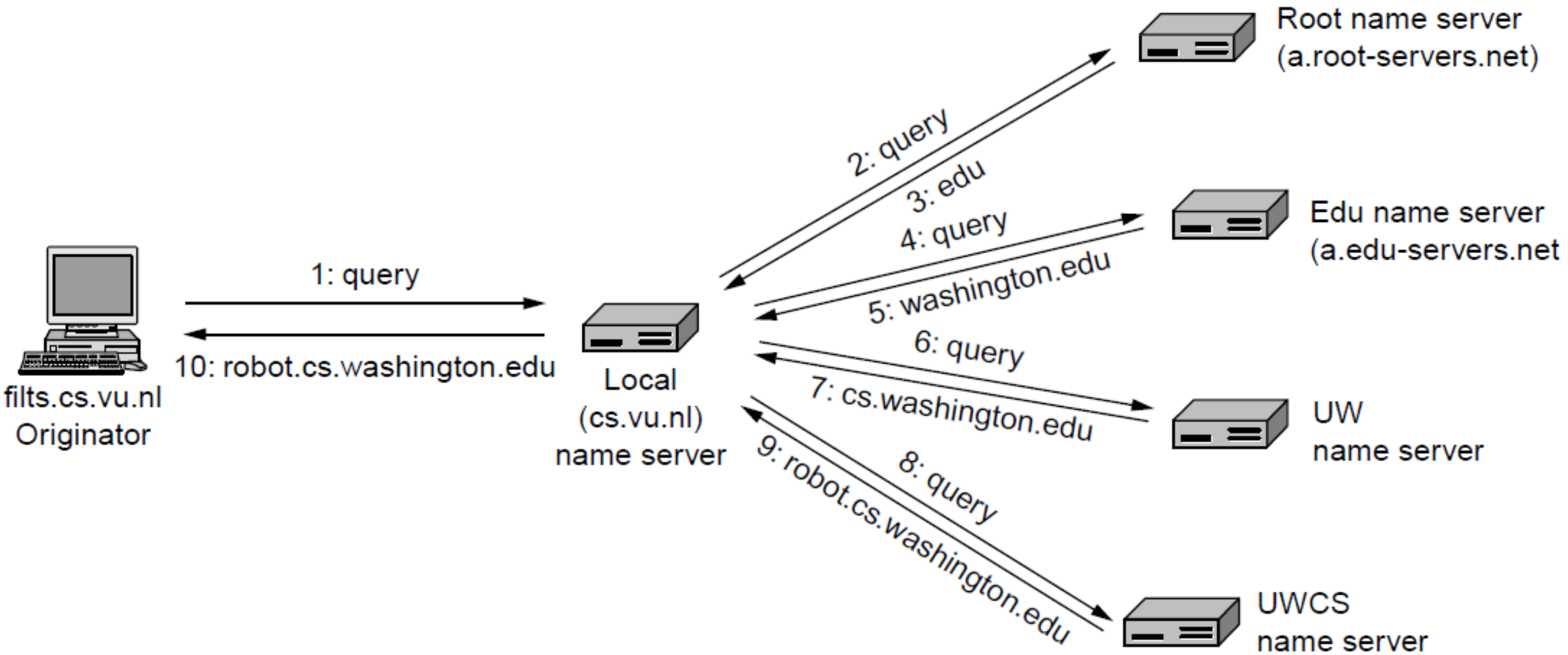
Mail gateways

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

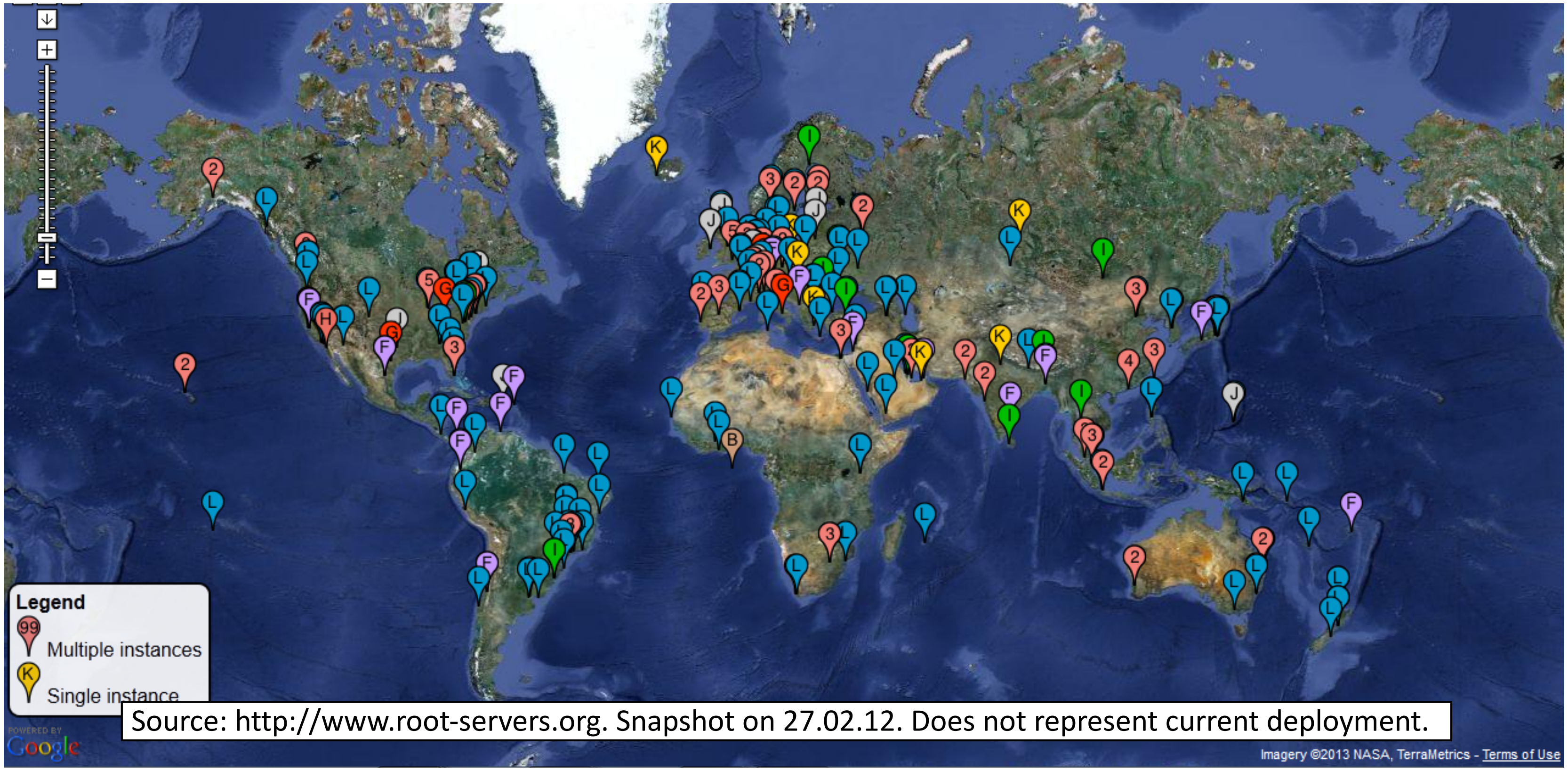
Question

- What are the performance and security implications of the DNS scheme?

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

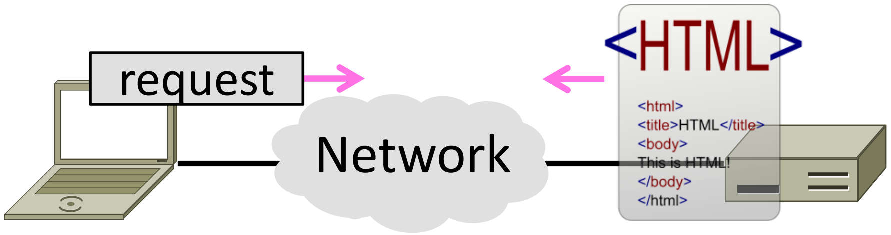
Root Server Deployment



Source: http://www.root-servers.org. Snapshot on 27.02.12. Does not represent current deployment.

