Introduction to Computer Networks

Application Layer Overview

Computer Science & Engineering
UNIVERSITY of WASHINGTON

Where we are in the Course

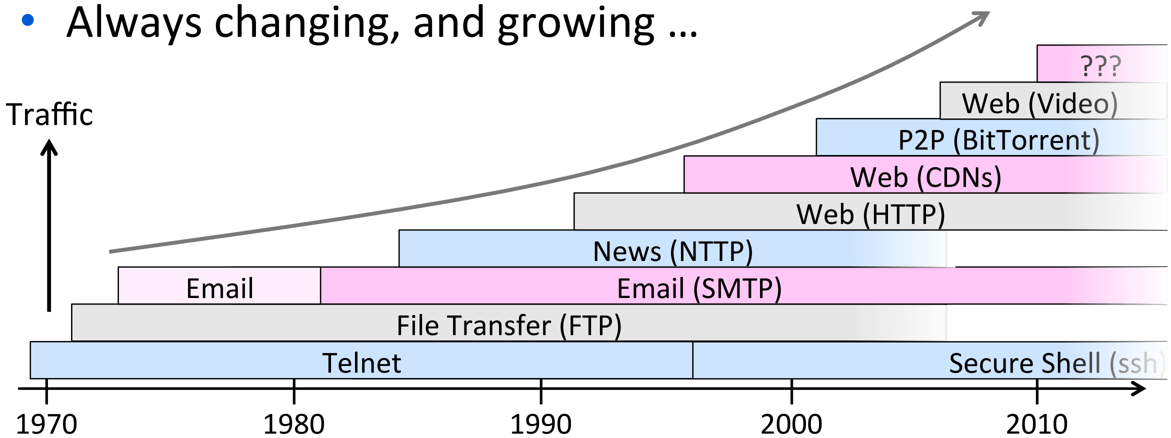
- Starting the Application Layer!
  - Builds distributed "network services" (DNS, Web) on Transport services

Application
Transport
Network
Link
Physical

95

Evolution of Internet Applications

- Always changing, and growing …



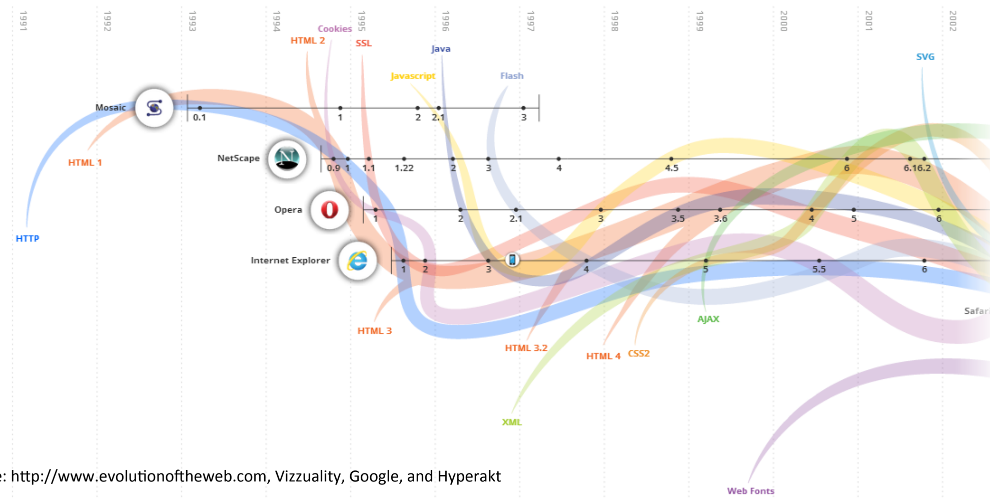
96

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

97

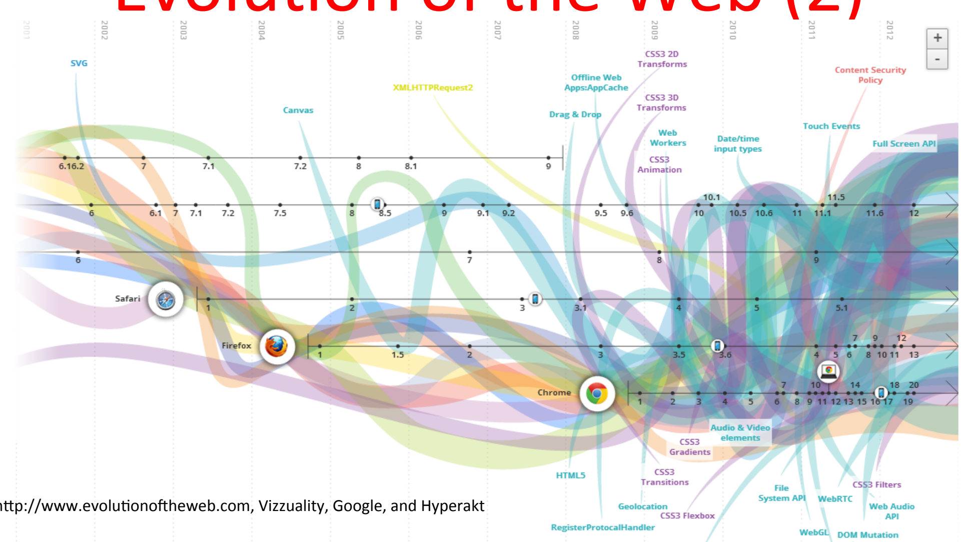
Evolution of the Web



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

98

Evolution of the Web (2)

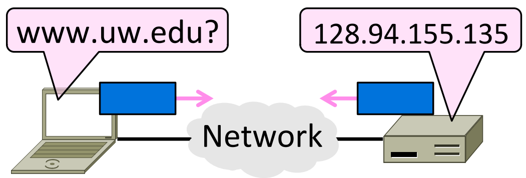


Source: http://www.evolutionoftheweb.com, Vizzuality, Google, and Hyperakt

99

Topic

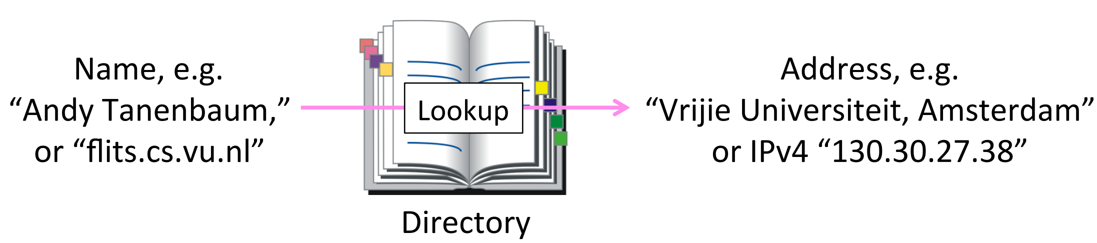
- The DNS (Domain Name System)
  - Human-readable host names, and more
  - Part 1: the distributed namespace



