CIS 5530: Networked Systems

HTTP Optimizations

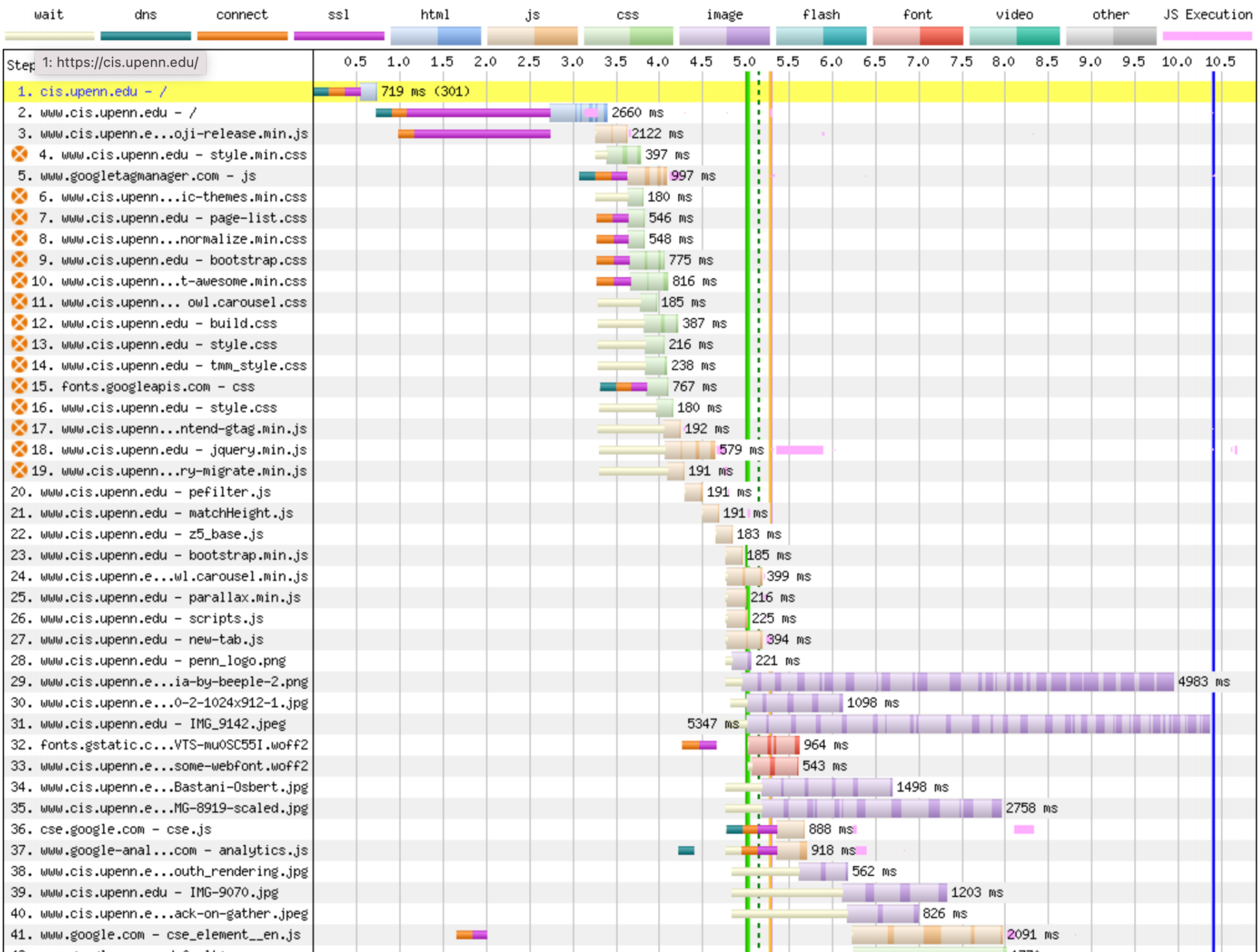
November 27, 2023

Review

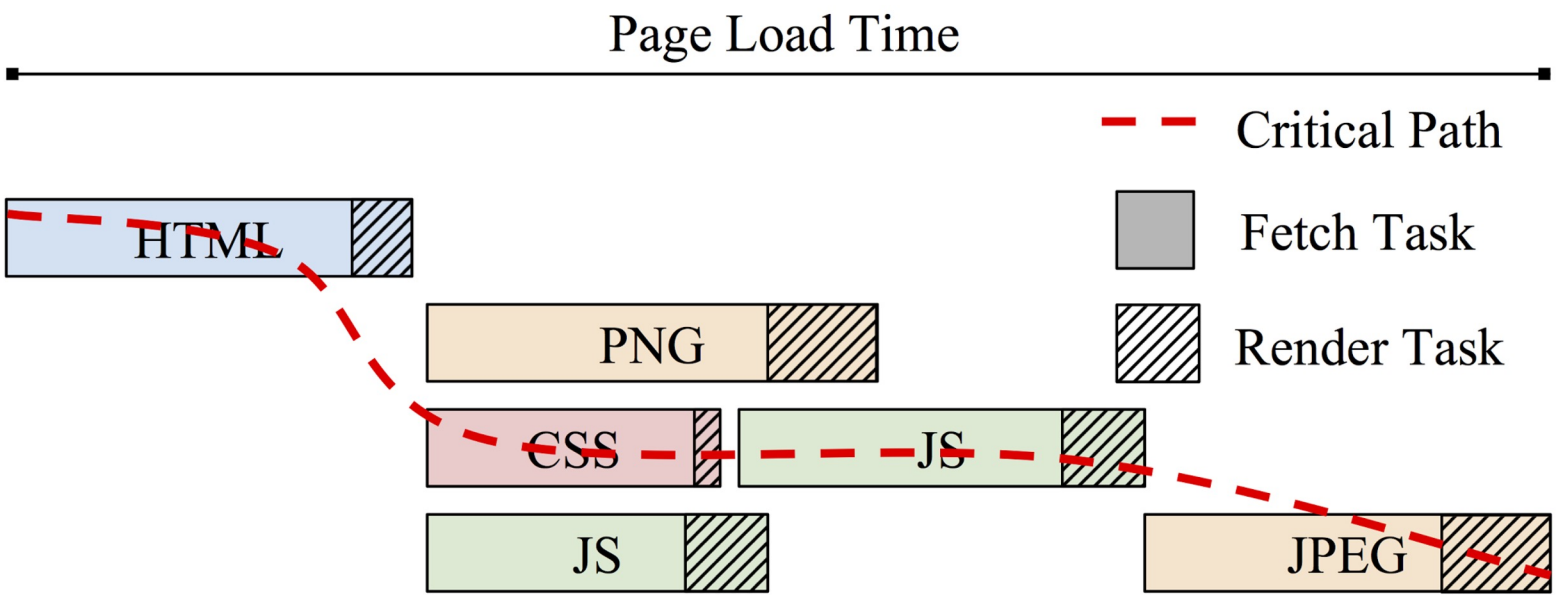
- Name three optimizations in typical HTTP/1.1 implementations

