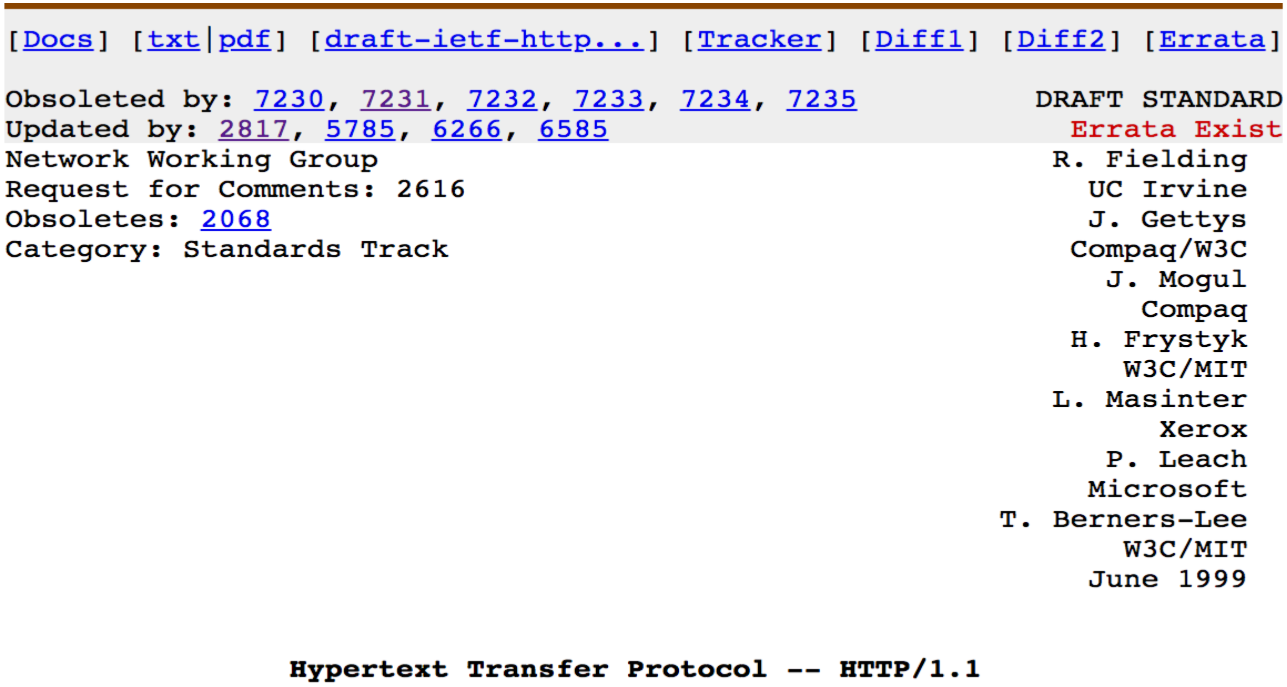


```
[Docs] [txt|pdf] [draft-ietf-http...] [Tracker] [Diff1] [Diff2] [Errata]

Obsoleted by: 7230, 7231, 7232, 7233, 7234, 7235          DRAFT STANDARD
Updated by: 2817, 5785, 6266, 6585                          Errata Exist
Network Working Group                                      R. Fielding
Request for Comments: 2616                                   UC Irvine
Obsoletes: 2068                                              J. Gettys
Category: Standards Track                                   Compaq/W3C
                                                              J. Mogul
                                                                Compaq
                                                           H. Frystyk
                                                               W3C/MIT
                                                          L. Masinter
                                                                 Xerox
                                                              P. Leach
                                                             Microsoft
                                                        T. Berners-Lee
                                                               W3C/MIT
                                                             June 1999


                Hypertext Transfer Protocol -- HTTP/1.1
```

HTTP and the Web

Kyle Jamieson

Lecture 15

COS 461: Computer Networks

Today

1. HTTP basics: headers, requests, responses

2. Web proxies; web caches

3. Web performance optimization

Two Forms of Header Formats

- Fixed: Every field (type, length) defined
  - Fast parsing (good for hardware implementations)
  - Not human readable
  - Fairly static (IPv6 ~20 years to deploy)
  - E.g., Ethernet, IP, TCP headers
- Today: **Variable length** headers
  - Slower parsing (hard to implement in hardware)
  - Human readable
  - Extensible
  - E.g., HTTP (Web), SMTP (Email), XML

HTTP Basics (Overview)

- HTTP over bidirectional byte stream (e.g. TCP)
- Interaction
  - Client looks up host (DNS)
  - Client sends ***request*** message to server
  - Server ***response*** message contains data or error
  - Requests & responses are encoded in text
- HTTP protocol itself is ***Stateless***
  - HTTP maintains no info about past client requests
  - "Cookies" allow server to identify client and associate requests into a client session

HTTP Request

- Request line
  - Method
    - GET – return URI
    - HEAD – return headers only of GET response
    - POST – send data to the server (forms, etc.)
  - URL (relative)
    - E.g., /index.html
  - HTTP version