100

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



101

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 …

102

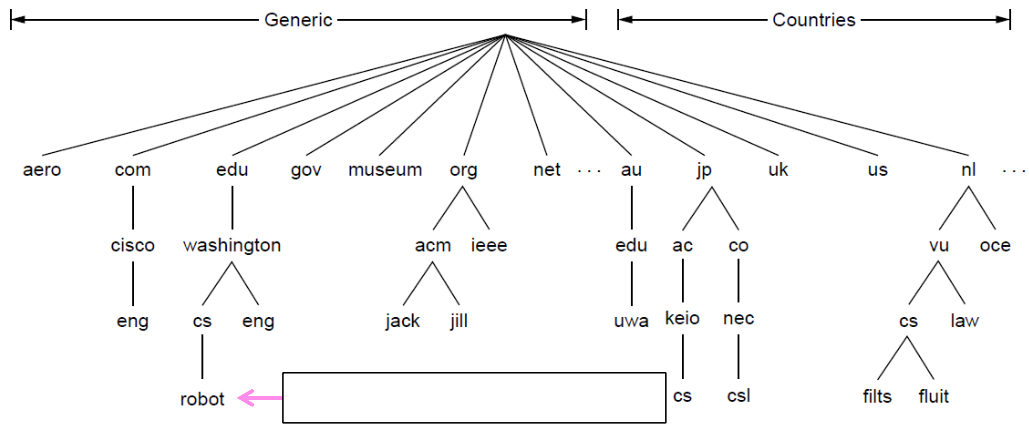
DNS

- 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

103

DNS Namespace

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

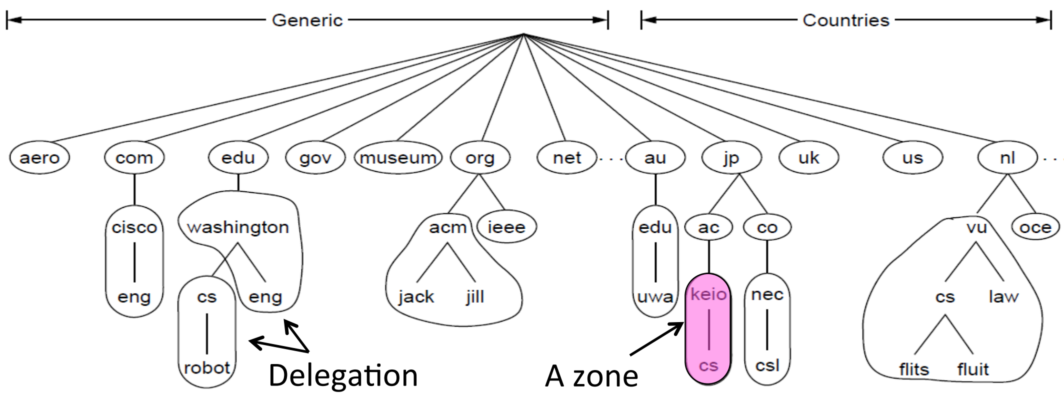


TLDs (Top-Level Domains)

- 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 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 ("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