Agenda

- HTTP
  - Fetching web pages
  - Optimizations NEXT


wait
dns
connect
ssl
html
js
css
image
flash
font
video
other
JS Execution
Step
1: https://cis.upenn.edu/
0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5 9.0 9.5 10.0 10.5
1. cis.upenn.edu - / 719 ms (301)
2. www.cis.upenn.edu - / 2660 ms
3. www.cis.upenn.e...oji-release.min.js 2122 ms
4. www.cis.upenn.edu - style.min.css 397 ms
5. www.googletagmanager.com - js 997 ms
6. www.cis.upenn...ic-themes.min.css 180 ms
7. www.cis.upenn.edu - page-list.css 546 ms
8. www.cis.upenn...normalize.min.css 548 ms
9. www.cis.upenn.edu - bootstrap.css 775 ms
10. www.cis.upenn...t-awesome.min.css 816 ms
11. www.cis.upenn... owl.carousel.css 185 ms
12. www.cis.upenn.edu - build.css 387 ms
13. www.cis.upenn.edu - style.css 216 ms
14. www.cis.upenn.edu - tmm_style.css 238 ms
15. fonts.googleapis.com - css 767 ms
16. www.cis.upenn.edu - style.css 180 ms
17. www.cis.upenn...ntend-gtag.min.js 192 ms
18. www.cis.upenn.edu - jquery.min.js 579 ms
19. www.cis.upenn...ry-migrate.min.js 191 ms
20. www.cis.upenn.edu - pefilter.js 191 ms
21. www.cis.upenn.edu - matchHeight.js 191 ms
22. www.cis.upenn.edu - z5_base.js 183 ms
23. www.cis.upenn.edu - bootstrap.min.js 185 ms
24. www.cis.upenn.e...wl.carousel.min.js 399 ms
25. www.cis.upenn.edu - parallax.min.js 216 ms
26. www.cis.upenn.edu - scripts.js 225 ms
27. www.cis.upenn.edu - new-tab.js 394 ms
28. www.cis.upenn.edu - penn_logo.png 221 ms
29. www.cis.upenn.e...ia-by-beeple-2.png 4983 ms
30. www.cis.upenn.e...0-2-1024x912-1.jpg 1098 ms
31. www.cis.upenn.edu - IMG_9142.jpeg 5347 ms
32. fonts.gstatic.c...VTS-mu0SC55I.woff2 964 ms
33. www.cis.upenn.e...some-webfont.woff2 543 ms
34. www.cis.upenn.e...Bastani-Osbert.jpg 1498 ms
35. www.cis.upenn.e...MG-8919-scaled.jpg 2758 ms
36. cse.google.com - cse.js 888 ms
37. www.google-anal...com - analytics.js 918 ms
38. www.cis.upenn.e...outh_rendering.jpg 562 ms
39. www.cis.upenn.edu - IMG-9070.jpg 1203 ms
40. www.cis.upenn.e...ack-on-gather.jpeg 826 ms
41. www.google.com - cse_element__en.js 2091 ms