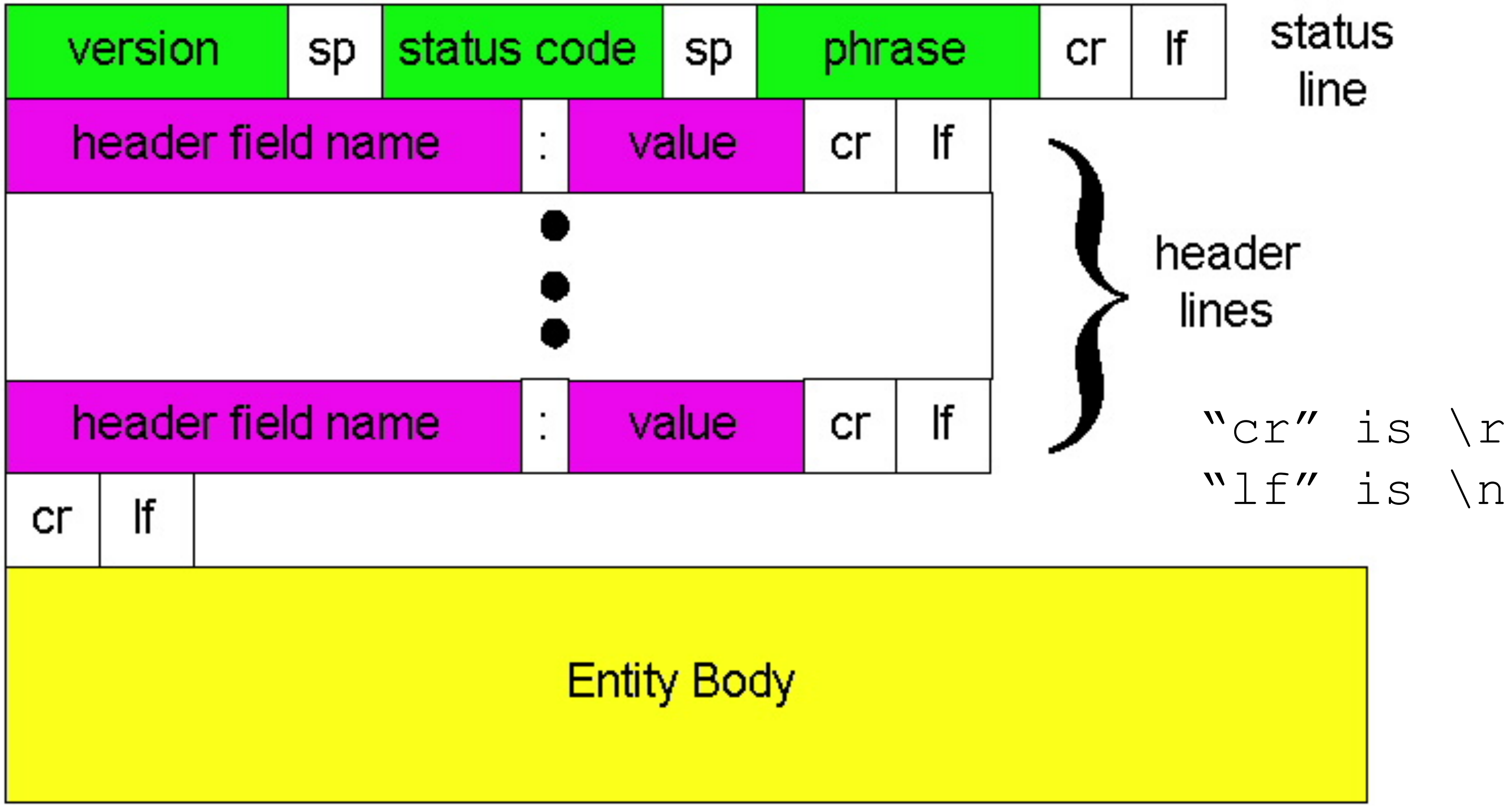
HTTP Request (cont.)

- Request headers
  - Variable length, human-readable
  - Uses:
    - Authorization – authentication info
    - Acceptable document types/encodings
    - From – user email
    - If-Modified-Since
    - Referrer – what caused this page to be requested
    - User-Agent – client software
- Blank-line
- Body

HTTP Request Example

```
GET /index.html HTTP/1.1
Host: www.example.com
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)
Connection: Keep-Alive
```

HTTP Response



version sp status code sp phrase cr lf

status line

header field name : value cr lf

header field name : value cr lf

header lines

cr lf

Entity Body

"cr" is \r
"lf" is \n

HTTP Response

- Status-line
  - HTTP version (now “1.1”)
  - 3 digit response code
    - 1XX – informational
    - 2XX – success
      - 200 OK
    - 3XX – redirection
      - 301 Moved Permanently
      - 303 Moved Temporarily
      - 304 Not Modified
    - 4XX – client error
      - 404 Not Found
    - 5XX – server error
      - 505 HTTP Version Not Supported
  - Reason phrase

HTTP Response (cont.)

- Headers
  - Variable length, human-readable
  - Uses:
    - Location – for redirection
    - Server – server software
    - WWW-Authenticate – request for authentication
    - Allow – list of methods supported (get, head, etc)
    - Content-Encoding – E.g x-gzip
    - Content-Length
    - Content-Type
    - Expires (caching)
    - Last-Modified (caching)
- Blank-line
- Body

HTTP Response Example

```
HTTP/1.1 200 OK
Date: Tue, 27 Mar 2001 03:49:38 GMT
Server: Apache/1.3.14 (Unix)  (Red-Hat/Linux) mod_ssl/2.7.1
     OpenSSL/0.9.5a DAV/1.0.2 PHP/4.0.1pl2 mod_perl/1.24
Last-Modified: Mon, 29 Jan 2001 17:54:18 GMT
Accept-Ranges: bytes
Content-Length: 4333
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html
…..
```

How to Mark End of Message?

- Close connection
  - Only server can do this
  - One request per TCP connection. Hurts performance.
- Content-Length
  - Must know size of transfer in advance
- No body content. Double CRLF marks end
  - E.g., 304 never have body content
- Transfer-Encoding: chunked (HTTP/1.1)
  - After headers, each chunk is content length in hex, CRLF, then body. Final chunk is length 0.

Example: Chunked Encoding

```
HTTP/1.1 200 OK <CRLF>
Transfer-Encoding: chunked <CRLF>
<CRLF>
25 <CRLF>
This is the data in the first chunk <CRLF>
1A <CRLF>
and this is the second one <CRLF>
0 <CRLF>
```

- Especially useful for dynamically-generated content, as length is not a priori known
  - Server would otherwise need to cache data until done generating, and then go back and fill-in length header before transmitting