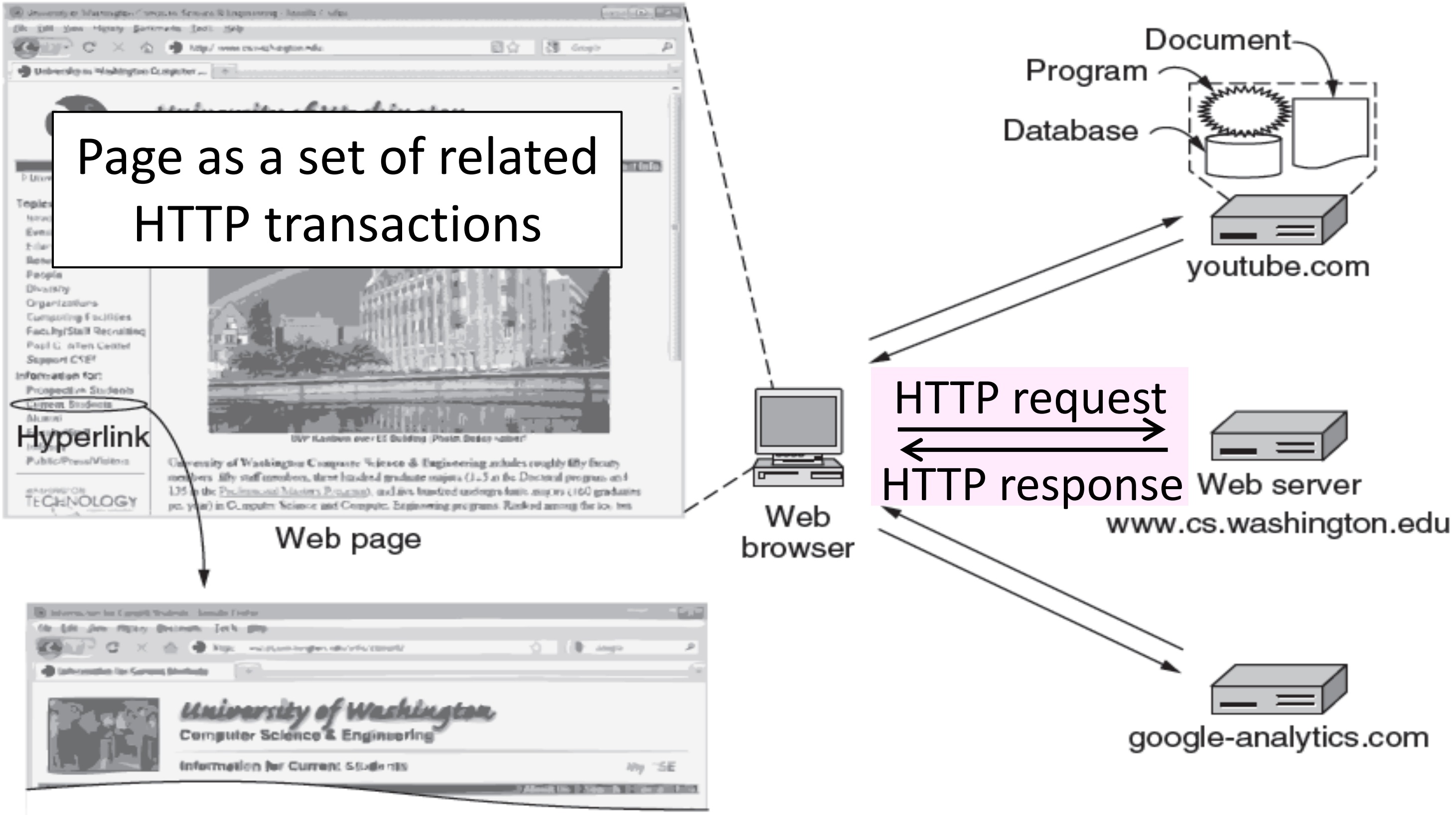
Topic

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



```
<html>
<title>HTML</title>
<body>
This is HTML!
</body>
</html>
```

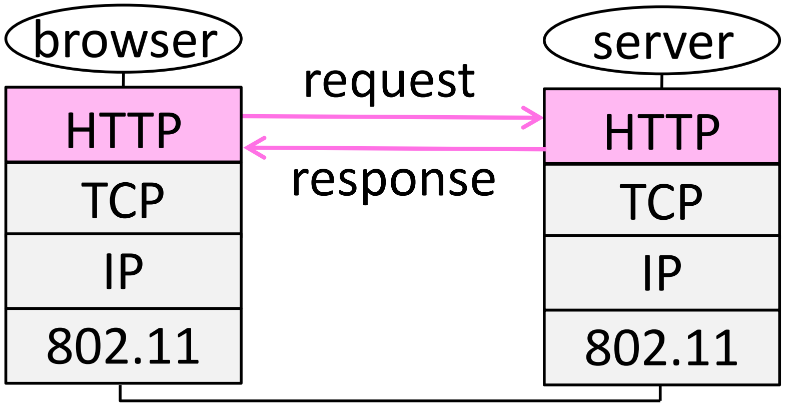
Web Context



Page as a set of related HTTP transactions

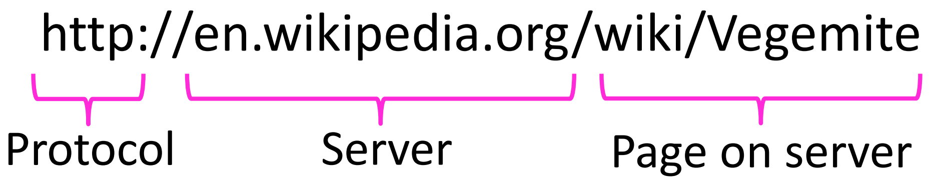
Web Protocol Context

- 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://en.wikipedia.org/wiki/Vegemite

Protocol

Server

Page on server

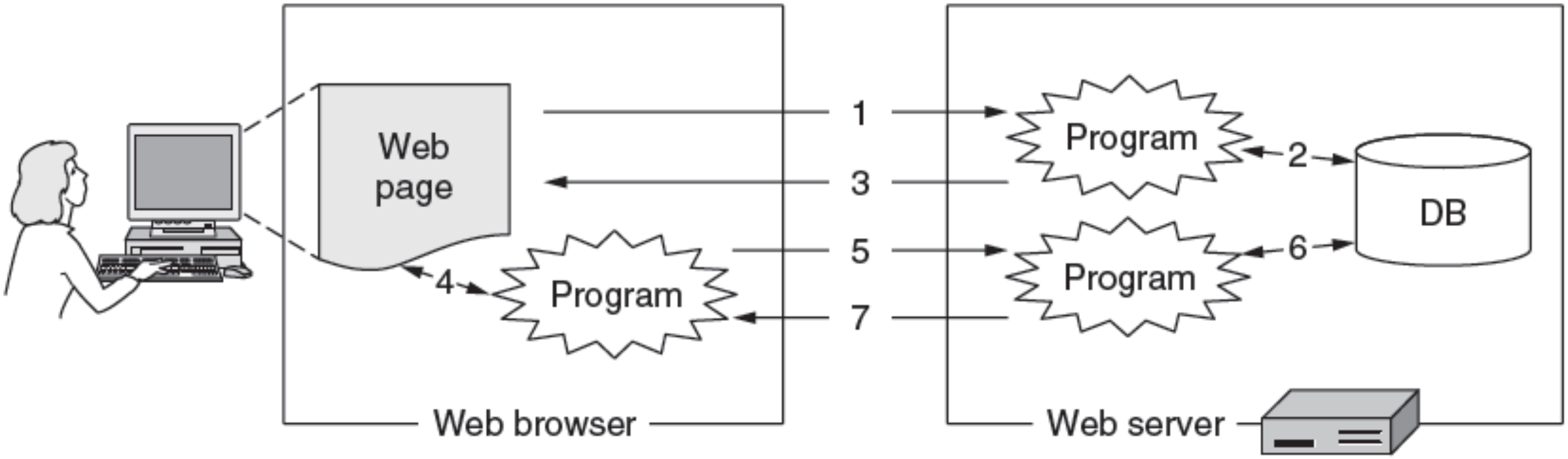
- Steps:
  - Resolve the server to IP address (DNS)
  - Set up TCP connection to the server
  - Send HTTP request for the page
  - (Await HTTP response for the page)

  ** Execute / fetch other Web 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



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)