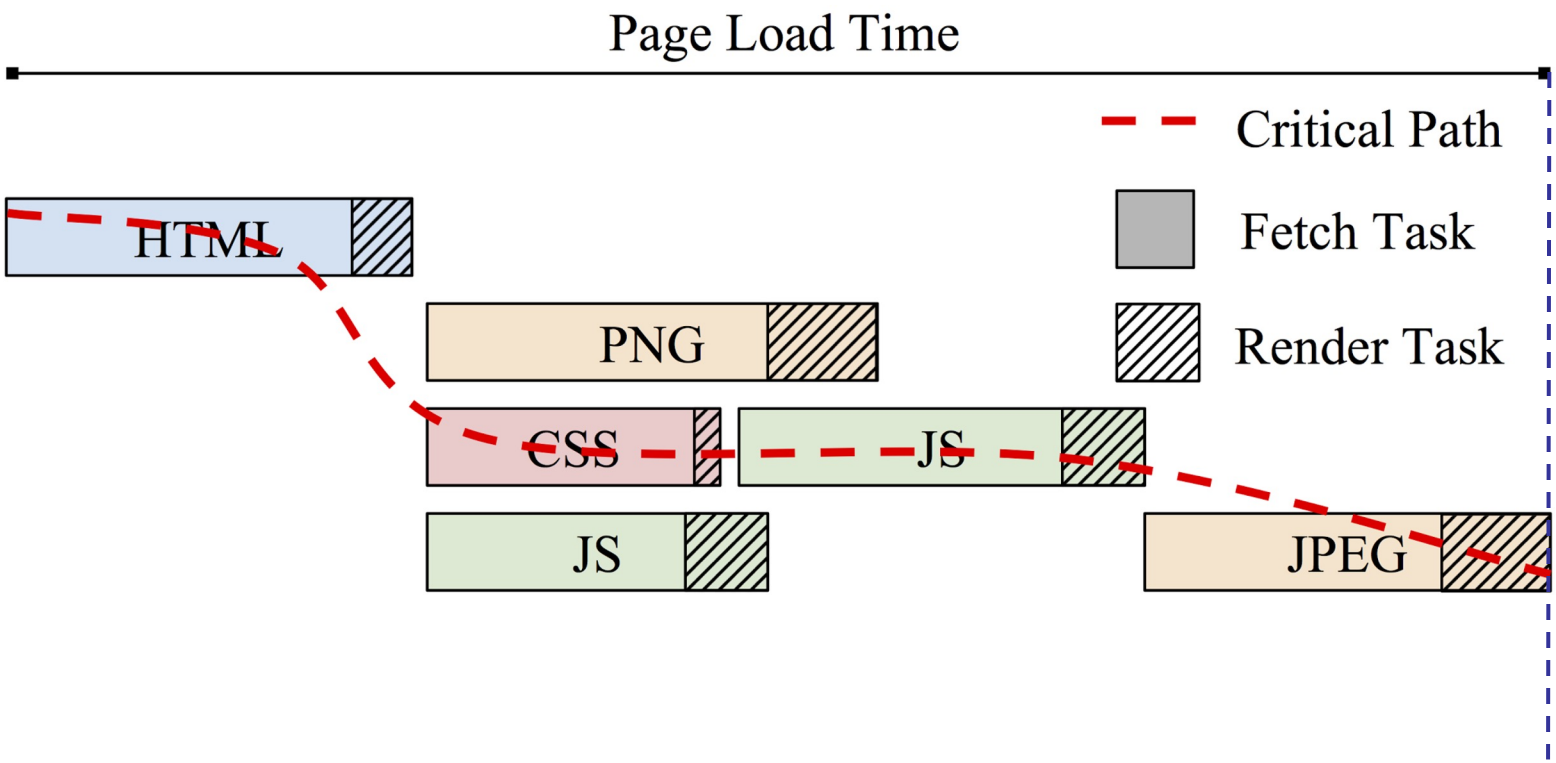
Background - Critical Path



Critical Path: the longest chain of dependent browser tasks

Fetch Delay: Network Delay
Render Delay: Computational Delay

Background - Page Load Time (PLT)



Order matters!

How can we reduce PLT?

Agenda

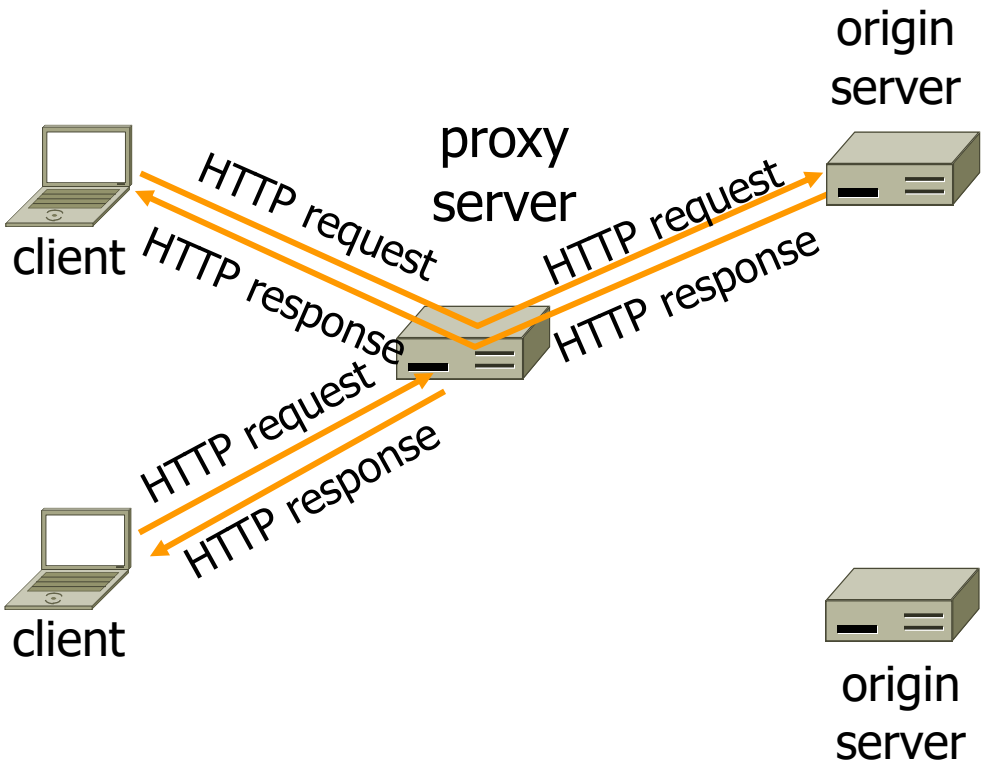
- HTTP ✓
  - Fetching web pages ✓
- HTTP Optimizations ✓
  - CDNs ← NEXT
  - HTTP/2

Proxies

- Computer that acts a broker between client and server
  - Speaks to server on client's behalf

HTTP Proxies

- Accept requests from multiple clients
- Takes request and reissues it to server
- Takes response and forwards to client

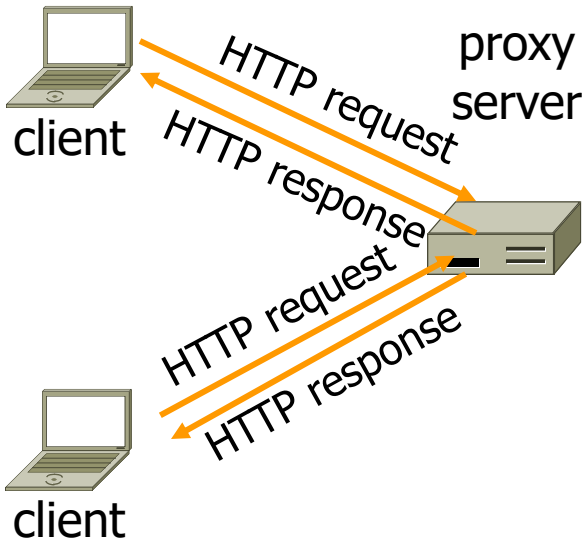


Types of Proxies

- Explicit
  - Requires configuring browser
- Implicit
  - Service provider deploys an "on path" proxy
  - … that intercepts and handles Web requests
- Also, forward and reverse