Web Proxies

HTTP Caching

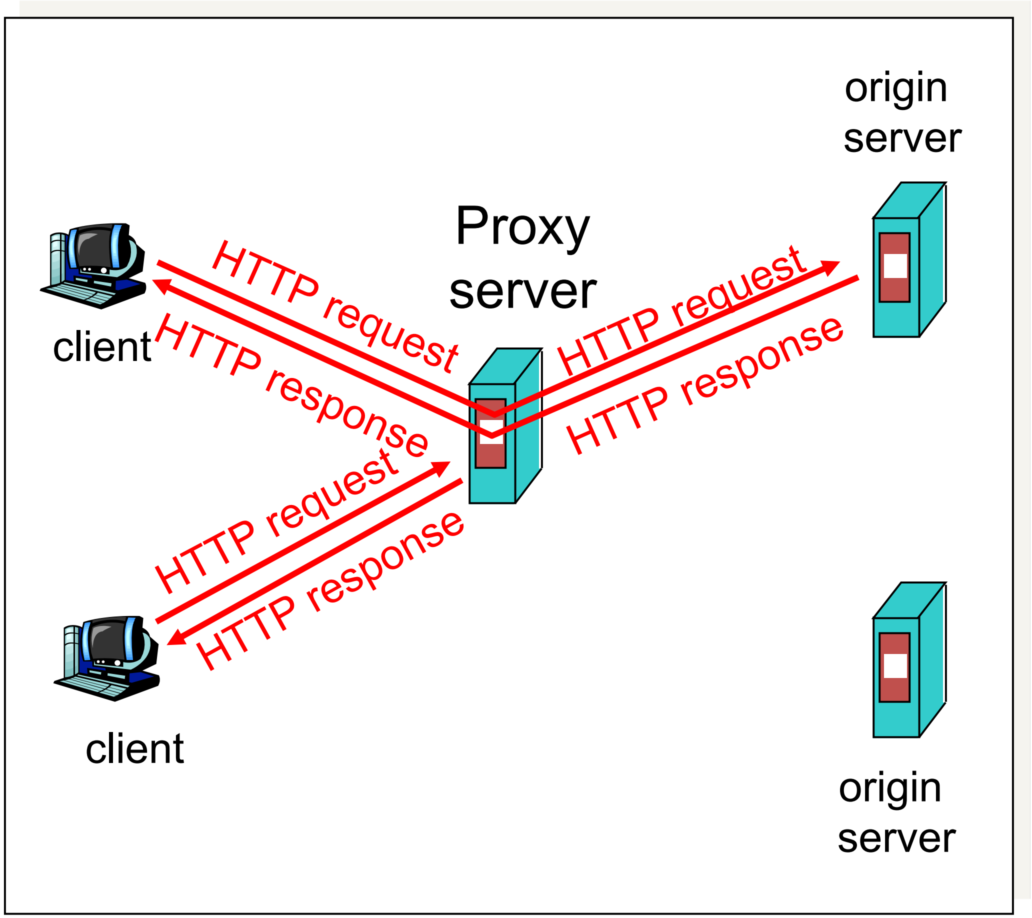
Proxies

- End host that acts a broker between client and server
  - Speaks to server on client’s behalf

- Why?
  - Privacy
  - Content filtering
  - Caching!!!

Proxies (Cont.)

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



HTTP Caching

- Why cache?
  - Lot of objects don’t change (images, js, css)
  - Reduce # of client connections
  - Reduce server load
  - Reduce overall network traffic; save $$$

Caching is Hard

- Significant fraction (>50%?) of distinct HTTP objects may be uncacheable
  - Dynamic data: Stock prices, scores, web cams
  - CGI scripts: results based on passed parameters
  - Cookies: results may be based on passed data
  - SSL: encrypted data is not cacheable
  - Advertising / analytics: owner wants to measure # hits
    - Random strings in content to ensure unique counting

- Yet significant fraction of HTTP bytes are cacheable
  - Images, video, CSS pages, etc.

- Want to limit staleness of cached objects

How long should the client cache for?

- Clients (and proxies) cache documents
  - When should origin be checked for changes?
  - Every time? Every session? Date?

- HTTP includes caching information in headers
  - HTTP 0.9/1.0 used: “Expires: <date>”; “Pragma: no-cache”
  - HTTP/1.1 has “Cache-Control”
    - “No-Cache”, “Max-age: <seconds>”
    - “ETag: <opaque value>

Why the changes between 1.0 and 1.1?

- Timestamps
  - Server hints when an object “Expires” (Expires: xxx)
  - Server provides last modified date, client can check if that’s still valid

- Problems
  - Client and server might not have synchronized clocks
  - Server replicas might not have synchronized clocks
  - Max-age solves this: relative seconds, not abs time

What if cache expires?

- Store past expiry time (if room in cache)
- Upon client request, cache revalidates with server