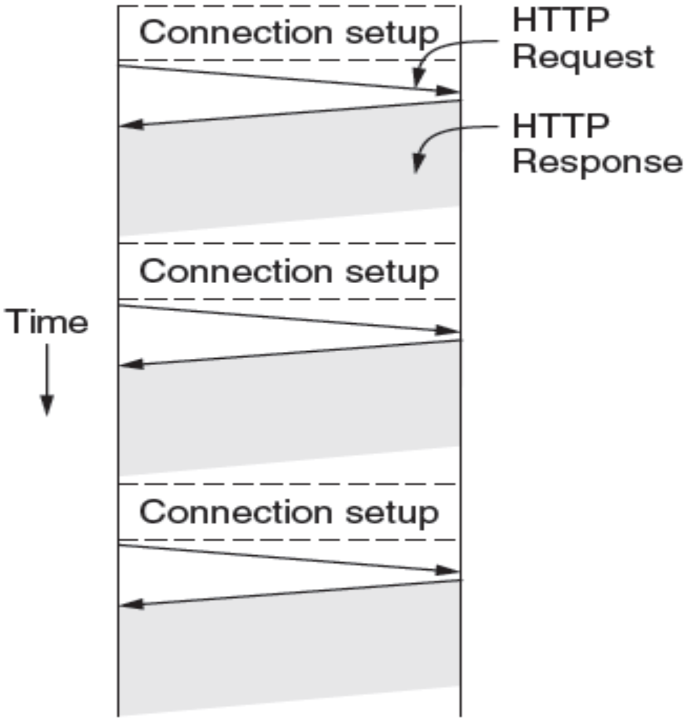
PLT (Page Load Time)

- 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

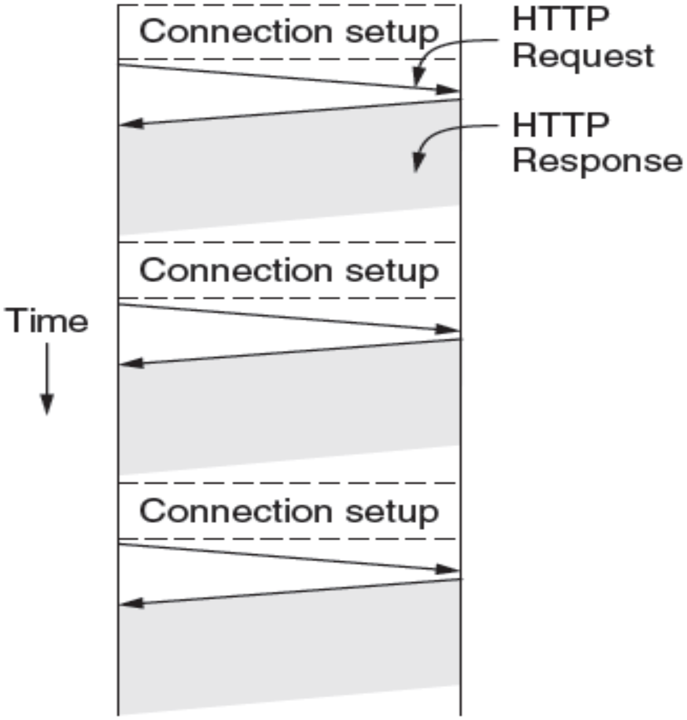
Early Performance

- HTTP/1.0 used one TCP connection to fetch one 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



- What performance optimizations were introduced by newer HTTP versions? Which ones are reliably used?

Parallel Connections

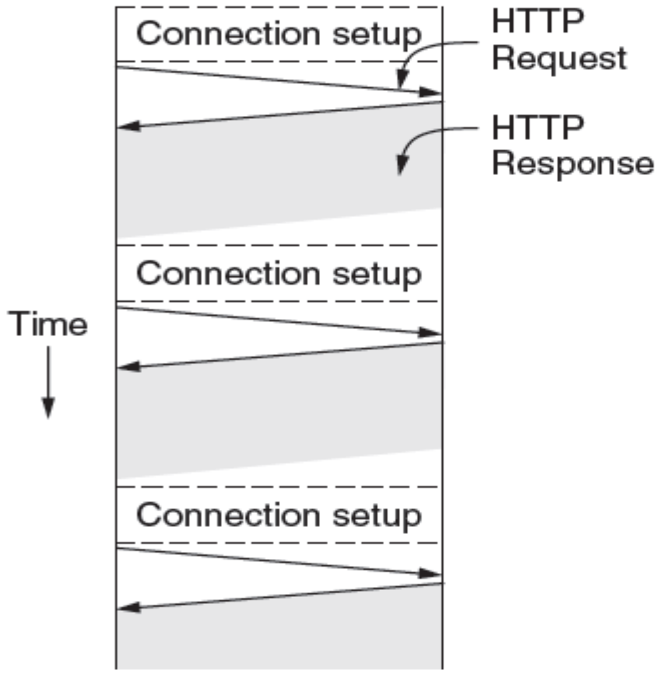
- One simple way to reduce PLT
  - Browser runs multiple (8, say) HTTP instances in parallel
  - Server is unchanged; already handled concurrent requests for many clients
- How does this help?
  - Single HTTP wasn’t using network much …
  - So parallel connections aren’t slowed much
  - Pulls in completion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

Persistent Connections

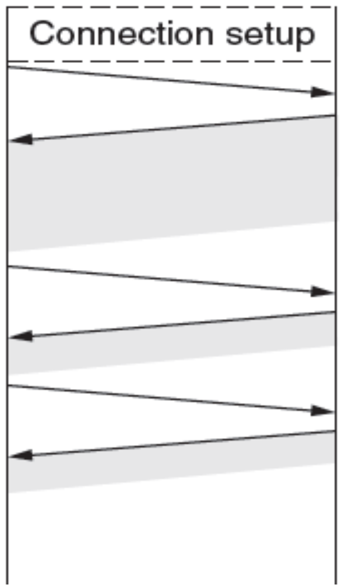
- Parallel connections compete with each other for network resources
  - 1 parallel client ≈ 8 sequential clients?
  - Exacerbates network bursts, and loss

- Persistent connection alternative
  - Make 1 TCP connection to 1 server
  - Use it for multiple HTTP requests

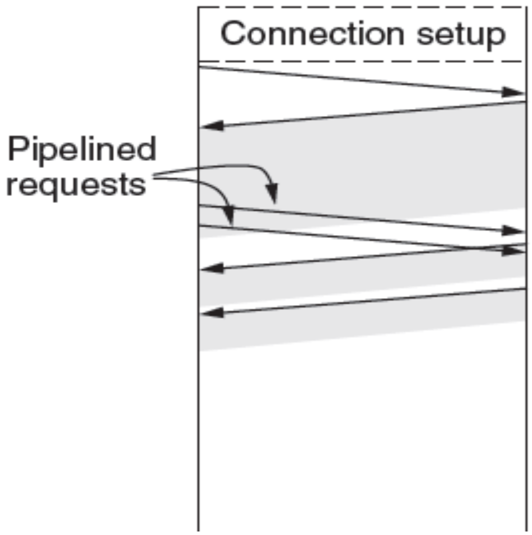
Persistent Connections (2)



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

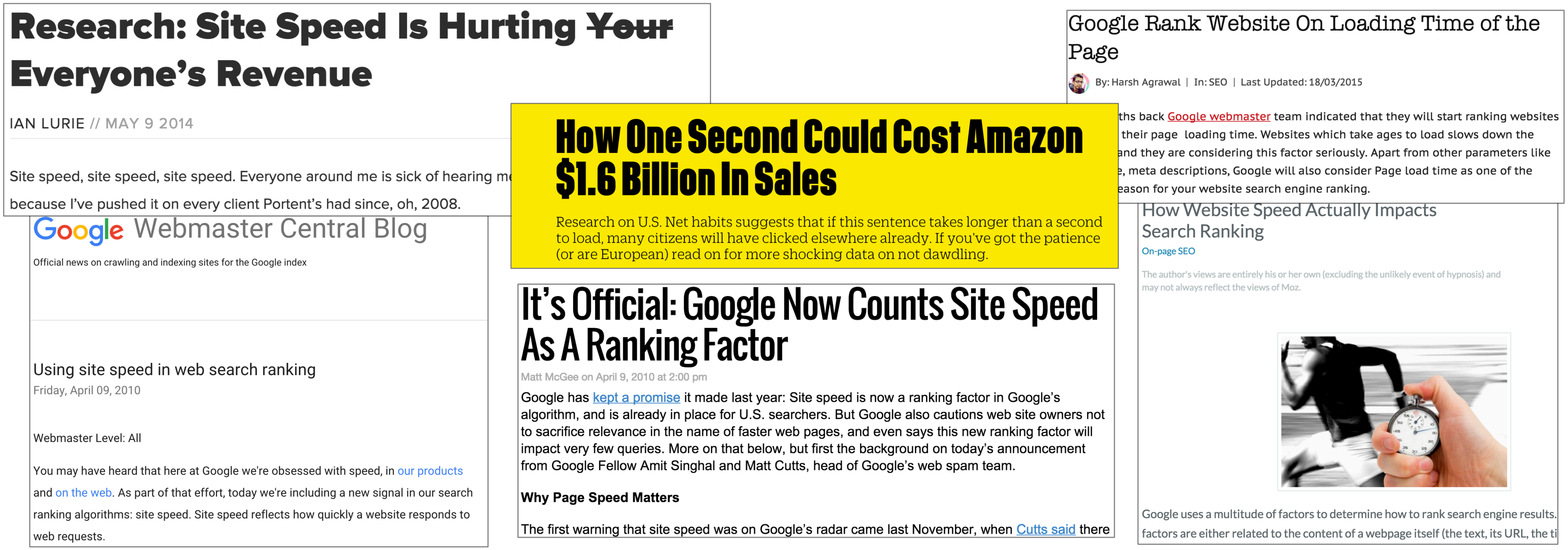
- Issues with persistent connections
  - How long to keep TCP connection?