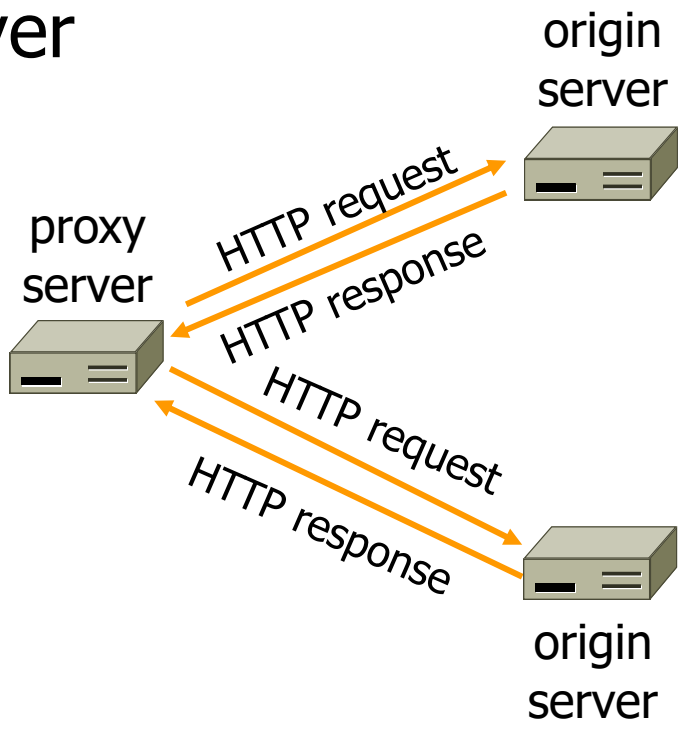
Forward Proxy

- Proxy on behalf of the client
- Usually under administrative control of client-side AS



Reverse Proxy

- Cache on behalf of the server
- Generally implicit

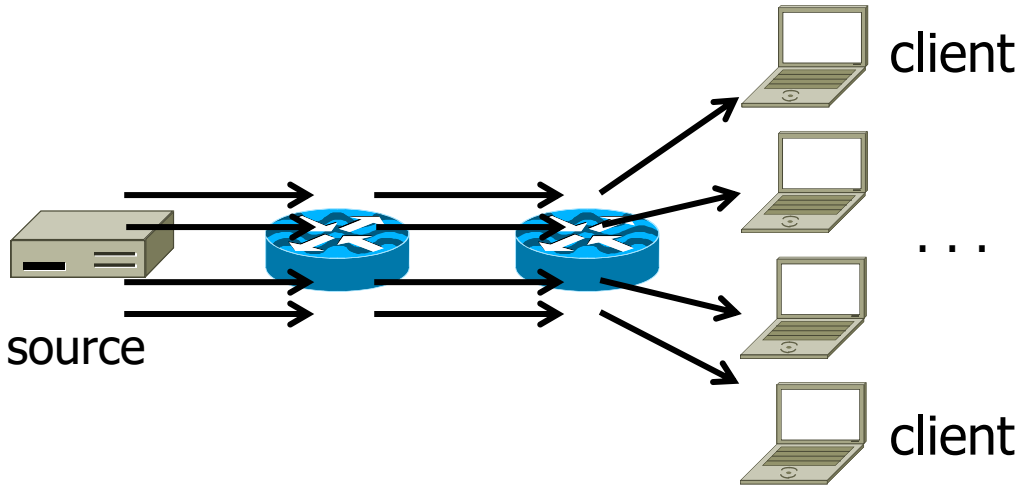


Examples of Proxies

- Forward:
  - Privacy – Hide your true IP
  - Firewalls – limit access to the Internet, secure the proxy
  - ISP caching – reduce traffic exiting the AS
- Reverse:
  - CDNs – reduce traffic exiting the AS
  - Load balancers