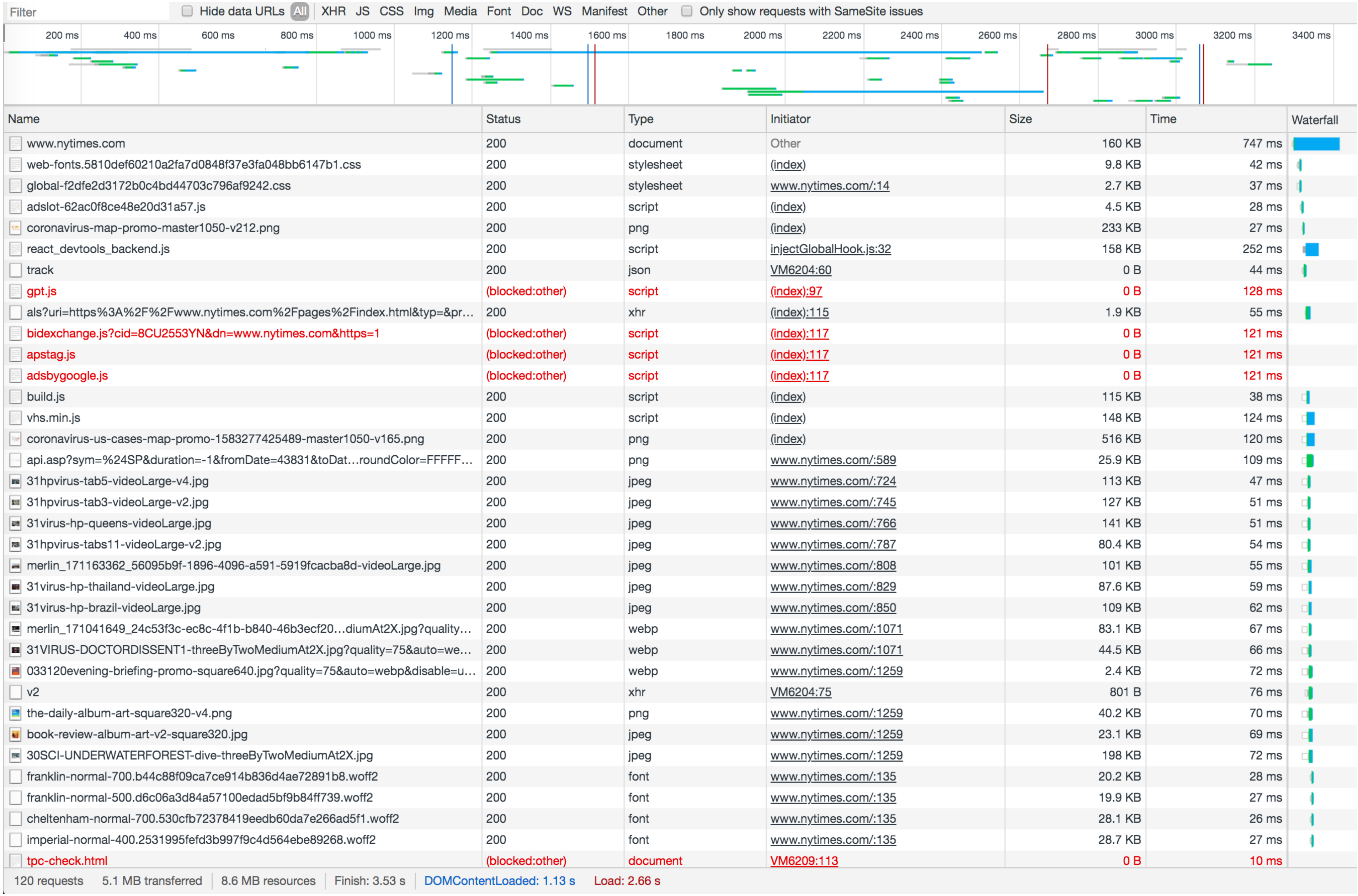
```
GET / HTTP/1.1
Accept-Language: en-us
If-Modified-Since: Mon, 29 Jan 2001 17:54:18 GMT
Host: www.example.com
Connection: Keep-Alive
```

```
HTTP/1.1 304 Not Modified
Date: Tue, 27 Mar 2001 03:50:51 GMT
Connection: Keep-Alive
```