Polaris: Faster Page Loads Using Finegrained Dependency Tracking

Slides courtesy of Ravi Netravali

Web Performance

- Users demand fast page loads
- Slow page loads lead to lost revenue and low search rank



Research: Site Speed Is Hurting ~~Your~~ Everyone's Revenue

IAN LURIE // MAY 9 2014

Site speed, site speed, site speed. Everyone around me is sick of hearing m
because I've pushed it on every client Portent's had since, oh, 2008.

Google Webmaster Central Blog

Official news on crawling and indexing sites for the Google index

Using site speed in web search ranking

Friday, April 09, 2010

Webmaster Level: All

You may have heard that here at Google we're obsessed with speed, in our products and on the web. As part of that effort, today we're including a new signal in our search ranking algorithms: site speed. Site speed reflects how quickly a website responds to web requests.

How One Second Could Cost Amazon $1.6 Billion In Sales

Research on U.S. Net habits suggests that if this sentence takes longer than a second to load, many citizens will have clicked elsewhere already. If you've got the patience (or are European) read on for more shocking data on not dawdling.

It's Official: Google Now Counts Site Speed As A Ranking Factor

Matt McGee on April 9, 2010 at 2:00 pm

Google has kept a promise it made last year: Site speed is now a ranking factor in Google's algorithm, and is already in place for U.S. searchers. But Google also cautions web site owners not to sacrifice relevance in the name of faster web pages, and even says this new ranking factor will impact very few queries. More on that below, but first the background on today's announcement from Google Fellow Amit Singhal and Matt Cutts, head of Google's web spam team.

Why Page Speed Matters

The first warning that site speed was on Google's radar came last November, when Cutts said there

Google Rank Website On Loading Time of the Page

By: Harsh Agrawal | In: SEO | Last Updated: 18/03/2015

ths back Google webmaster team indicated that they will start ranking websites
their page loading time. Websites which take ages to load slows down the
and they are considering this factor seriously. Apart from other parameters like
e, meta descriptions, Google will also consider Page load time as one of the
eason for your website search engine ranking.

How Website Speed Actually Impacts Search Ranking

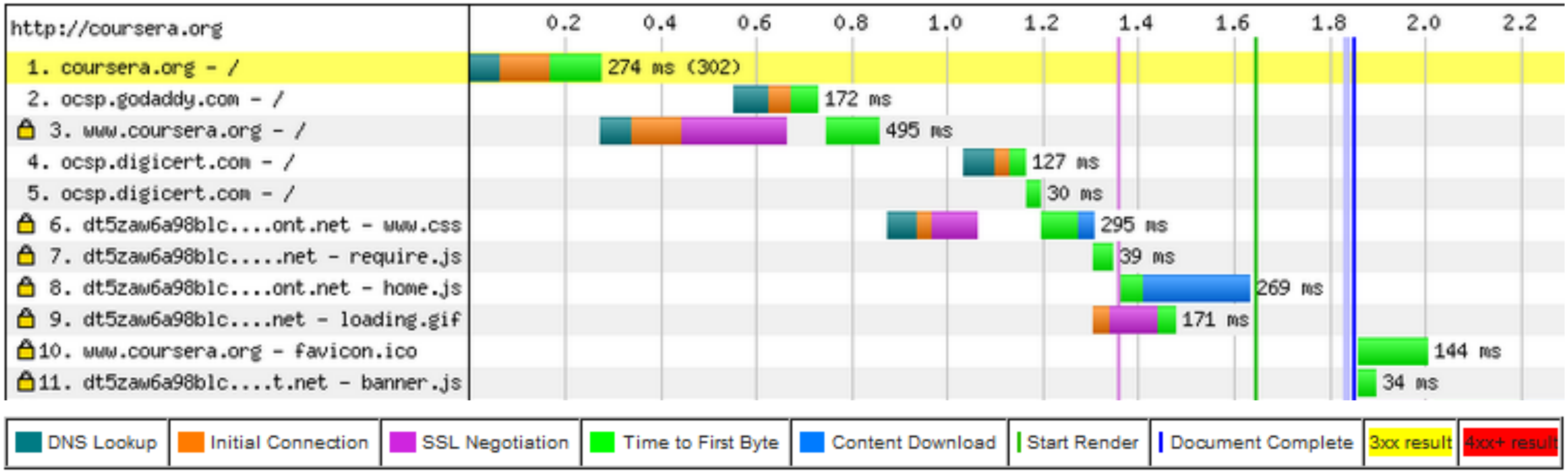
On-page SEO

The author's views are entirely his or her own (excluding the unlikely event of hypnosis) and may not always reflect the views of Moz.

Google uses a multitude of factors to determine how to rank search engine results.
factors are either related to the content of a webpage itself (the text, its URL, the ti

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)

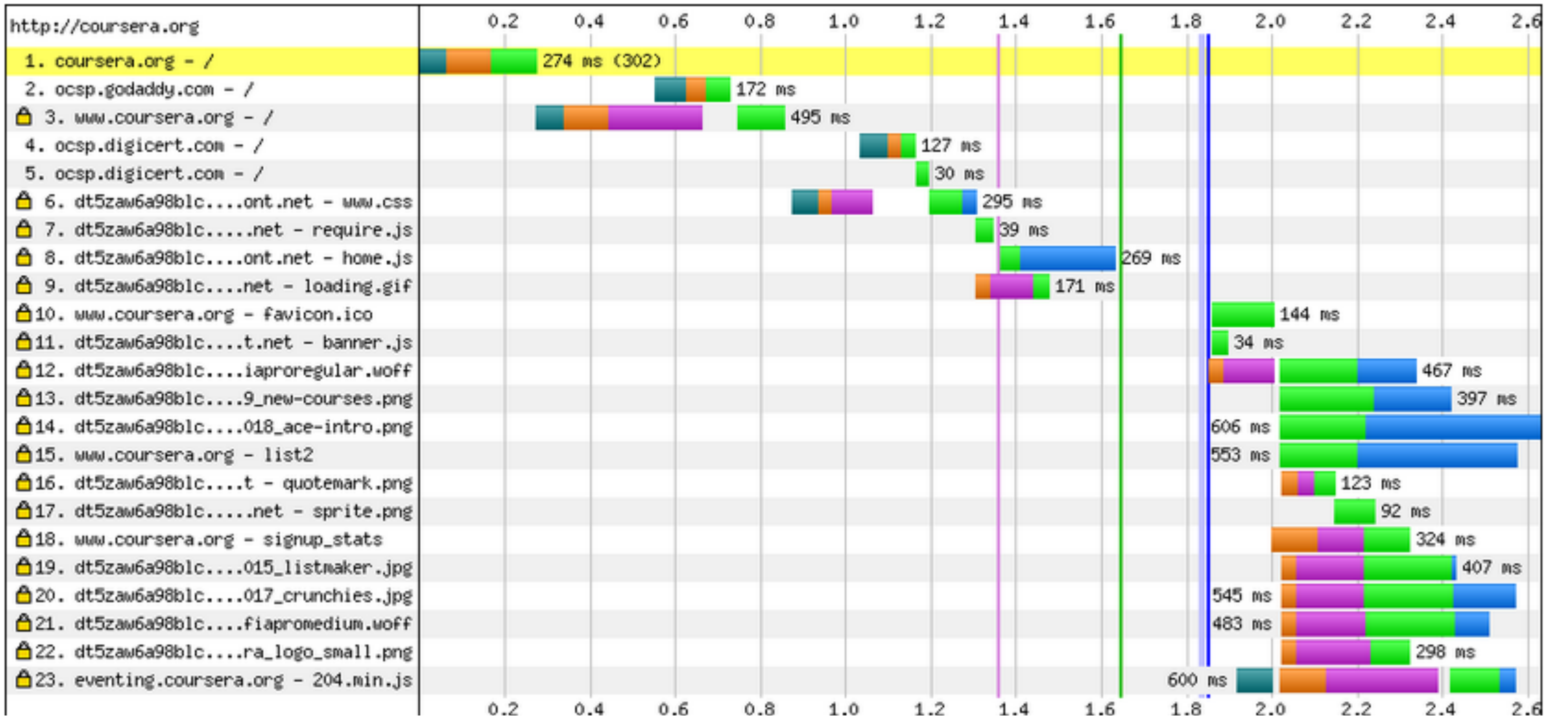
Modern Web Pages (2)

Yikes!