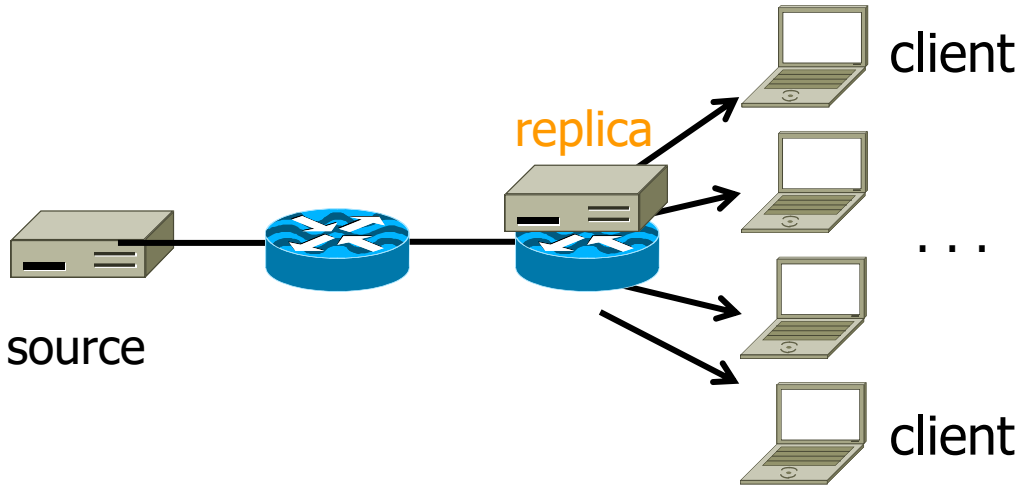
Before CDNs

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



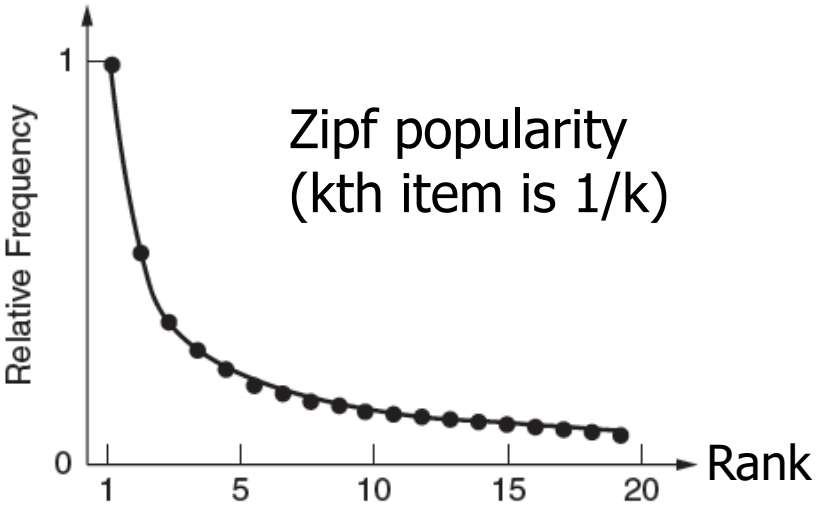
After CDNs

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



Popularity of Content

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



George Zipf (1902-1950)

Source: Wikipedia

Content Distribution Networks (CDN)

- Caching and replication as a service
- Large-scale distributed storage infrastructure (usually) administered by one entity
  - e.g., Akamai has servers in 20,000+ locations
- Combination of caching and replication
  - Pull: Direct result of clients'requests (caching)
  - Push: Expectation of high access rate (replication)
- Can do some processing to handle dynamic webpage content

CDNs (e.g., Akamai)

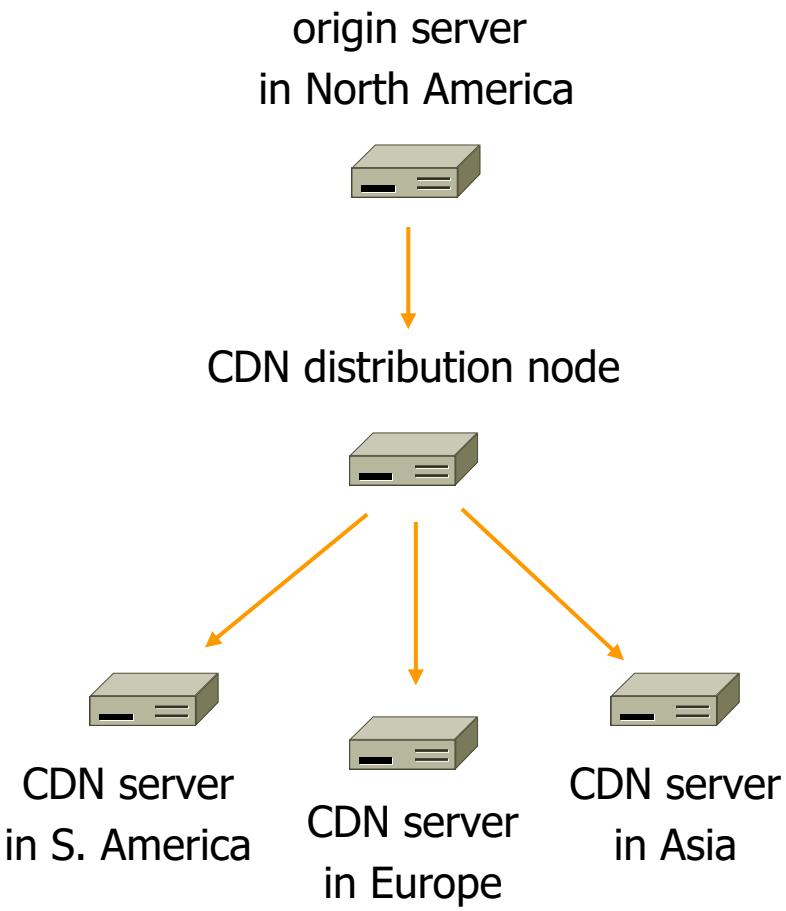
The Akamai EdgePlatform:

240,000+ Servers	2200+ POPs	1600+ Networks	800+ Cities	130 Countries

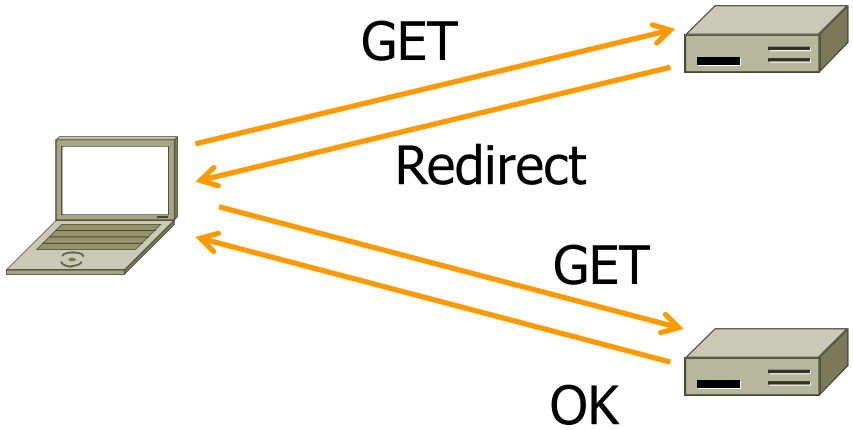
Resulting in traffic of:
30 terabits / sec
3+ trillion hits / day
Responsible for 15-30% of all web traffic

Server Selection Policies

- Live server
  - For availability
- Lowest load
  - To balance load across the servers
- Closest
  - Nearest geographically, or in round-trip time
- Best performance
  - Throughput, latency, …
- Cheapest bandwidth, electricity, …