108

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            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
www            86400  IN  CNAME  star.cs.vu.nl   of computers
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
```

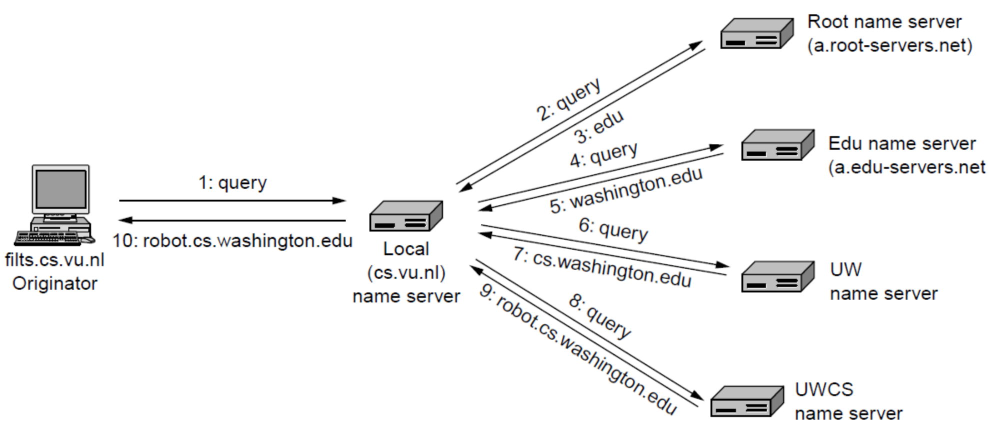
109

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

112

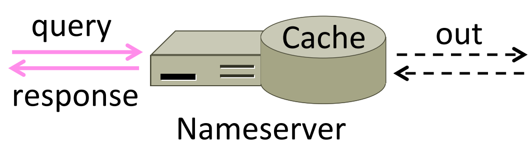
Iterative vs. Recursive Queries (2)

- 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

113

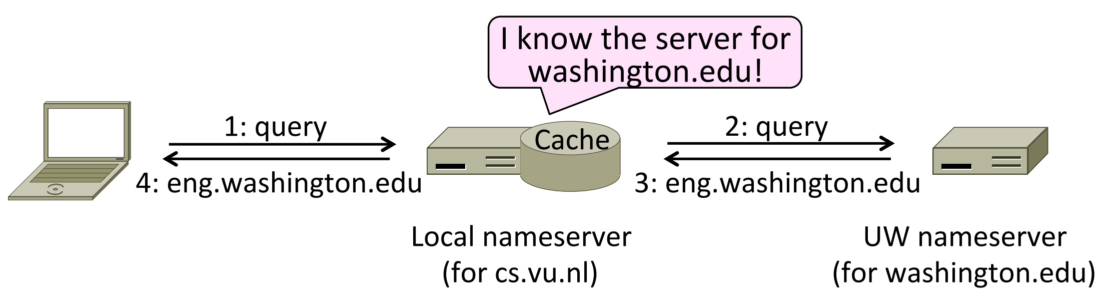
Caching

- 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

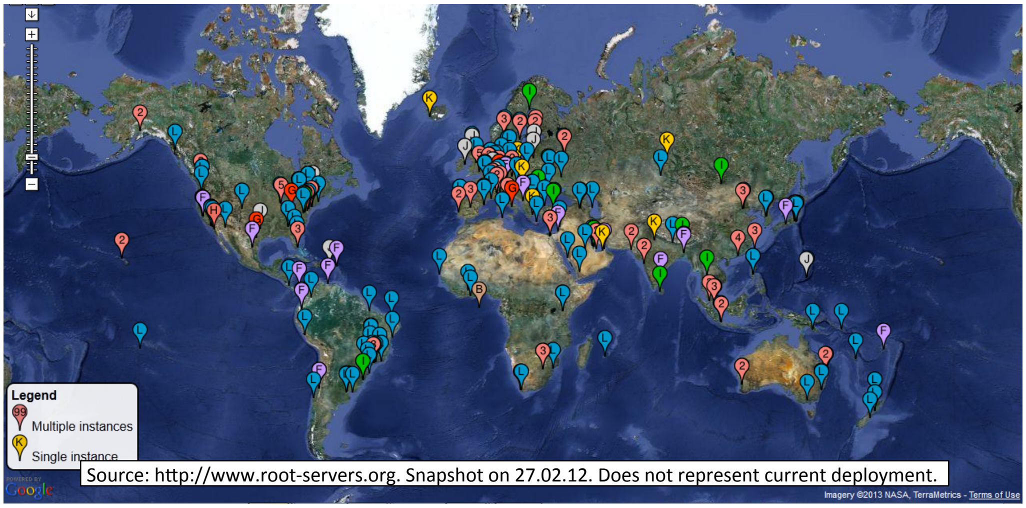
116

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.x.x)
  - Servers are IPv4 and IPv6 reachable

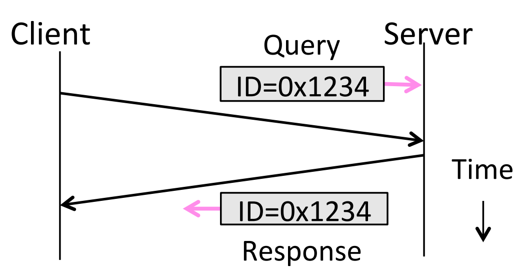
117

Root Server Deployment



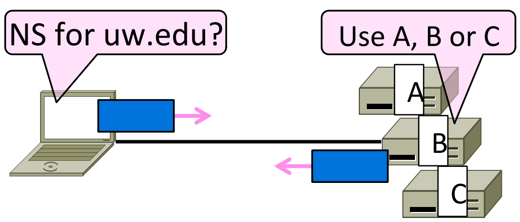
DNS Protocol

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



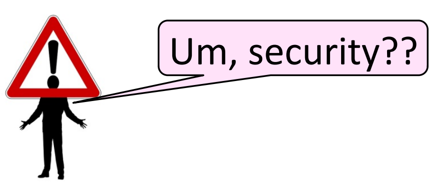
DNS Protocol (2)

- 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. We'll look at it later



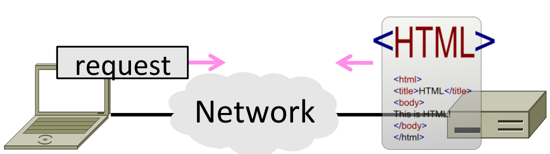
Introduction to Computer Networks

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

Computer Science & Engineering
UNIVERSITY of WASHINGTON

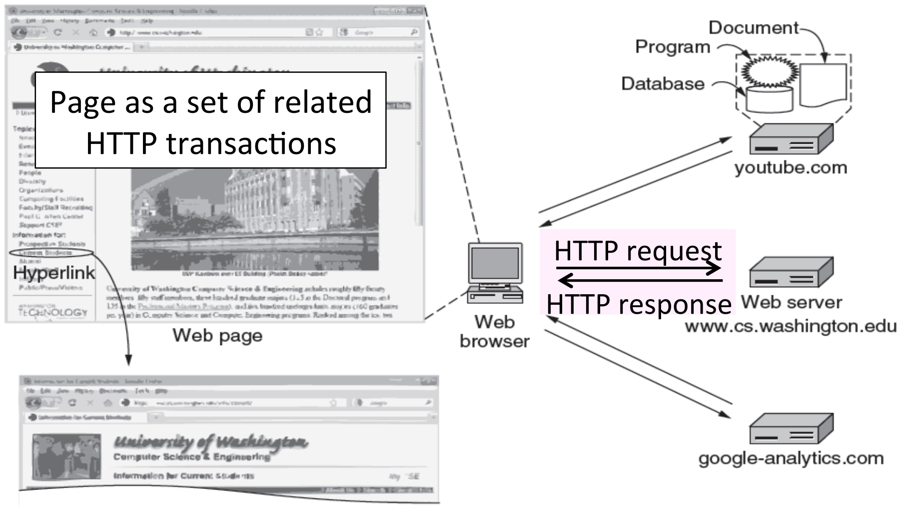
Topic

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



123

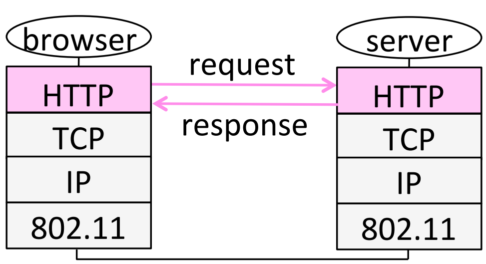
Web Context



124

Web Protocol Context

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



125

Fetching a Web page with HTTP

- Start with the page URL:

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

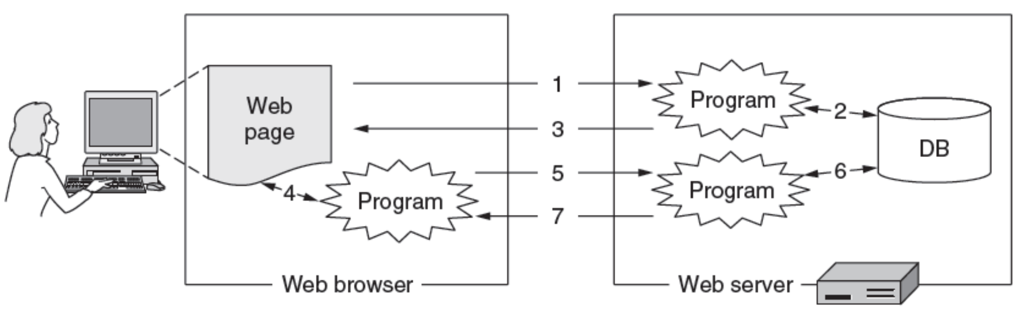
  Protocol Server Page on server

- 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

126

Static vs Dynamic Web pages

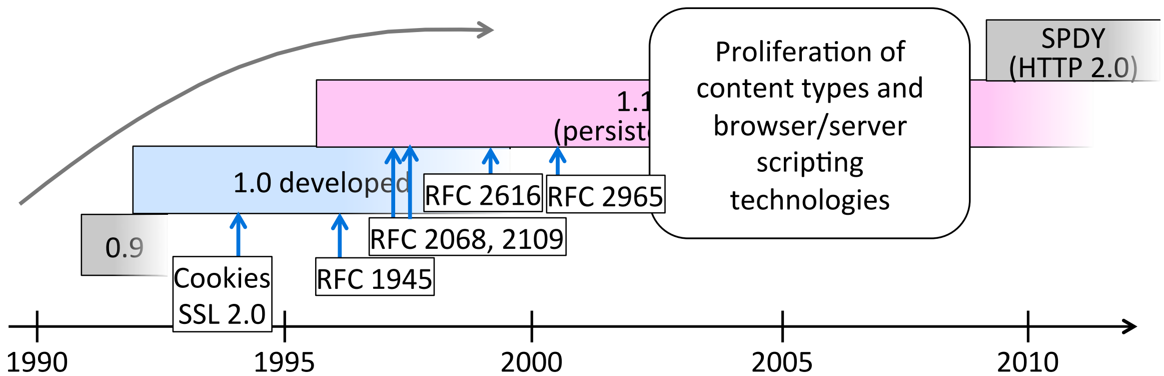
- Static web page is a file contents, e.g., image
- Dynamic web page is the result of program execution
  – Javascript on client, PHP on server, or both



127

Evolution of HTTP

- Consider security (SSL/TLS for HTTPS) later



128

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)

129

HTTP Protocol (2)

- Commands used in the request

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