HTTP xfer = single object

Web pages = many objects

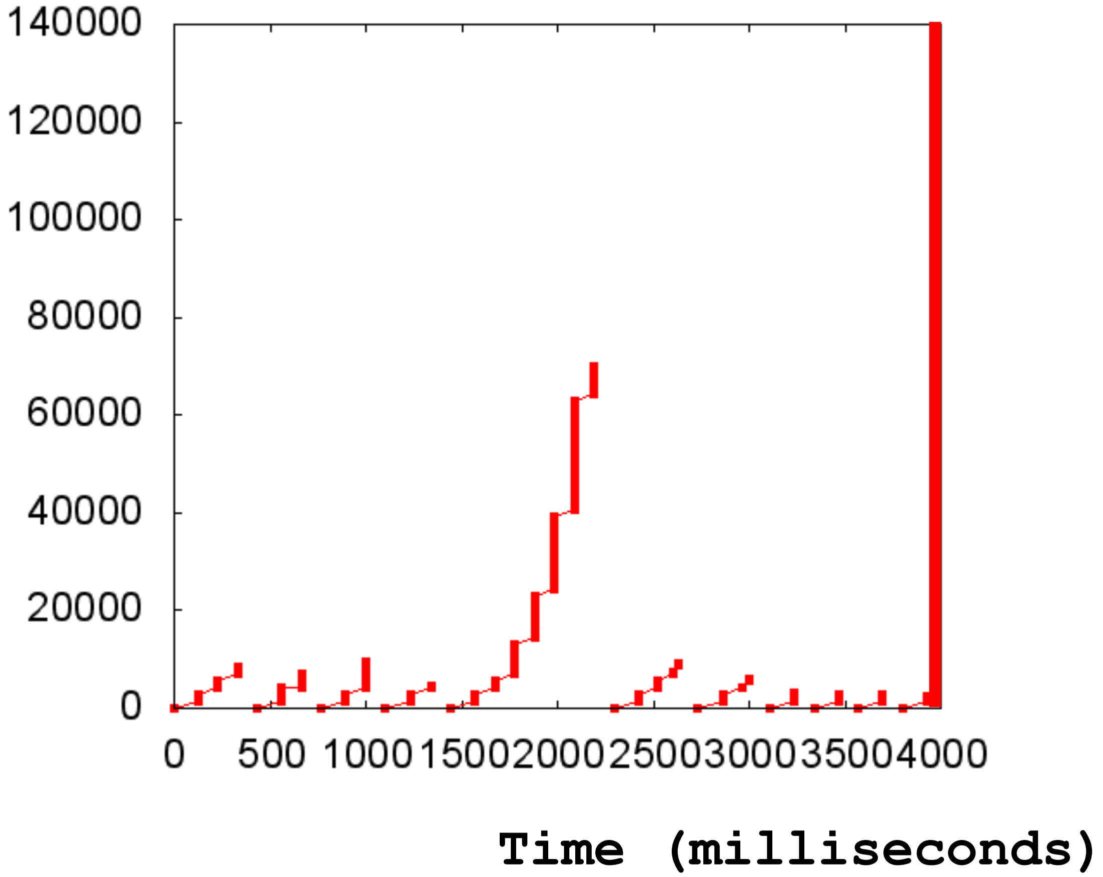
nytimes.com



Name	Status	Type	Initiator	Size	Time	Waterfall
www.nytimes.com	200	document	Other	160 KB	747 ms	
web-fonts.5810def60210a2fa7d0848f37e3fa048bb6147b1.css	200	stylesheet	(index)	9.8 KB	42 ms	
global-f2dfe2d3172b0c4bd44703c796af9242.css	200	stylesheet	www.nytimes.com/:14	2.7 KB	37 ms	
adslot-62ac0f8ce48e20d31a57.js	200	script	(index)	4.5 KB	28 ms	
coronavirus-map-promo-master1050-v212.png	200	png	(index)	233 KB	27 ms	
react_devtools_backend.js	200	script	injectGlobalHook.js:32	158 KB	252 ms	
track	200	json	VM6204:60	0 B	44 ms	
gpt.js	(blocked:other)	script	(index):97	0 B	128 ms	
als?uri=https%3A%2F%2Fwww.nytimes.com%2Fpages%2Findex.html&typ=&pr…	200	xhr	(index):115	1.9 KB	55 ms	
bidexchange.js?cid=8CU2553YN&dn=www.nytimes.com&https=1	(blocked:other)	script	(index):117	0 B	121 ms	
apstag.js	(blocked:other)	script	(index):117	0 B	121 ms	
adsbygoogle.js	(blocked:other)	script	(index):117	0 B	121 ms	
build.js	200	script	(index)	115 KB	38 ms	
vhs.min.js	200	script	(index)	148 KB	124 ms	
coronavirus-us-cases-map-promo-1583277425489-master1050-v165.png	200	png	(index)	516 KB	120 ms	
api.asp?sym=%24SP&duration=-1&fromDate=43831&toDat…roundColor=FFFFF…	200	png	www.nytimes.com/:589	25.9 KB	109 ms	
31hpvirus-tab5-videoLarge-v4.jpg	200	jpeg	www.nytimes.com/:724	113 KB	47 ms	
31hpvirus-tab3-videoLarge-v2.jpg	200	jpeg	www.nytimes.com/:745	127 KB	51 ms	
31virus-hp-queens-videoLarge.jpg	200	jpeg	www.nytimes.com/:766	141 KB	51 ms	
31hpvirus-tabs11-videoLarge-v2.jpg	200	jpeg	www.nytimes.com/:787	80.4 KB	54 ms	
merlin_171163362_56095b9f-1896-4096-a591-5919fcacba8d-videoLarge.jpg	200	jpeg	www.nytimes.com/:808	101 KB	55 ms	
31virus-hp-thailand-videoLarge.jpg	200	jpeg	www.nytimes.com/:829	87.6 KB	59 ms	
31virus-hp-brazil-videoLarge.jpg	200	jpeg	www.nytimes.com/:850	109 KB	62 ms	
merlin_171041649_24c53f3c-ec8c-4f1b-b840-46b3ecf20…diumAt2X.jpg?quality…	200	webp	www.nytimes.com/:1071	83.1 KB	67 ms	
31VIRUS-DOCTORDISSENT1-threeByTwoMediumAt2X.jpg?quality=75&auto=we…	200	webp	www.nytimes.com/:1071	44.5 KB	66 ms	
033120evening-briefing-promo-square640.jpg?quality=75&auto=webp&disable=u…	200	webp	www.nytimes.com/:1259	2.4 KB	72 ms	
v2	200	xhr	VM6204:75	801 B	76 ms	
the-daily-album-art-square320-v4.png	200	png	www.nytimes.com/:1259	40.2 KB	70 ms	
book-review-album-art-v2-square320.jpg	200	jpeg	www.nytimes.com/:1259	23.1 KB	69 ms	
30SCI-UNDERWATERFOREST-dive-threeByTwoMediumAt2X.jpg	200	jpeg	www.nytimes.com/:1259	198 KB	72 ms	
franklin-normal-700.b44c88f09ca7ce914b836d4ae72891b8.woff2	200	font	www.nytimes.com/:135	20.2 KB	28 ms	
franklin-normal-500.d6c06a3d84a57100edad5bf9b84ff739.woff2	200	font	www.nytimes.com/:135	19.9 KB	27 ms	
cheltenham-normal-700.530cfb72378419eedb60da7e266ad5f1.woff2	200	font	www.nytimes.com/:135	28.1 KB	26 ms	
imperial-normal-400.2531995fefd3b997f9c4d564ebe89268.woff2	200	font	www.nytimes.com/:135	28.7 KB	27 ms	
tpc-check.html	(blocked:other)	document	VM6209:113	0 B	10 ms	

120 requests | 5.1 MB transferred | 8.6 MB resources | Finish: 3.53 s | DOMContentLoaded: 1.13 s | Load: 2.66 s

HTTP/1.0 fetching items: Received sequence number plot



Fetch an 8.5 Kbyte page with 10 embedded objects, most < 10 Kbyte

All TCP connections stay in slow start, except for the large object

How to handle many requests?

- Maximize goodput by reusing connections
  - Avoid connection (TCP) setup
  - Avoid TCP slow-start
- Client-server will maintain existing TCP connection for up to K idle seconds

GET / HTTP/1.1

Host: www.example.com

Connection: Keep-Alive

HTTP/1.1 200 OK

Date: Tue, 27 Mar 2001 03:50:51 GMT

Connection: Keep-Alive

Three approaches to multiple requests

Parallel Connections

Conn 1:

- Request 1
- Response 1

Conn 2:

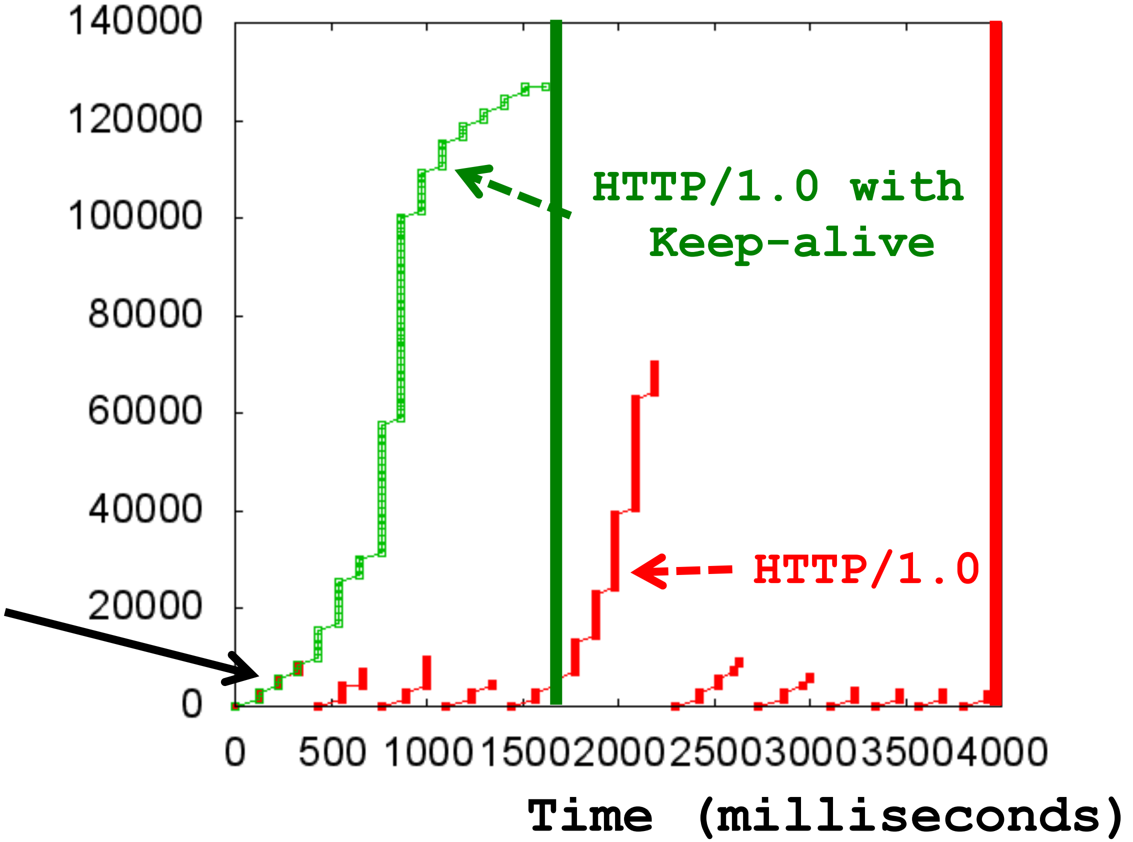
- Request 2
- Response 2

Persistent Connections

Conn 1:

- Request 1
- Response 1
- Request 2
- Response 2
- Request 3
- Response 3

Persistent connections avoid unnecessary slow starts



Fetch an 8.5 Kbyte page with 10 embedded objects, most < 10 Kbyte

Leave TCP connection open after server response, next HTTP request reuses it

Only incur one slow start, but takes an RTT to issue next request

Three approaches to multiple requests

Parallel Connections

Conn 1:

- Request 1
- Response 1

Conn 2:

- Request 2
- Response 2

Persistent Connections

Conn 1:

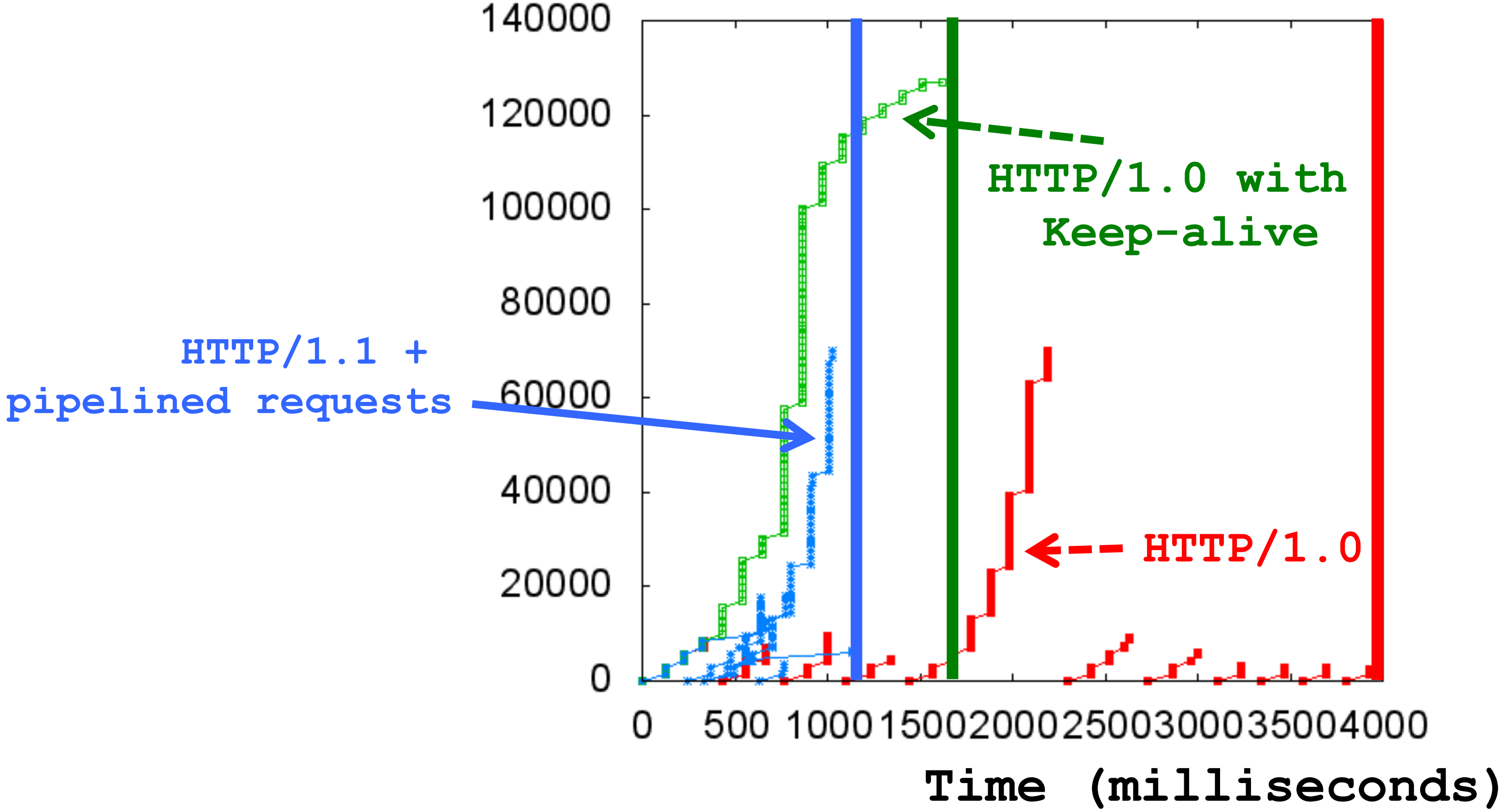
- Request 1
- Response 1
- Request 2
- Response 2
- Request 3
- Response 3