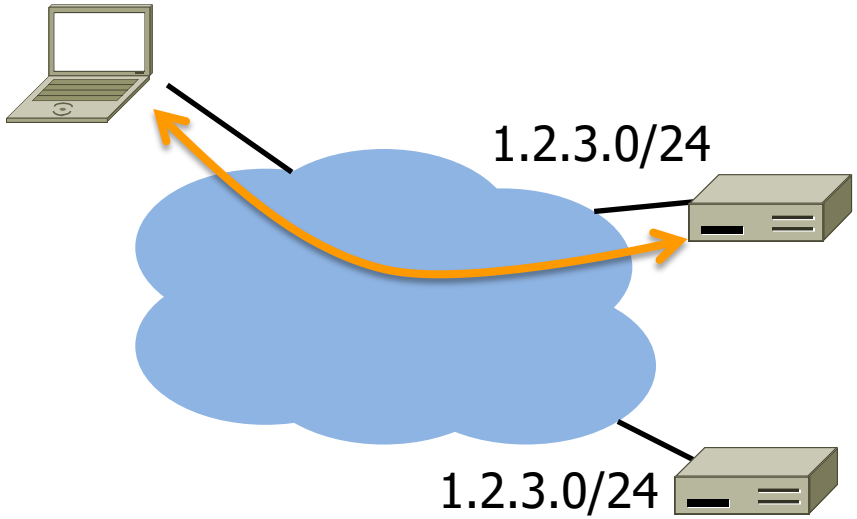


Mechanism: HTTP Redirection



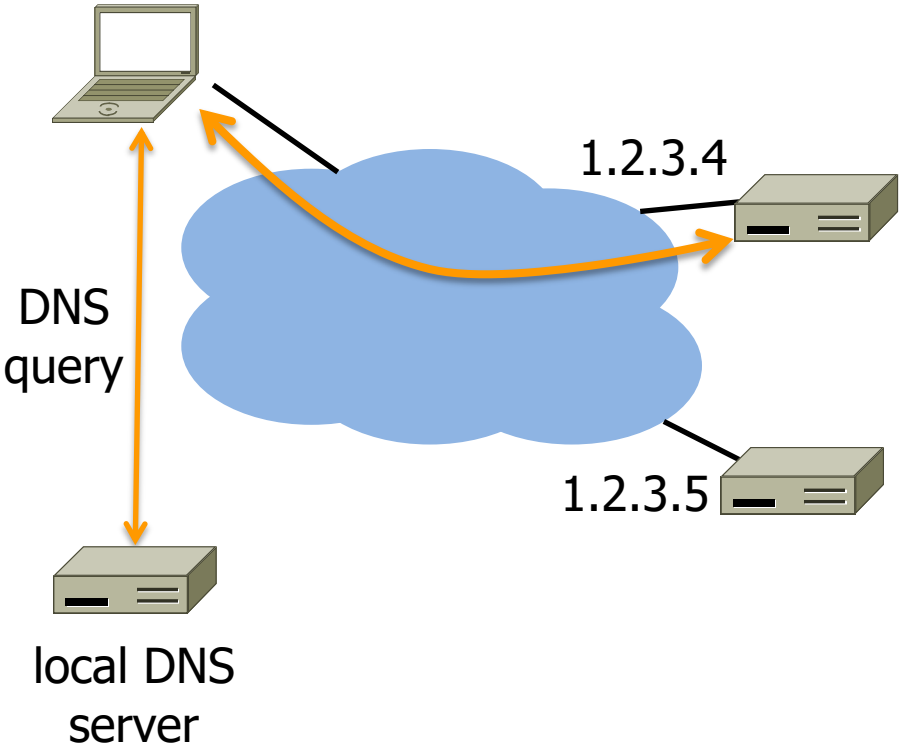
- Advantages
  - Fine-grain control
  - Selection based on client IP address
- Disadvantages
  - Extra round-trips for TCP connection to server
  - Overhead on the server

Mechanism: Anycast Routing



- Advantages
  - No extra round trips
  - Route to nearby server
- Disadvantages
  - Does not consider network or server load
  - Different packets may go to different servers
  - Used only for simple request-response apps

Mechanism: DNS



- Advantages
  - Avoid TCP set-up delay
  - DNS caching reduces overhead
  - Relatively fine control
- Disadvantage
  - Based on IP address of local DNS server
  - "Hidden load" effect
  - DNS TTL limits adaptation

Agenda

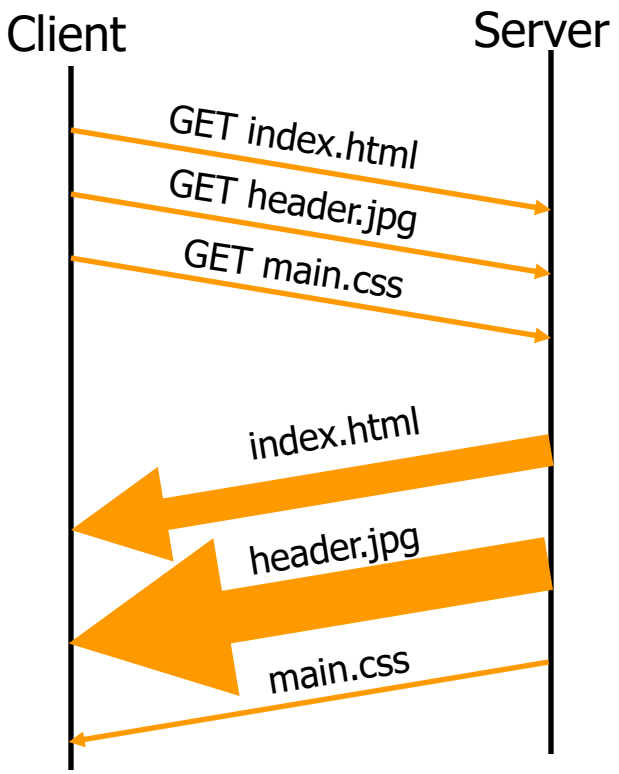
- Fetching web pages ✓
- HTTP Optimizations ✓
  - CDNs ✓
  - HTTP/2 ← NEXT
- Video Streaming

Problems with HTTP/1.1

- Pipelining created "Head of Line" blocking
  - "slow" objects delay later requests