-23 requests

-1 Mb data

-2.6 secs



http://coursera.org

1. coursera.org - /	274 ms (302)
2. ocsp.godaddy.com - /	172 ms
3. www.coursera.org - /	495 ms
4. ocsp.digicert.com - /	127 ms
5. ocsp.digicert.com - /	30 ms
6. dt5zaw6a98blc....ont.net - www.css	295 ms
7. dt5zaw6a98blc.....net - require.js	39 ms
8. dt5zaw6a98blc....ont.net - home.js	269 ms
9. dt5zaw6a98blc....net - loading.gif	171 ms
10. www.coursera.org - favicon.ico	144 ms
11. dt5zaw6a98blc....t.net - banner.js	34 ms
12. dt5zaw6a98blc....iaproregular.woff	467 ms
13. dt5zaw6a98blc....9_new-courses.png	397 ms
14. dt5zaw6a98blc....018_ace-intro.png	606 ms
15. www.coursera.org - list2	553 ms
16. dt5zaw6a98blc....t - quotemark.png	123 ms
17. dt5zaw6a98blc.....net - sprite.png	92 ms
18. www.coursera.org - signup_stats	324 ms
19. dt5zaw6a98blc....015_listmaker.jpg	407 ms
20. dt5zaw6a98blc....017_crunchies.jpg	545 ms
21. dt5zaw6a98blc....fiapromedium.woff	483 ms
22. dt5zaw6a98blc....ra_logo_small.png	298 ms
23. eventing.coursera.org - 204.min.js	600 ms

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

Question

- How can we optimize this page load process?

Network Protocols

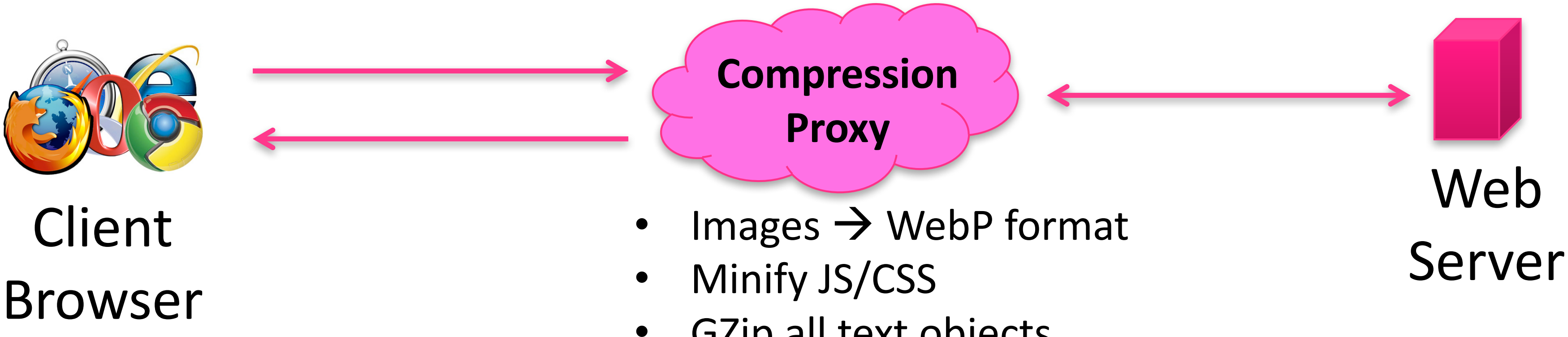
- SPDY/HTTP2
  - Multiplexes requests onto single TCP connection (one per origin)
  - Compresses HTTP headers
  - Mandatory TLS
  - Server Push: let’s servers proactively push objects to clients, without explicit requests (saves RTTs)
- QUIC
  - UDP rather than TCP
  - Reduces connection establishment for secure connections
  - Multiplexing without HOL blocking
  - Pluggable congestion control

Caches

- Browser caches
  - Caching rules specified in server-generated HTTP headers
  - Content served only when cached HTTP headers exactly match those in new request
- CDNs in network (e.g., Akamai)
  - Run in network and shared across clients
- Challenge: dynamically generated content, personalization

Compression Proxies

- Compress objects in-flight between clients and servers → main goal is to reduce bandwidth usage!



Compression Proxy

Client Browser

Web Server

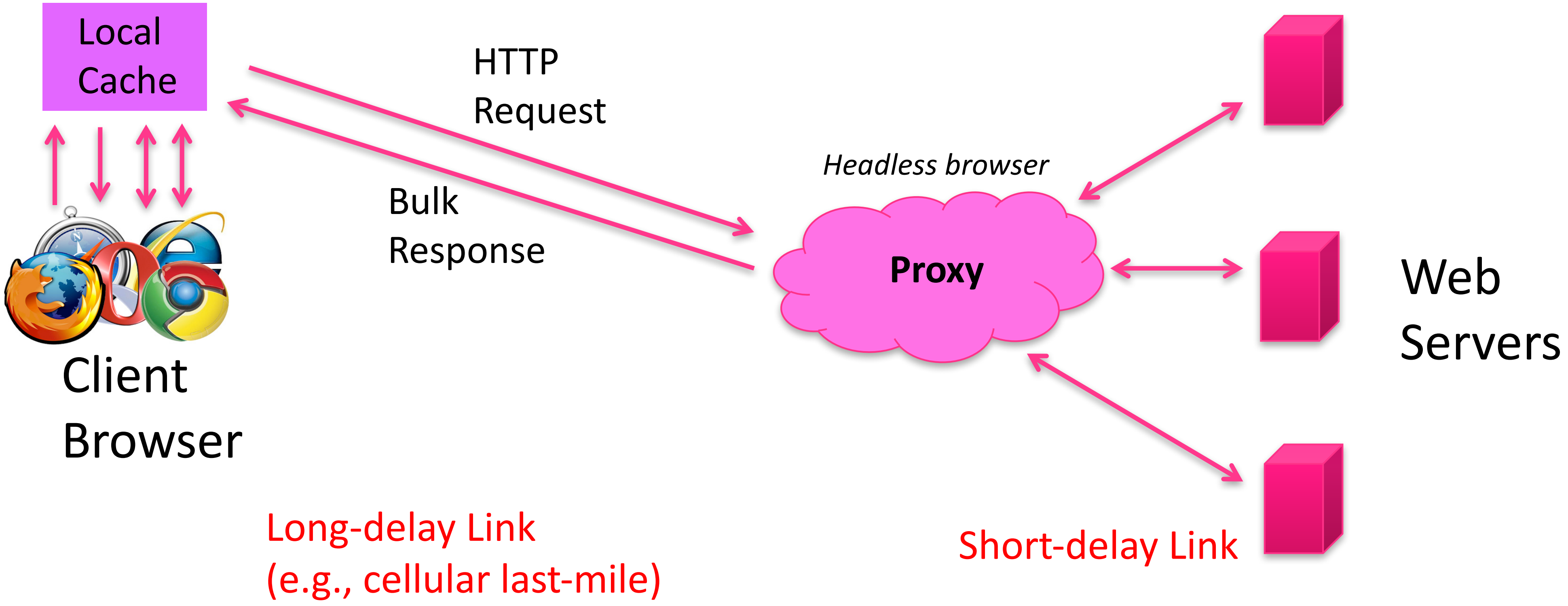
- Images → WebP format
- Minify JS/CSS
- GZip all text objects

Example: Google Flywheel (NSDI ‘15)