Pipelined Connections

Conn 1:

- Request 1
- Request 2
- Request 3
- Response 1
- Response 2
- Response 3

Pipelined + Parallel Connections overlap RTTs



Fetch an 8.5 Kbyte page with 10 embedded objects, most < 10 Kbyte

Send multiple HTTP requests simultaneously

Overlaps RTTs of all requests

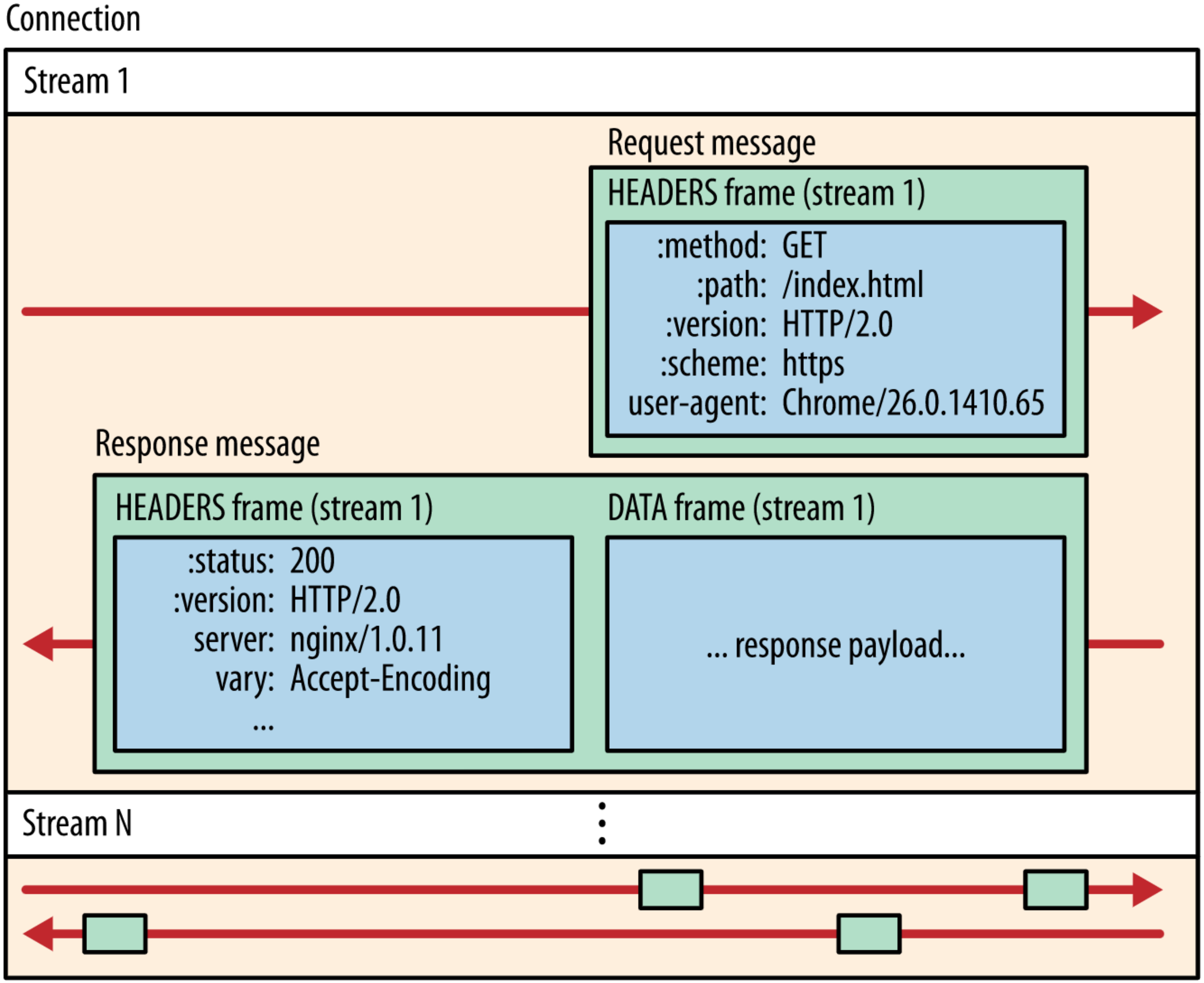
What are challenges with pipelining?