Problems with HTTP/1.1

- Pipelining created "Head of Line" blocking
  - "slow" objects delay later requests



https://developers.google.com/web/fundamentals/performance/http2

Problems with HTTP/1.1

- Pipelining created "Head of Line" blocking
  - "slow" objects delay later requests
- Networks got faster
  - Multiple TCP connections to achieve parallel transfers
- HTTP headers became bloated
  - Cost higher for small objects
- Webpage processing got complicated
  - Priorities: Javascript versus images
  - Dependencies: Extra RTTs for "dependent" objects

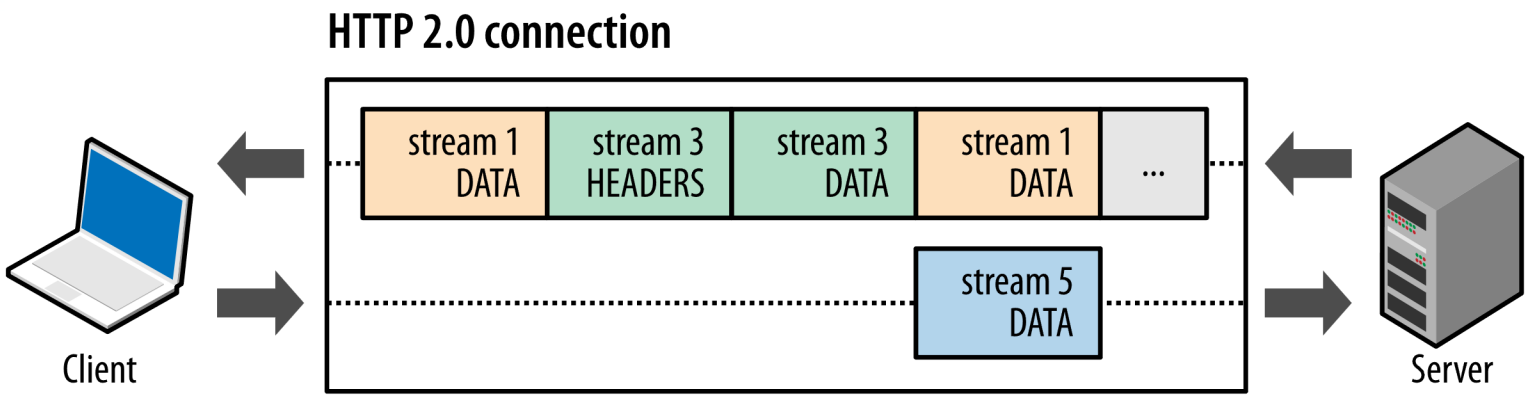
HTTP/2

- Started with Google's SPDY protocol
  - Announced 2009
  - Wide deployment by 2012
  - RFC published 2015
- Semantics (header options, etc) similar to HTTP/1.1
- All about performance
  - Single connection
  - Binary instead of ASCII

HTTP/2 Key Features

1. Binary framing
2. Multiplexing and prioritization
3. Compressed headers
4. Server push

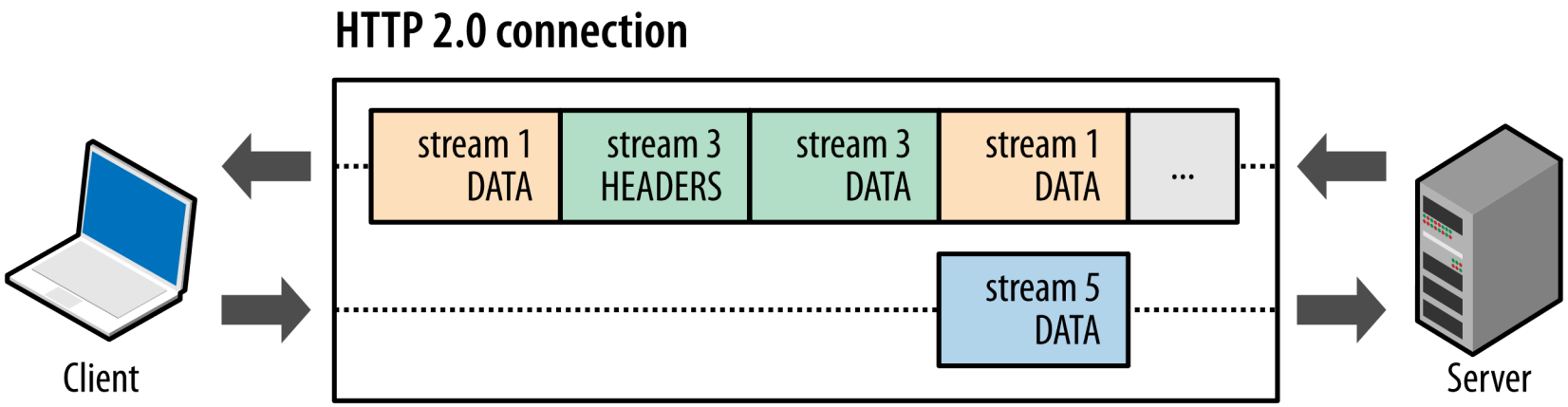
1. HTTP/2 Framing



Bit	+0..7	+8..15	+16..23	+24..31
0	Length			Type
32	Flags			
40	R	Stream Identifier		
...	Frame Payload			