Fetch page → GET

Upload data → POST

130

HTTP Protocol (3)

- Codes returned with the response

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

Yes! → 2xx

131

HTTP Protocol (4)

- Many header fields specify capabilities and content
  - E.g., Content-Type: text/html, Cookie: lect=8-4-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

Introduction to Computer Networks

HTTP Performance (§7.3.4, §7.5.2)

Computer Science & Engineering
UNIVERSITY of WASHINGTON

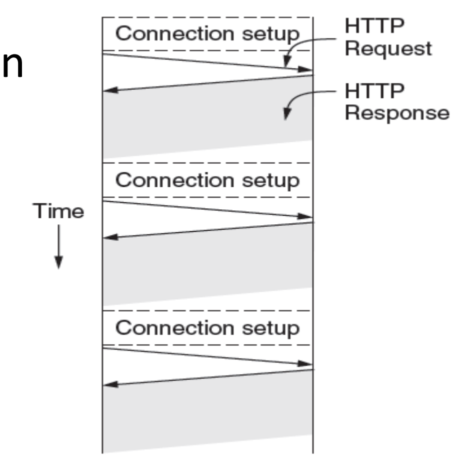
PLT (Page Load Time)

- 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

134

Early Performance

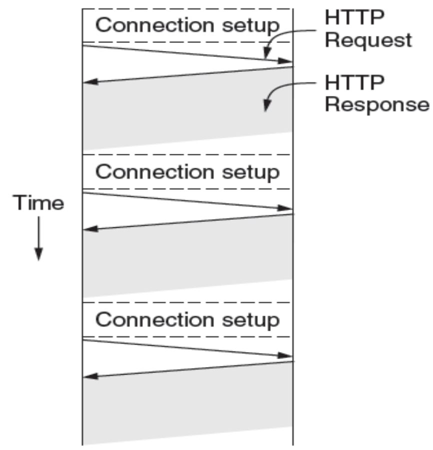
- HTTP/1.0 used one TCP connection to fetch one web resource
  - Made HTTP very easy to build
  - But gave fairly poor PLT…



135

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



136

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

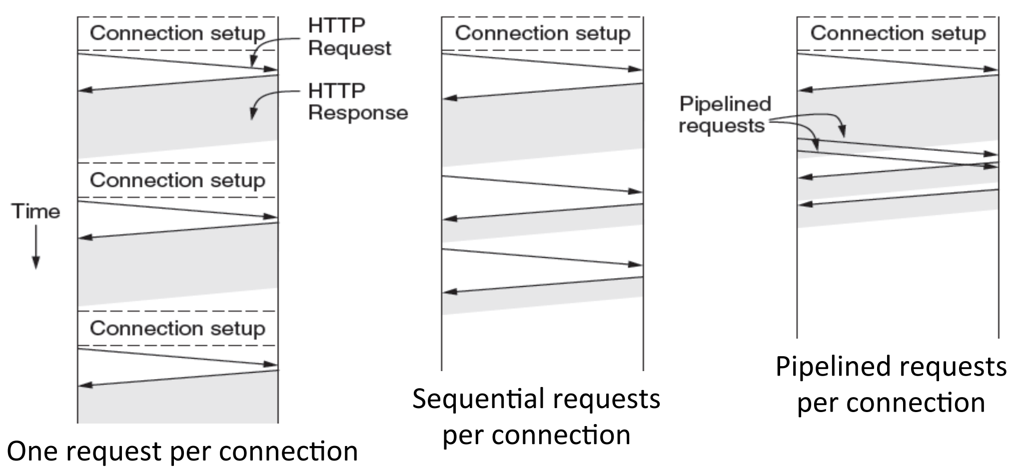
137

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

138

Persistent Connections (2)



139

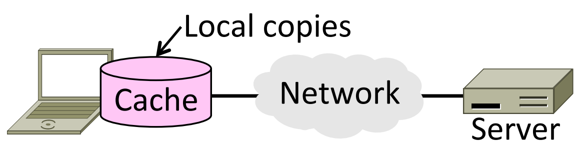
Persistent Connections (3)

- 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?)

140

Web Caching

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

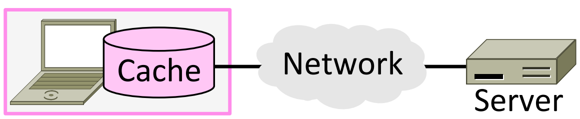


- Key question:
  - When is it OK to reuse local copy?

141

Web Caching (2)

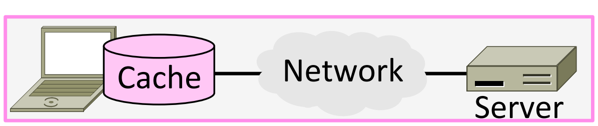
- Locally determine copy is still valid
  - Based on expiry information such as "Expires" header from server
  - Or use a heuristic to guess (cacheable, freshly valid, not modified recently)
  - Content is then available right away



142

Web Caching (3)

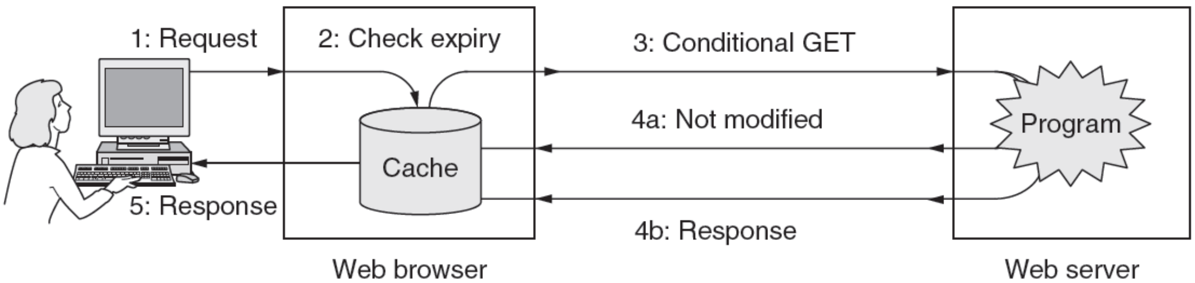
- Revalidate copy with server
  - Based on timestamp of copy such as "Last-Modified" header from server
  - Or based on content of copy such as "Etag" header from server
  - Content is available after 1 RTT



143

Web Caching (4)

- Putting the pieces together:



144

Introduction to Computer Networks

CDNs (Content Delivery Networks) (§7.5.3)

Computer Science & Engineering
UNIVERSITY of WASHINGTON

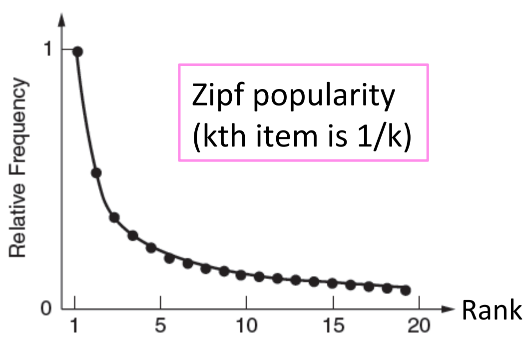
Context

- 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