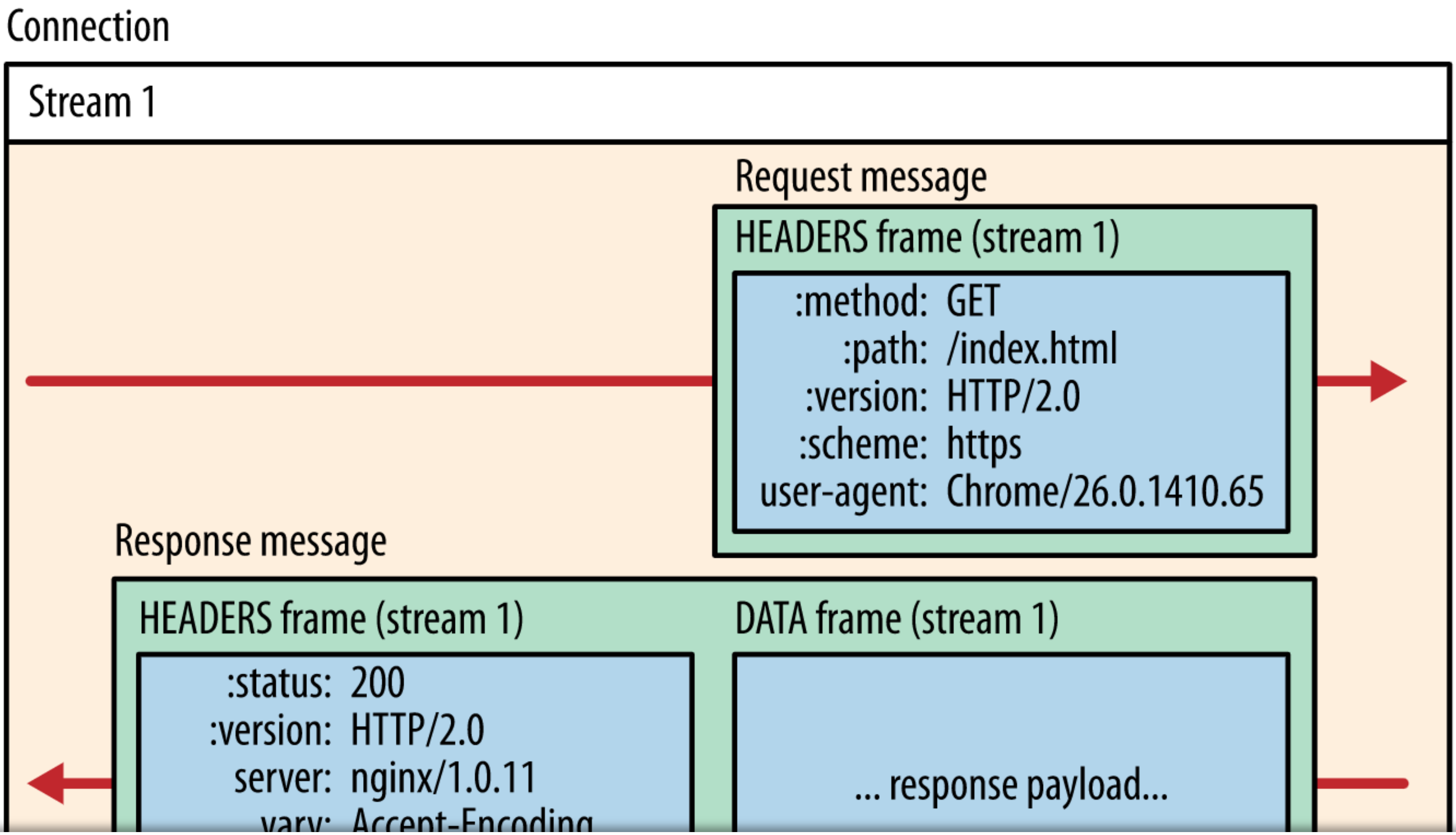
- Head-of-line blocking
  - Small xfers can "block" behind large xfer
- No reordering
  - HTTP response does not "identify" which request it's in response to; obvious in simple request/response
- Can behave *worse* than parallel + persistent
  - Can send expensive query 1 on conn 1, while sending many cheap queries on conn 2

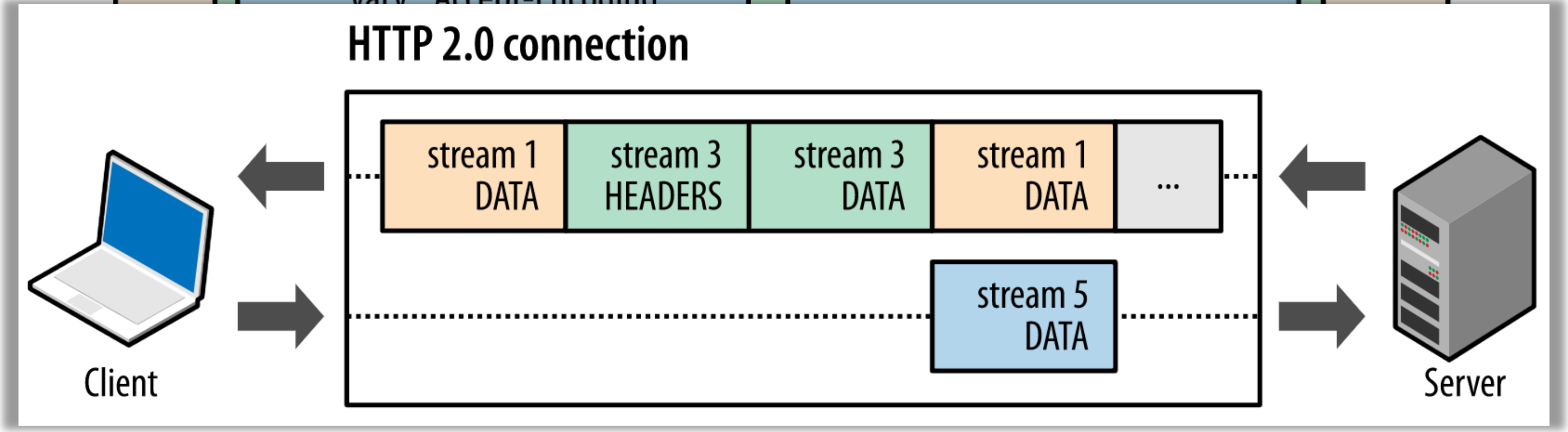
Google's SPDY → HTTP/2 Standard

- Server "push" for content
  - One client request, multiple responses
  - After all, server knows that after parsing HTML, client will immediately request embedded URLs

- Better pipelining and xfer
  - Multiplexing multiple xfers w/o HOL blocking
  - Request prioritization
  - Header compression


Connection
Stream 1
Request message
HEADERS frame (stream 1)
:method: GET
:path: /index.html
:version: HTTP/2.0
:scheme: https
user-agent: Chrome/26.0.1410.65
Response message
HEADERS frame (stream 1)
:status: 200
:version: HTTP/2.0
server: nginx/1.0.11
vary: Accept-Encoding
...
DATA frame (stream 1)
... response payload...
Stream N


Connection
Stream 1
Request message
HEADERS frame (stream 1)
:method: GET
:path: /index.html
:version: HTTP/2.0
:scheme: https
user-agent: Chrome/26.0.1410.65
Response message
HEADERS frame (stream 1)
:status: 200
:version: HTTP/2.0
server: nginx/1.0.11
vary: Accept-Encoding
DATA frame (stream 1)
... response payload...


HTTP 2.0 connection
stream 1 DATA
stream 3 HEADERS
stream 3 DATA
stream 1 DATA
...
stream 5 DATA
Client
Server

Summary

- HTTP: dominant application layer protocol for the web

- HTTP caching had a limited impact (CDNs next)

- Recent optimization and evolution of HTTP for performance and efficiency