2. Multiplexing and Prioritization

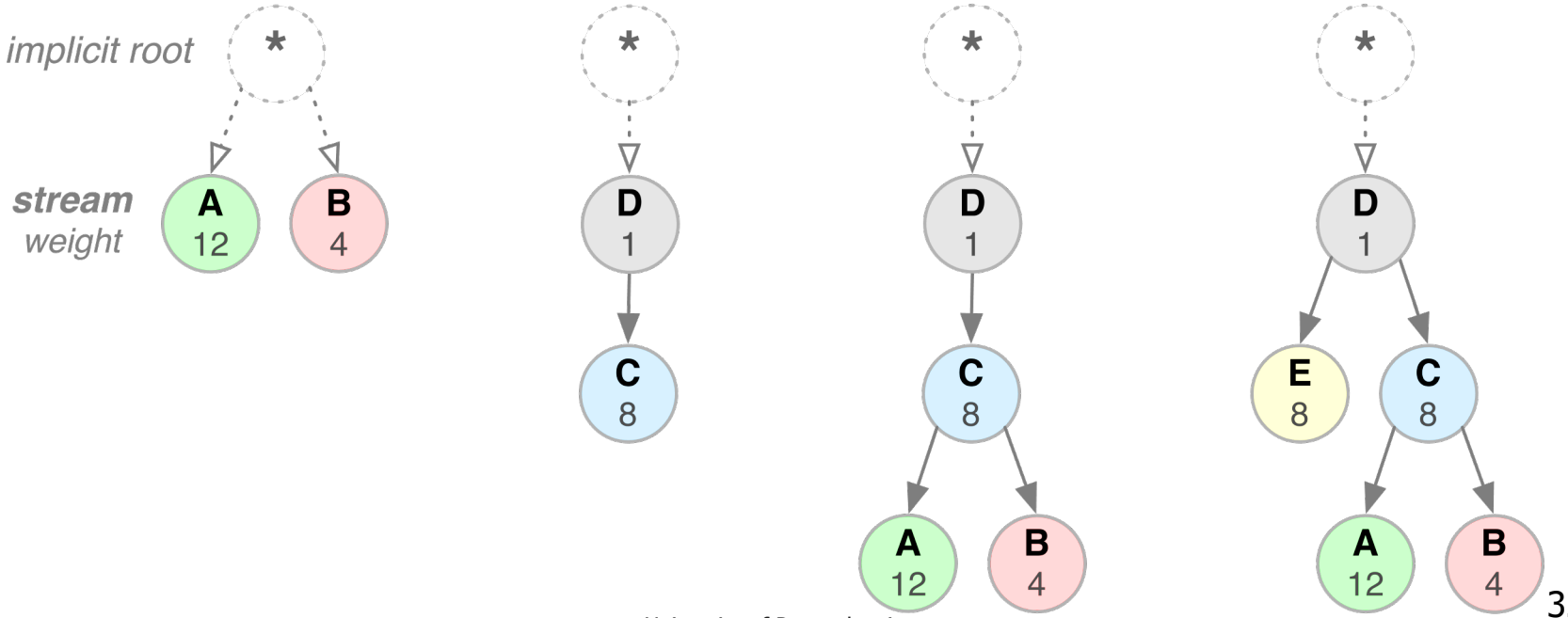
- Multiplexing
  - Interleave requests over a single TCP connection
  - Why is that okay?



https://developers.google.com/web/fundamentals/performance/http2

2. Multiplexing and Prioritization

- Multiplexing
  - Interleave requests over a single TCP connection
- Prioritization
  - Interleave responses based on priority, dependencies

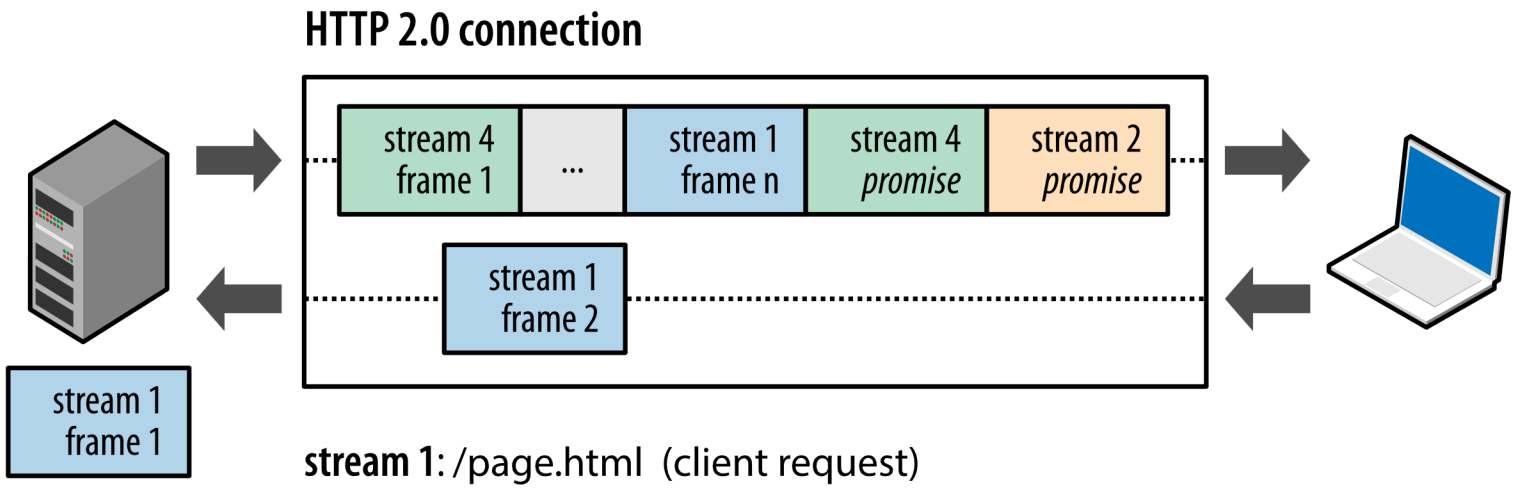


3. Compressed headers

- Significant redundancy
  - User-Agent
  - Cookie headers
- Compress relatively effectively
- Out-of-band compression dictionary

4. Server push

- No client request needed
- Server uses knowledge of resources



stream 1: /page.html (client request)
stream 2: /script.js (push promise)
stream 4: /style.css (push promise)

HTTP/3: QUIC

- Proposed standard (as of June 2022)
- Similar goals:
  - Faster startup
  - Better TCP effects
  - Elimination of HoL blocking
- TCP + TLS + SPDY over UDP
- Faster connection establishment than TLS/TCP
  - 0-RTT usually, 1-RTT sometimes
- Deals better with packet loss than TCP
- Has Stream-level and Connection-level Flow Control
- FEC recovery, Multipath