148

Popularity of Content

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

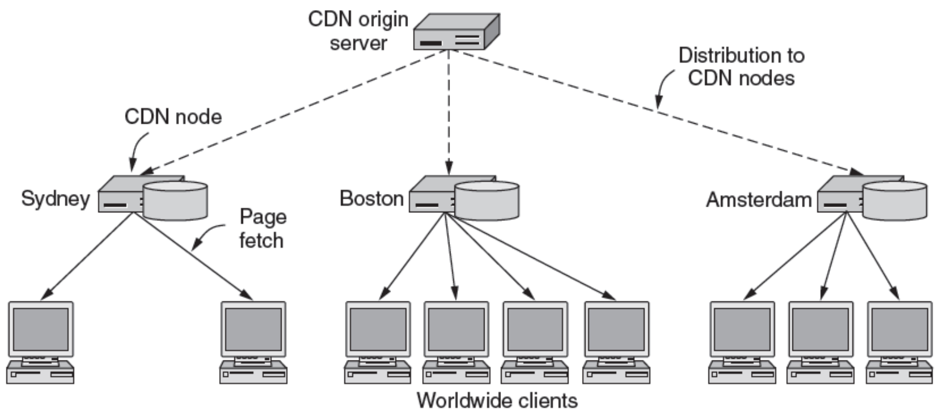


George Zipf (1902-1950)

Source: Wikipedia

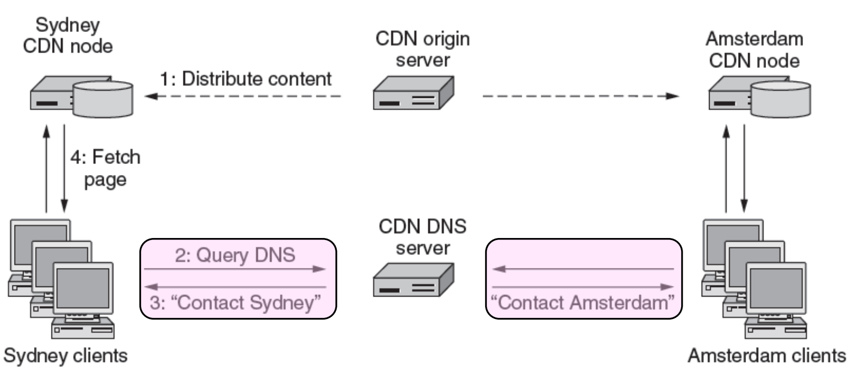
151

Content Delivery Network



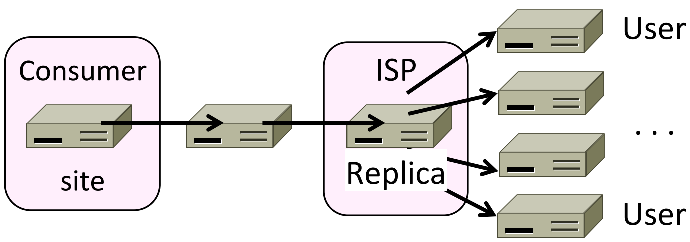
Content Delivery Network (2)

- DNS resolution of site gives different answers to clients
  - Tell each client the site is the nearest replica (map client IP)



Business Model

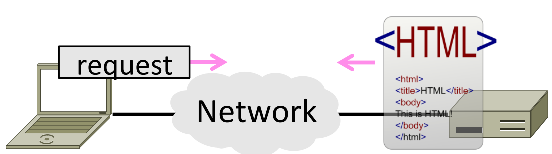
- Clever model pioneered by Akamai
  - Placing site replica at an ISP is win-win
  - Improves site experience and reduces bandwidth usage of ISP



155

Topic

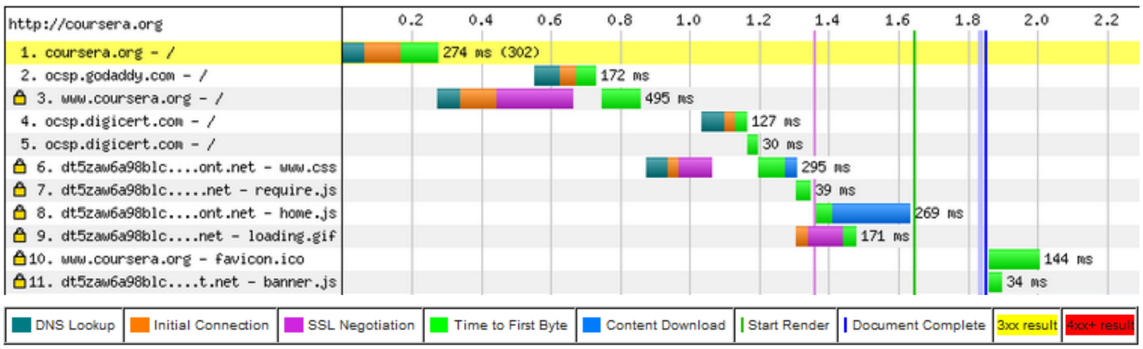
- The Future of HTTP
  - How will we make the web faster?
  - A brief look at some approaches



156

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)

157

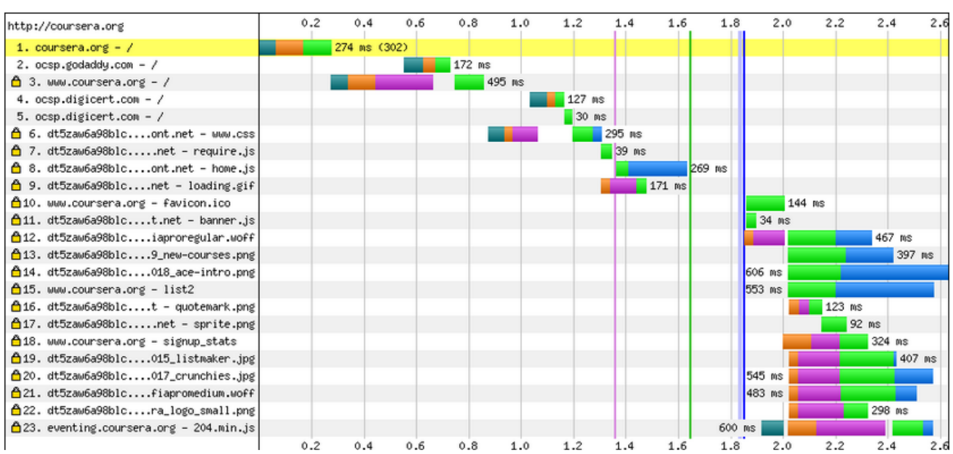
Modern Web Pages (2)

Yikes!

-23 requests

-1 Mb data

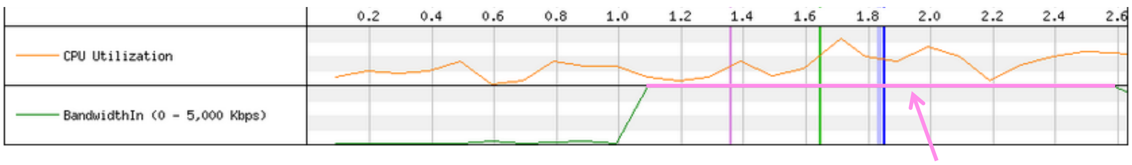
-2.6 secs



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

158

Modern Web Pages (3)



Yay! (Network used well)

- 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

159

Recent work to reduce PLT

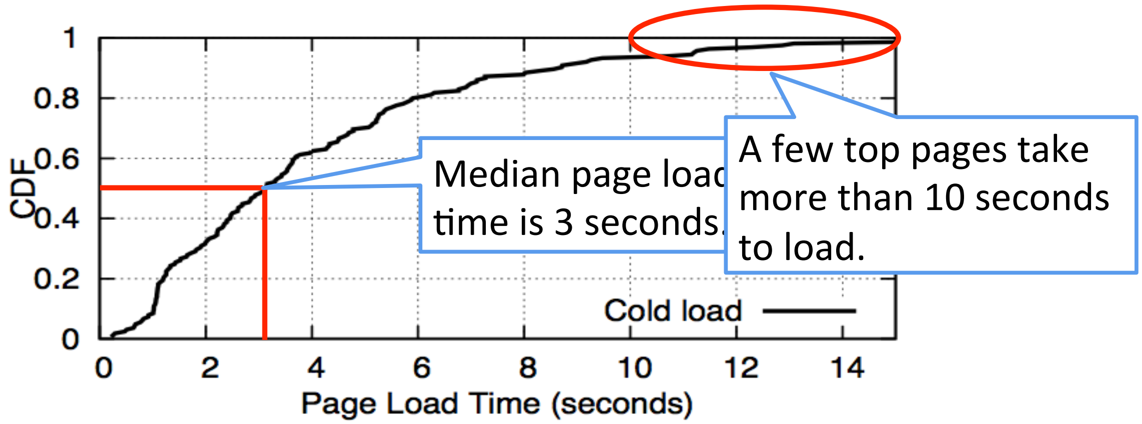
Pages grow ever more complex!
- Larger, more dynamic, and secure
- How will we reduce PLT?

1. Tools to study page load process
2. Better use of the network
   - HTTP/2 effort based on SPDY
3. Better content structures
   - mod_pagespeed server extension

160

Page load is critical, but often slow

- Amazon can increase 1% revenue by decreasing page load time by 0.1s.



Many techniques aim to optimize page load time

- Best practices: JaveScript/CSS placement, Image minification, …
- Server placement: CDNs
- Web pages and cache: mod_pagespeed, Silo
- TCP/DNS: TCP fast open, ASAP, DNS pre-resolution, TCP pre-connect

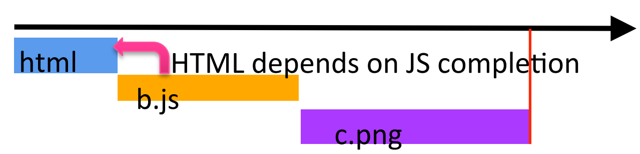
…, but optimizations don't always work.

…because , page load bottlenecks are not well understood.

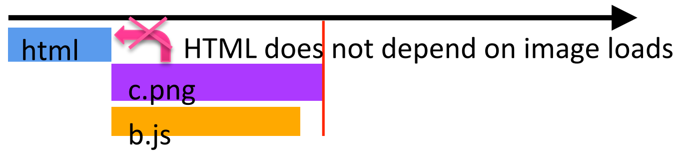
- Page load process is not strictly streamlined
- Page load activities are inter-dependent, leading to bottlenecks

Dependency example