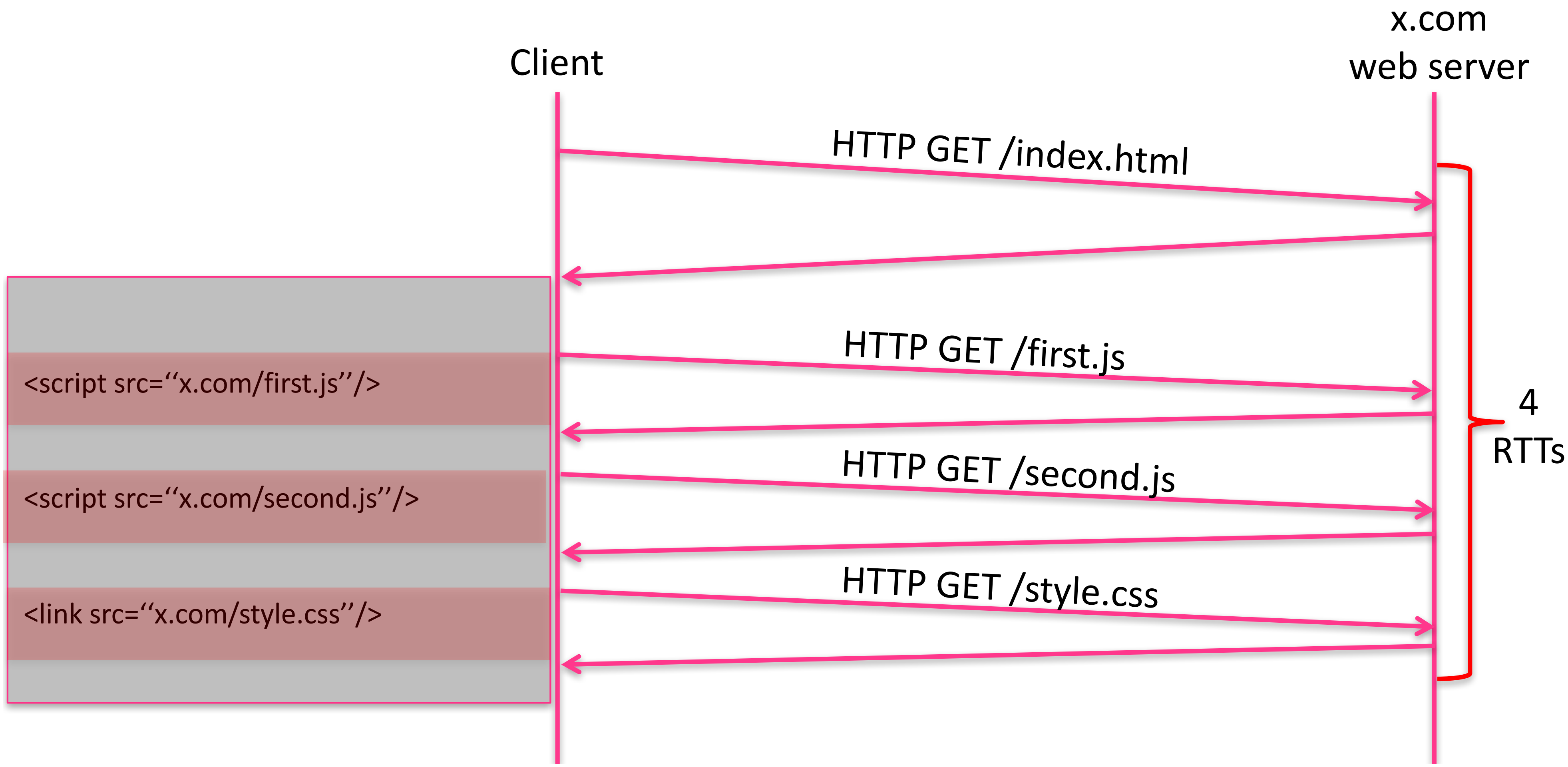
- Pages 58% smaller at the median (most benefit from image compression)
- Page load time increased by 6% at median
  - Does not reduce # of RTTs required to load page
  - Indirection to contact proxy can increase magnitude of each RTT

Cloud Browsers

- Incur RTTs required to load page over low-delay, high bandwidth proxy links
- Examples: Opera Mini, Amazon Silk, Parcel, Cumulus



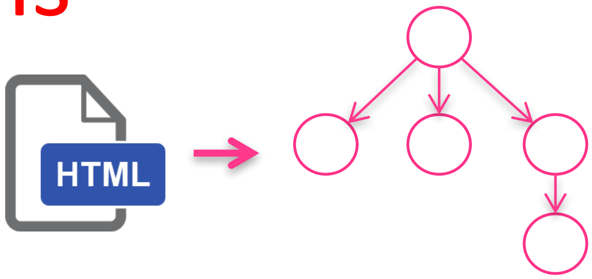
Page Load

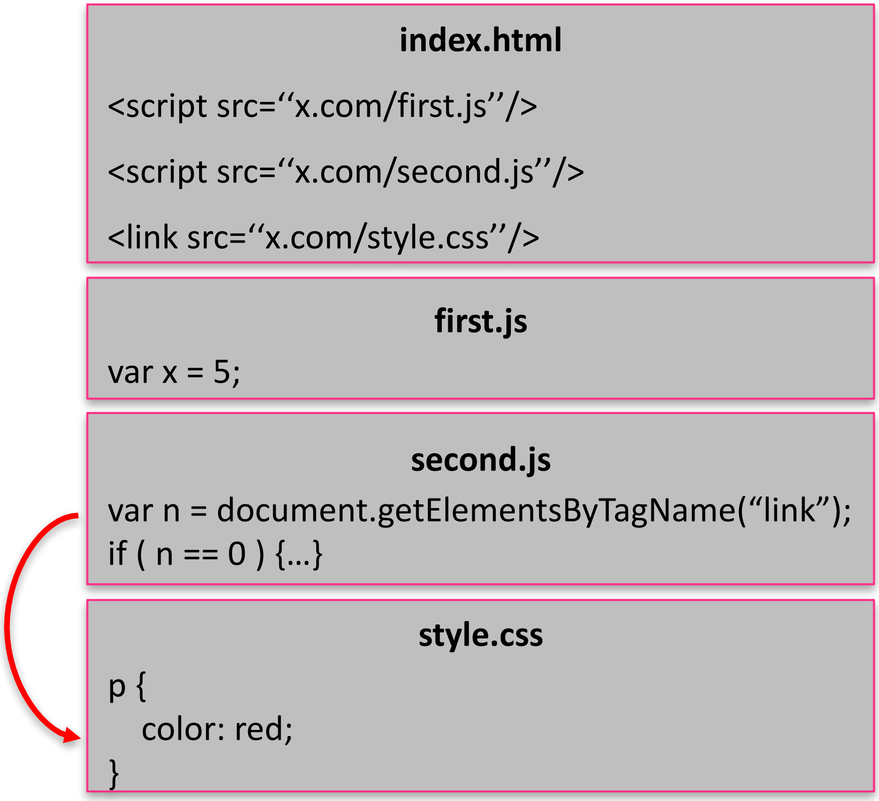

x.com
web server
Client
HTTP GET /index.html
<script src="x.com/first.js"/>
HTTP GET /first.js
<script src="x.com/second.js"/>
HTTP GET /second.js
<link src="x.com/style.css"/>
HTTP GET /style.css
4
RTTs

Dependency Graphs

Model page loads as directed acyclic graphs

- Page load time = time to completely resolve dependency graph





index.html

```
<script src=“x.com/first.js”/>

<script src=“x.com/second.js”/>

<link src=“x.com/style.css”/>
```

first.js

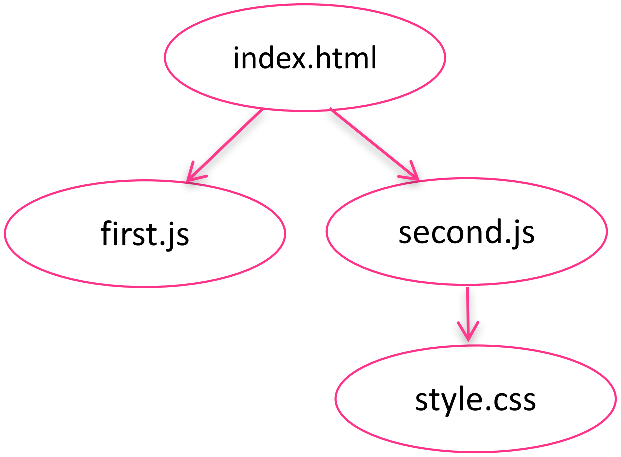
```
var x = 5;
```

second.js

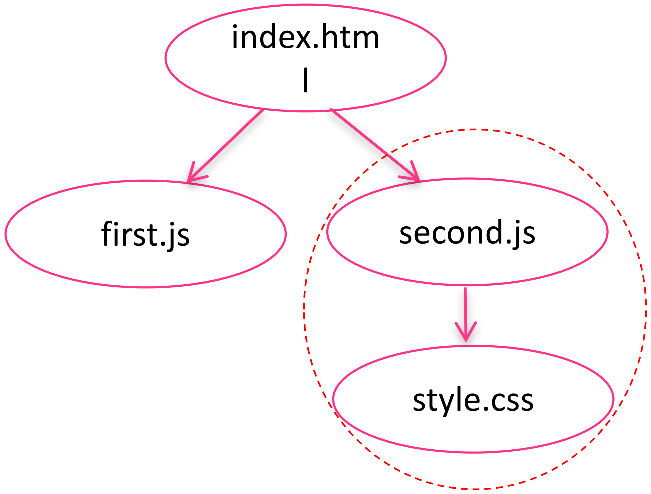
```
var n = document.getElementsByTagName(“link”);
if ( n == 0 ) {…}
```