```
<html>
  <script src="b.js"></script>
  <img src="c.png"/>
</html>
```



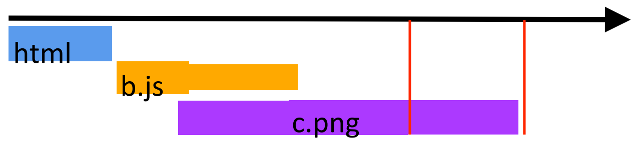

```
<html>
  <img src="c.png"/>
  <script src="b.js"></script>
</html>
```



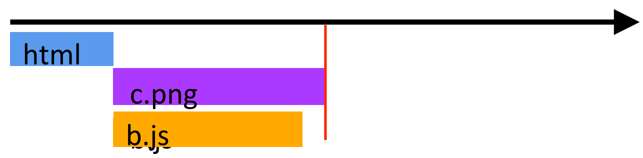
Example: Dependency and bottleneck

Possible optimization: Make JS loads faster

```
<html>
  <script src="b.js"></script>
  <img src="c.png"/>
</html>
```




```
<html>
   <img src="c.png"/>
  <script src="b.js"></script>
</html>
```

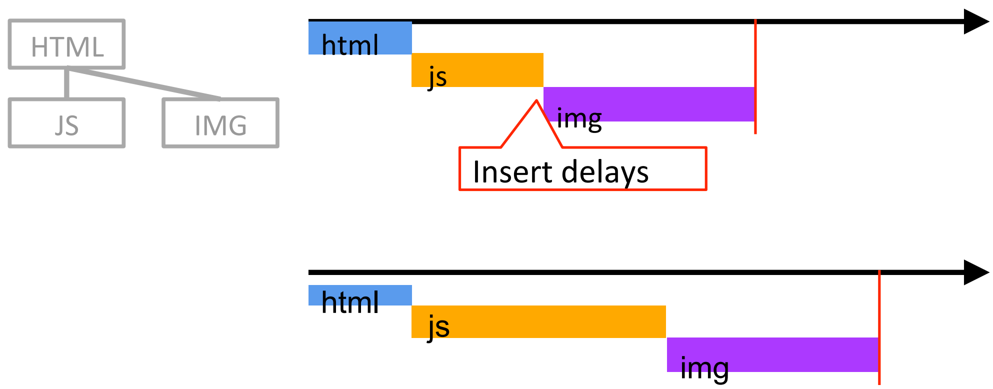


Dependency and page structure key are the key to identifying bottlenecks

Methodology to infer dependencies

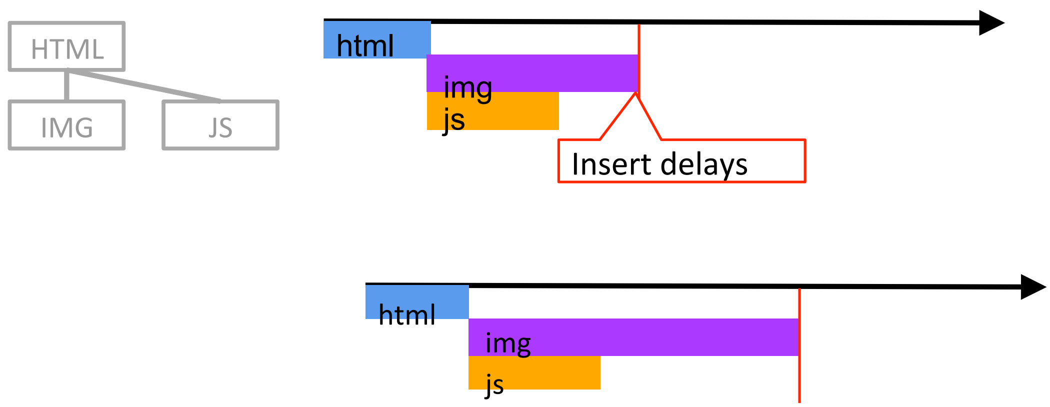
- Design test pages
- Examine documentation
- Inspect browser code

Test pages (1 of 2)



Suggests the JS loads affects image load

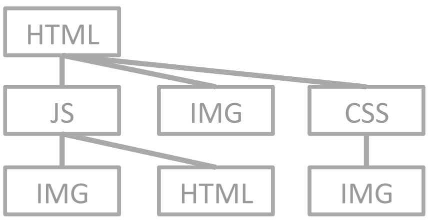
Test pages (2 of 2)



Suggests image loads *does not* affect JS load

Reverse engineer page loads with test pages

- Use developer tools to measure timing

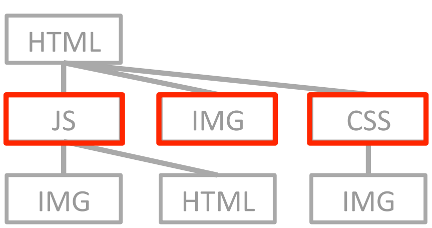


An example Web page

8

Reverse engineer page loads with test pages

- Use developer tools to measure timing
  - An object follows another



An example Web page

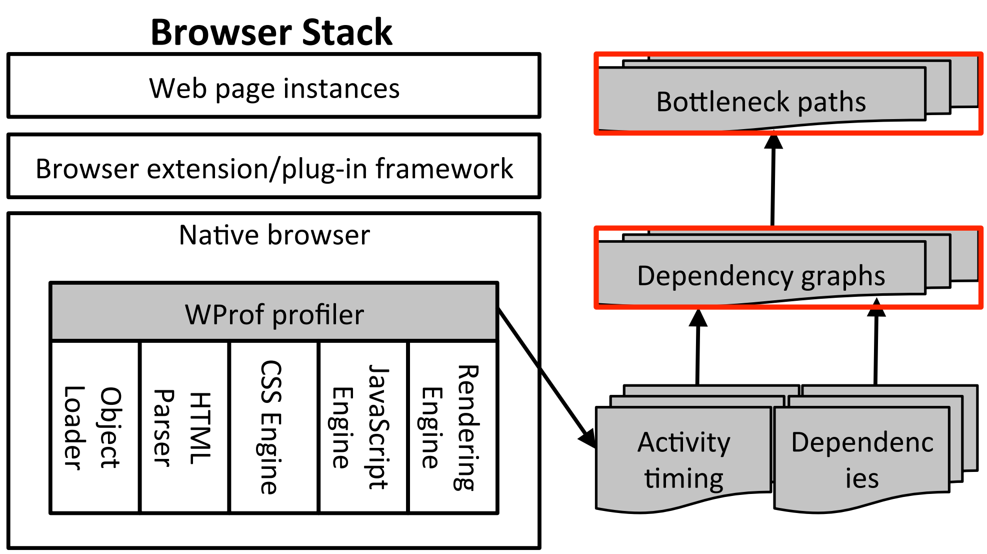
8

Dependency policies

Dependency	Name	Definition
Flow	F1	Loading an object → Parsing the tag that references the object
	F2	Evaluating an object → Loading the object
	F3	Parsing the HTML page → Loading the first block of the HTML page*
	F4	Rendering the DOM tree → Updating the DOM
	F5	Loading an object referenced by a JavaScript or CSS → Evaluating the JavaScript or CSS*
	F6	Downloading/Evaluating an object → Listener triggers or timers
Output	O1	Parsing the next tag → Completion of a previous JavaScript download and evaluation
	O2	JavaScript evaluation → Completion of a previous CSS evaluation
	O3	Parsing the next tag → Completion of a previous CSS download and evaluation
Lazy/Eager binding	B1	[Lazy] Loading an image appeared in a CSS → Parsing the tag decorated by the image
	B2	[Lazy] Loading an image appeared in a CSS → Evaluation of any CSS that appears in front of the tag decorated by the image
	B3	[Eager] Preloading embedded objects does not depend on the status of HTML parsing. (breaks F1)
Resource constraint	R1	Number of objects fetched from different servers → Number of TCP connections allowed per domain
	R2	Browsers may execute key computational activities on the same thread, creating dependencies among the activities.This dependency is determined by the scheduling policy.

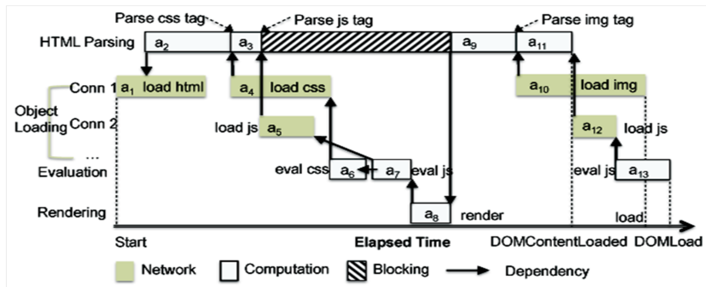
* An activity depends on *partial* completion of another activity.

WProf architecture



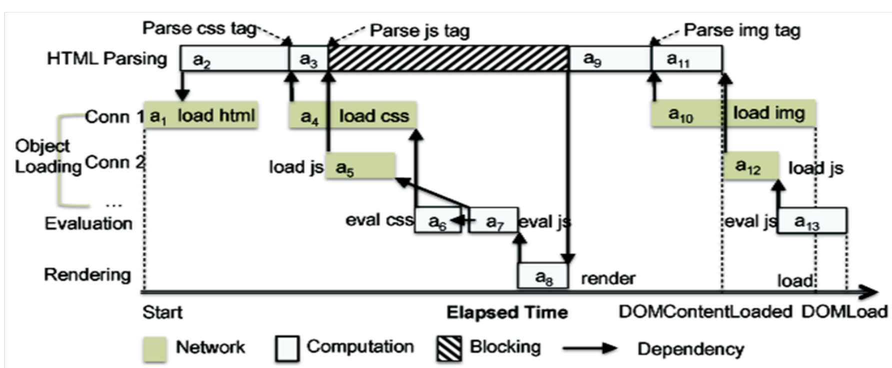
Dependency graph

```
<html>
  <head>
    <link rel="stylesheet" src="./main.css">
    <script src="./main.js" />
  </head>
  <!--request a JS-->
  <body onload="...">
    <img src="test.png" />
  </body>
</html>
```



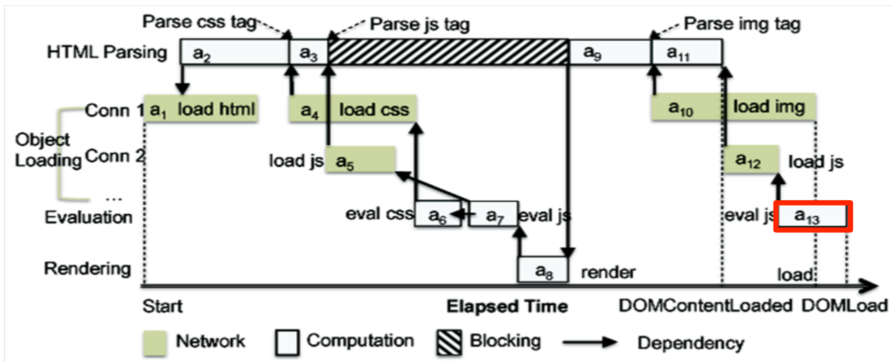
Critical path analysis

Critical path: the longest bottleneck path.



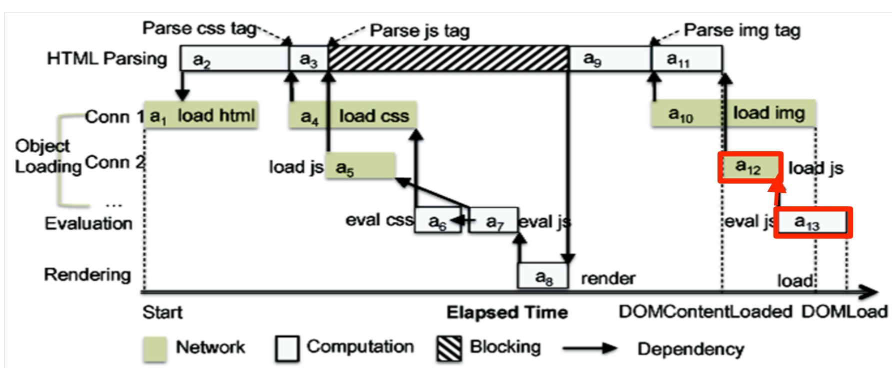
Critical path analysis

Critical path: the longest bottleneck path.



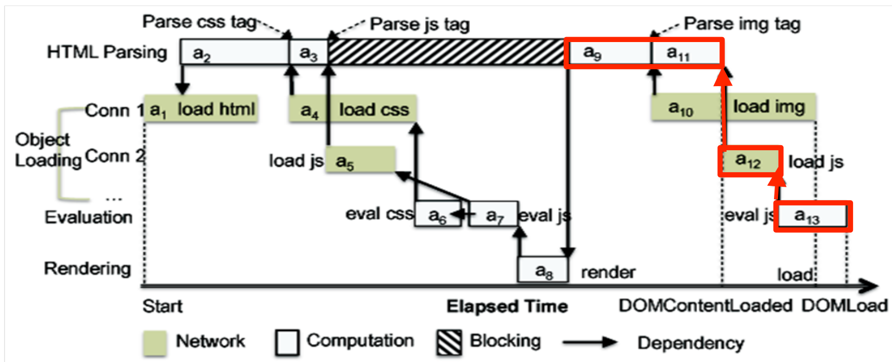
Critical path analysis

Critical path: the longest bottleneck path.



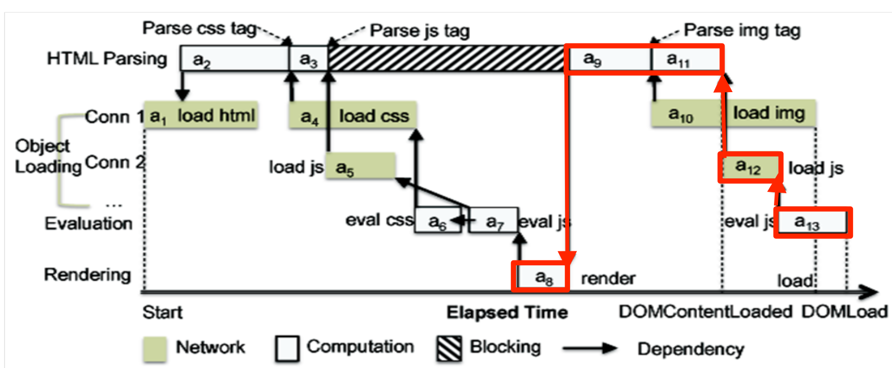
Critical path analysis

Critical path: the longest bottleneck path.



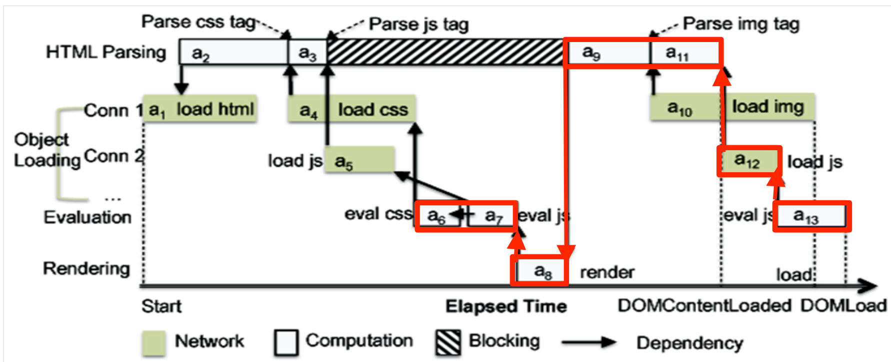
Critical path analysis

Critical path: the longest bottleneck path.



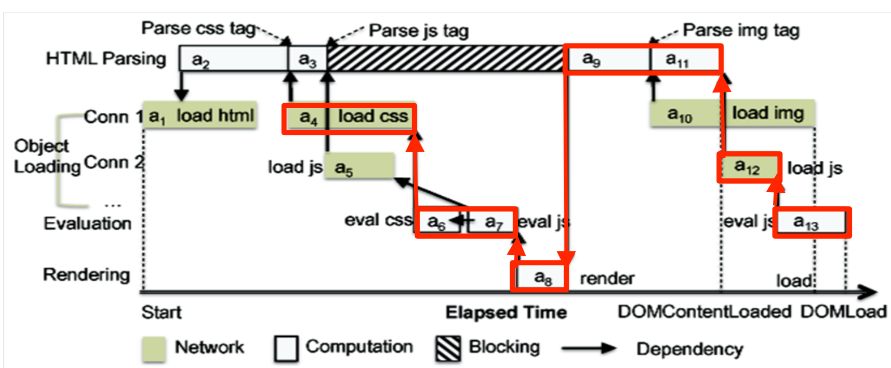
Critical path analysis

Critical path: the longest bottleneck path.



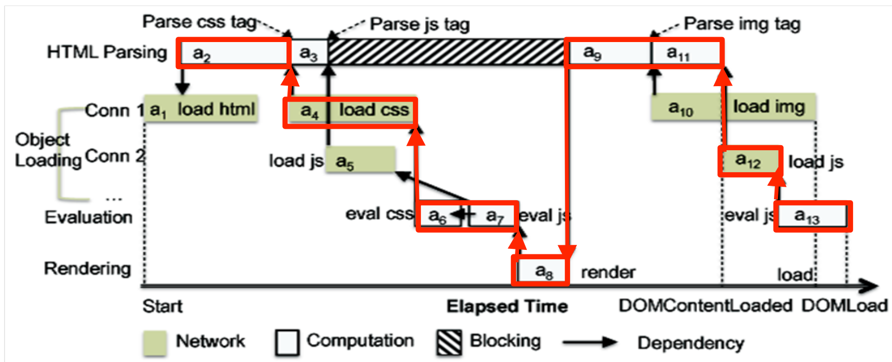
Critical path analysis

Critical path: the longest bottleneck path.



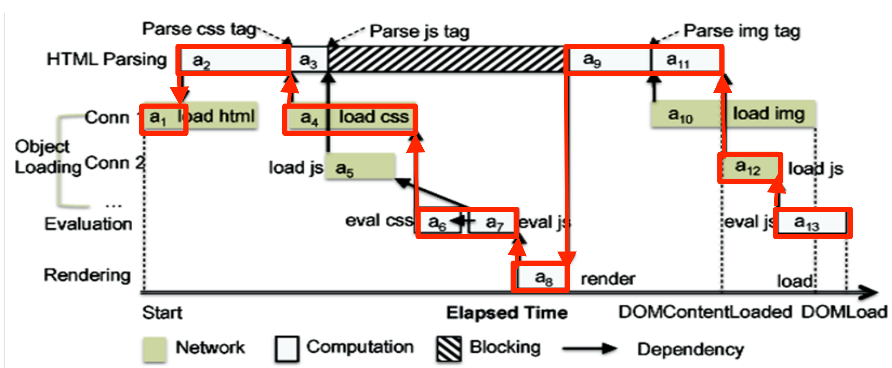
Critical path analysis

Critical path: the longest bottleneck path.



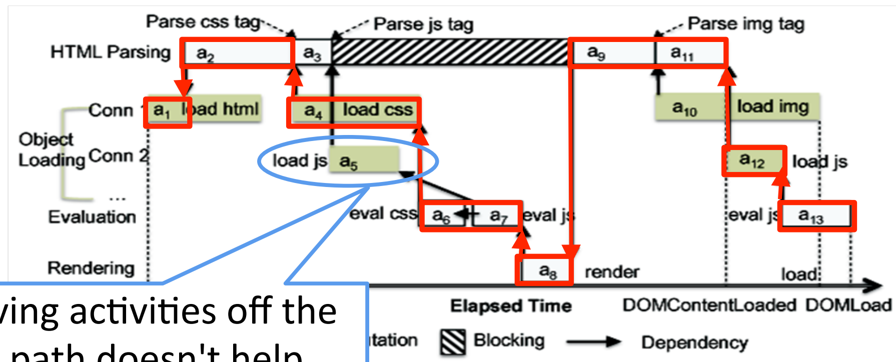
Critical path analysis

Critical path: the longest bottleneck path.



Critical path analysis

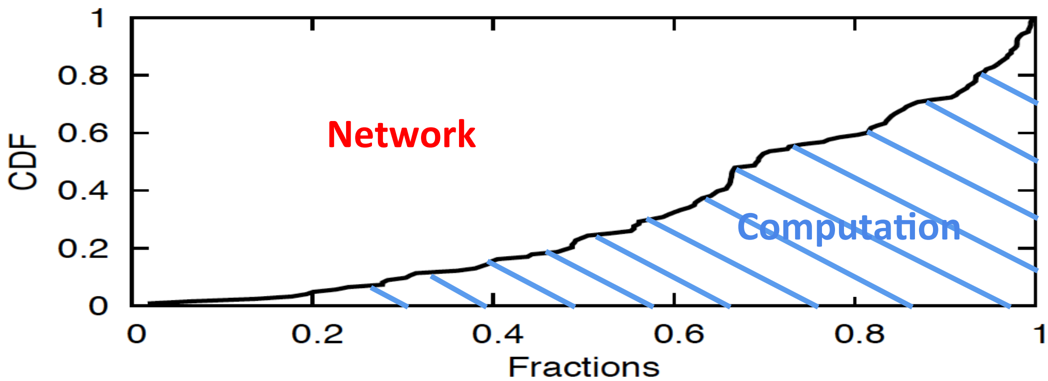
Critical path: the longest bottleneck path.



Improving activities off the critical path doesn't help page load.

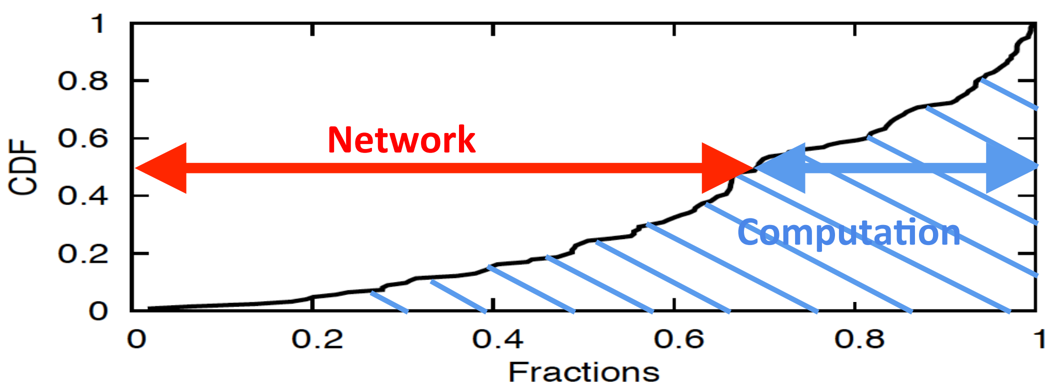
How much does the network contribute to page load time?

Computation is significant



Network/Computation as a fraction of page load time

Computation is significant



Computation is ~35% of page load time (median) on the critical path.

How much does caching help page load performance?

How much does caching help?

- Caching eliminates 80% Web object loads

- It doesn't reduce page load time as much
- Caching only eliminates 40% Web object loads on the critical path

Recent work to reduce PLT

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

1. Tools to study page load process
2. Better use of the network
   - HTTP/2 effort based on SPDY
3. Better content structures
   - mod_pagespeed server extension

189

HTTP/1.1

- Opens too many TCP connections
- Lacks control over the transfer of Web objects
- Single TCP segment cannot carry more than one HTTP request or response