style.css

```
p {
   color: red;
}
```



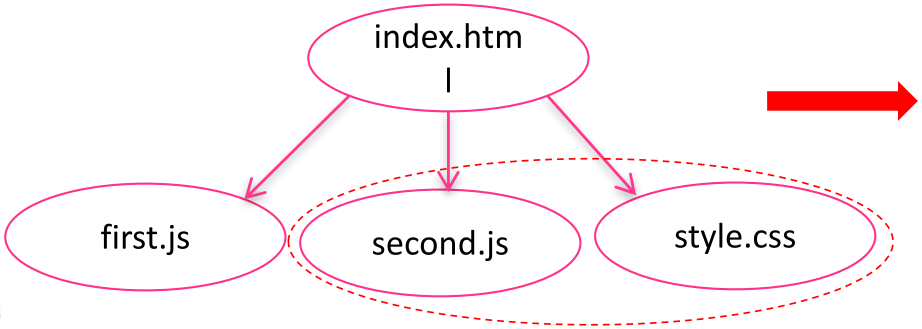
Dependency Graphs



True Dependency Graph

3 RTTs

Missing Dependencies



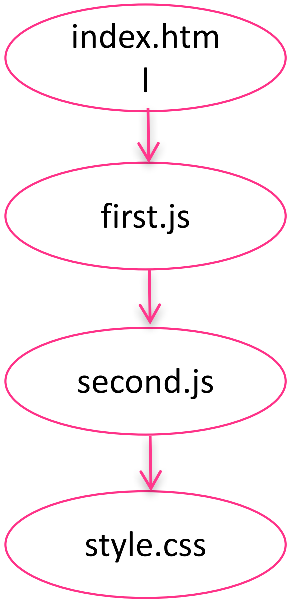
Lexical HTML Dependencies

```
<script src=‘‘x.com/first.js’’/>

<script src=‘‘x.com/second.js”/>

<link src=‘‘x.com/style.css’’/>
```

Conservative Assumptions

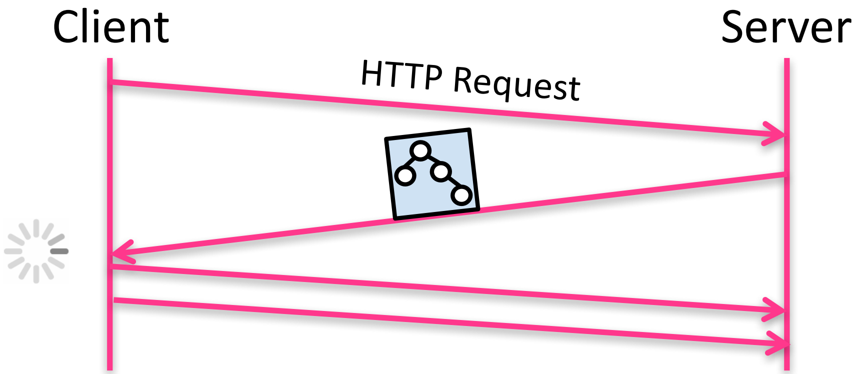


HTML Tag Order

4 RTTs

Outline

- Scout: tracks fine-grained dependencies between page's objects
  - Traditional dependency graphs miss 30% of edges
- Polaris: dynamic client-side scheduler written in JavaScript
  - Uses fine-grained dependencies to reduce page load times



  - 34% faster (1.3 seconds) on 12 Mbits/s link with 100 ms RTT

Scout

- Scout tracks many different dependencies across a page's state

3 Types of Dependencies

Write/Read

first.js

```
x = 6;
```

second.js

```
y = x + 5;
```

Read/Write

first.js

```
x = [1,3,5];
```

second.js

```
y = x.length;
```

third.js

```
x.push(7);
```

Write/Write

first.js

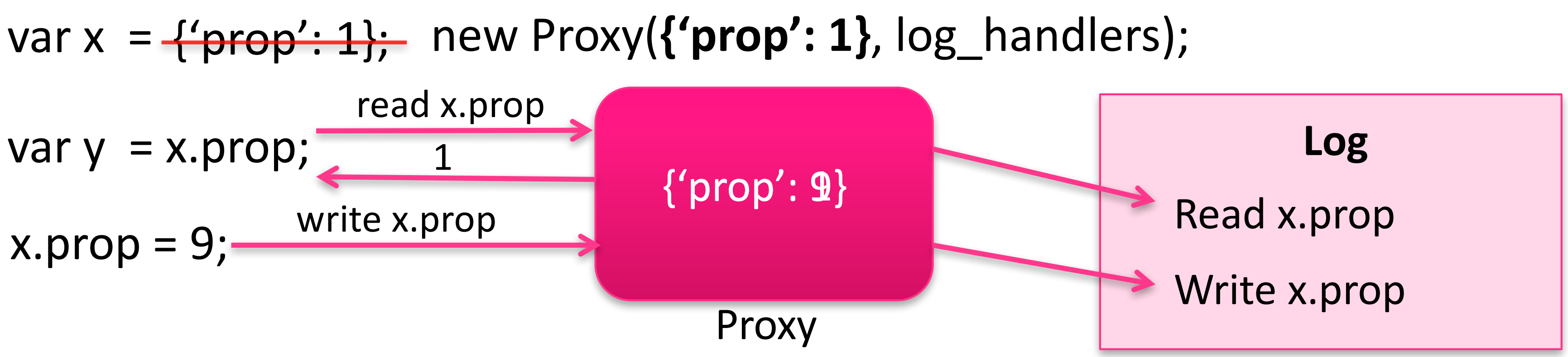
```
alert(“first message”);
```

second.js

```
alert(“second message”);
```

Tracking Dependencies

- JS proxy objects



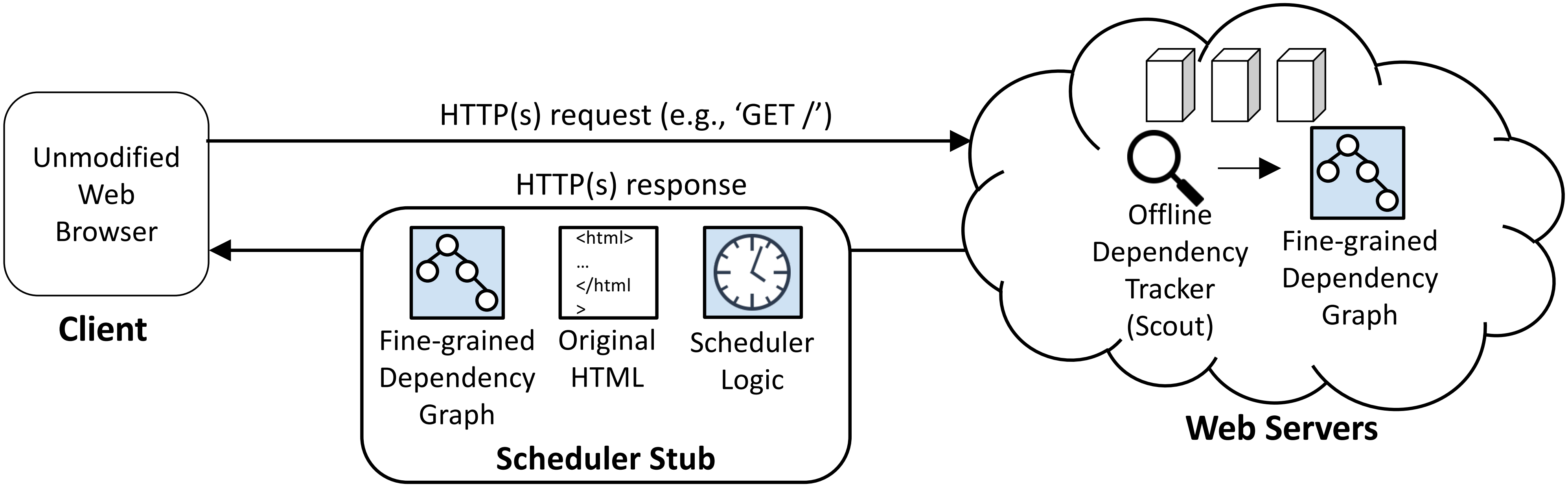
var x = ~~{‘prop’: 1};~~ new Proxy(**{‘prop’: 1}**, log_handlers);

var y = x.prop;

x.prop = 9;