SPDY is proposed to address the issues

190

SPDY

- Multiplexes HTTP data into a single TCP conn.
- Prioritizes Web objects
- Allows servers to initiate Web object transfers
- Compresses headers, not only payloads

Unclear how much SPDY helps

- SPDY whitepaper from Google
  - SPDY helps 27% - 60%
- Other studies from Microsoft, Akamai, and Wprof
  - SPDY sometimes helps and sometimes hurts
  - Overall, SPDY helps < 10%

Challenges in understanding SPDY

- Many factors external to SPDY affect SPDY
  – E.g., network parameters, TCP settings
- Page load time varies significantly
- Dependencies between network and computation significantly affect page loads

Approach

- Isolate the contributing factors to SPDY

Synthetic Web pages → Real object sizes and no. → Real objects, depend-encies, and computation

- Control variability in page loads by
  – Using a custom emulator instead of browsers
  – Experimenting in a controlled network

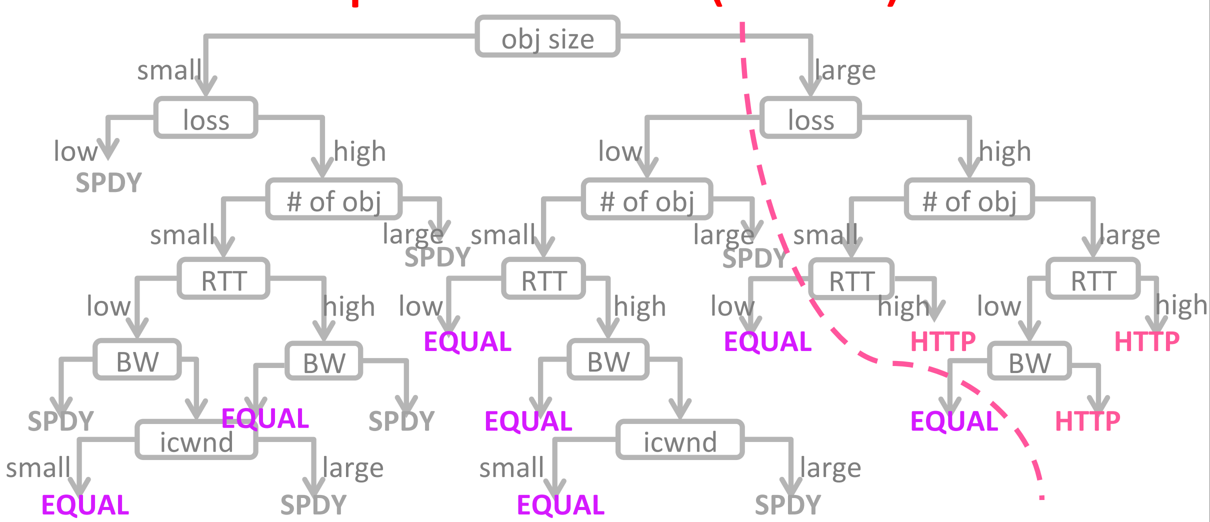
Experiment 1

- Make HTTP requests; artificial web pages

Factors	Range
RTT	20ms, 100ms, 200ms
Bandwidth	1Mbps, 10Mbps
Packet loss rate	0, .5%, 1%, 2%

195

Experiment 1 (cont.)



196

Experiment 2

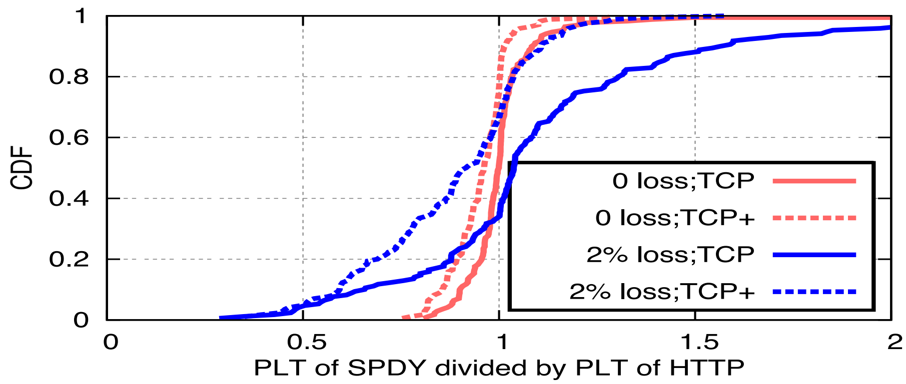
- Make HTTP requests
- Consider real Web object sizes from the top 200 Web pages

SPDY helps quite a bit

Experiment 3

- Emulate the page load process with EpLoad
  - Goal: controls the variability of page loads
- Epload records/replays page loads
  - Recorder: capture the dependency graph
  - Replayer: make network requests while simulating the computation portions

Experiment 3 (cont.)



RTT=20ms, Bandwidth=10Mbps

SPDY helps marginally

199

How to improve SPDY?

- Develop better policies for prioritization and server push
- Leverage information from dependency graphs
  - Web objects that are closer to the root should be assigned a higher priority or be pushed earlier

200

Findings with SPDY policies

- Prioritization helps little
  - Priorities are embedded in the dependency graph
  - Prioritization doesn't break the dependency graph
- Server push can help quite a bit
  - Server push breaks the dependency graph

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

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)