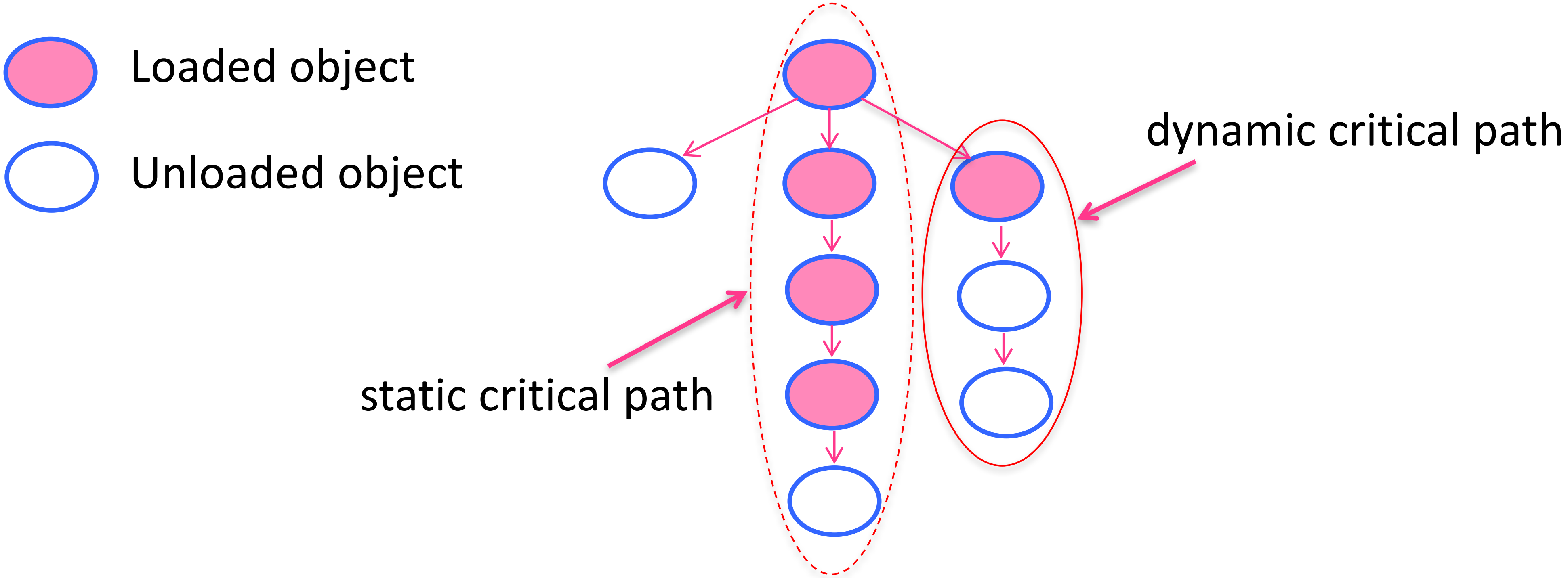
- Many others described in paper
  - Global variables
  - Recursive proxying (e.g., x.y.z)
  - DOM (e.g., document.getElementById(“foo”))

Polaris

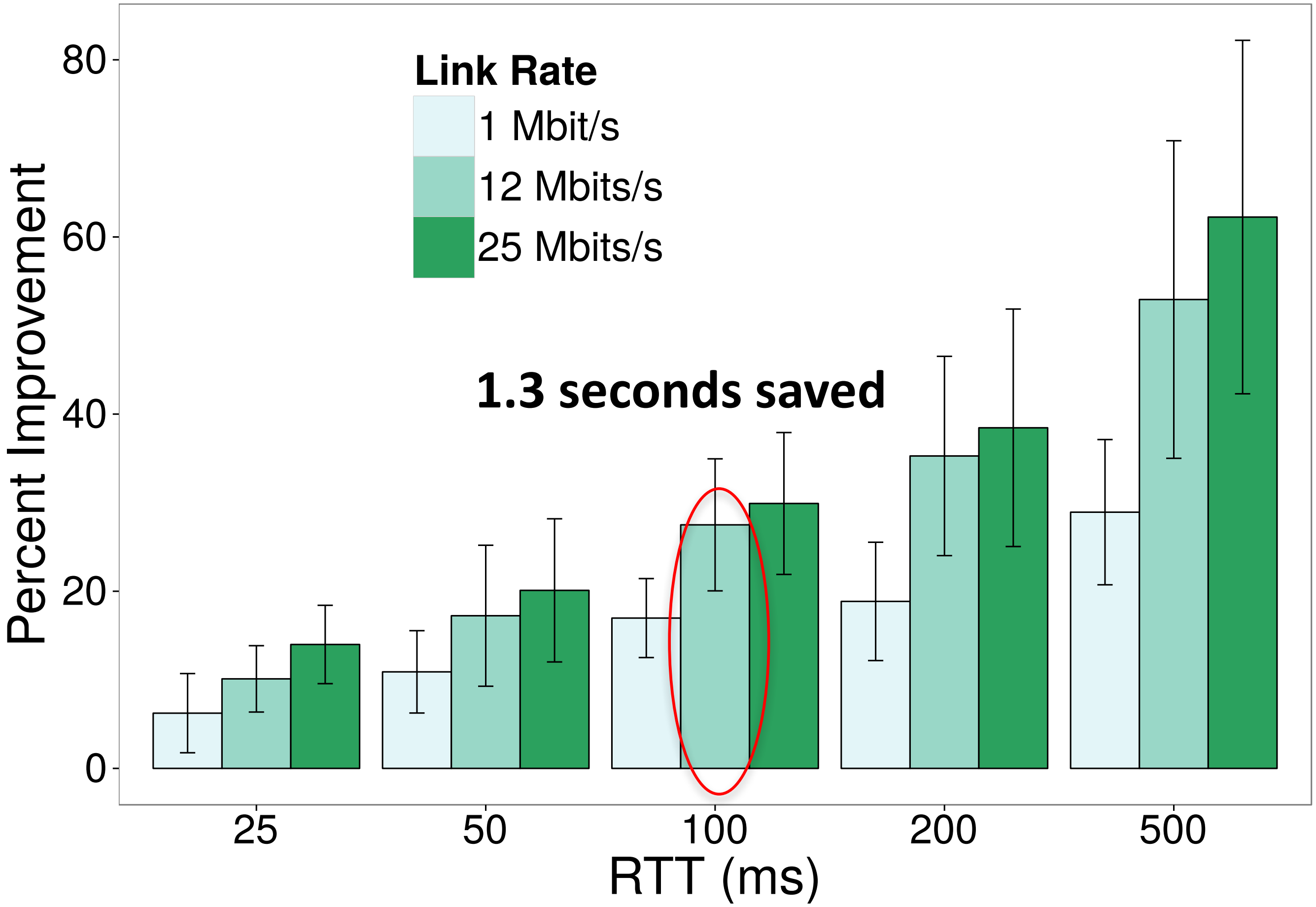

Unmodified Web Browser
Client
HTTP(s) request (e.g., 'GET /')
HTTP(s) response
<html>
…
</html
>
Fine-grained Dependency Graph
Original HTML
Scheduler Logic
Scheduler Stub
Offline Dependency Tracker (Scout)
Fine-grained Dependency Graph
Web Servers

Request Scheduling with Polaris

Always fetch objects on the **dynamic critical path**



Evaluating Polaris



- Gains increase with increasing RTT
- Gains increase with increasing link rate
- Baseline is Firefox
- Large error bars: page structure matters too!

Dynamically Generated Dependency Graphs

- Scout can be integrated into the pipeline that generates dynamic content

- JavaScript nondeterminism (e.g., Math.random())
  - Eliminate it (e.g., deterministic seed for Math.random())
  - Track all possible execution paths (ensures correctness, but overconstrains page load)

- Content may vary dynamically, but page structure is often stable (Klotski, NSDI '15)
  - Example: Washington Post uses fixed templates for articles

- If pages have random structures, Polaris (nor any prior structure-based optimizer) may not reduce page load times
  - But the page will still load to completion (defaults to conservative approach)!