

# Web Server Design

## Lecture 10 – HTTPS, HTTP/2, HTTP/3

Old Dominion University

Department of Computer Science

CS 431/531 Fall 2019

**Sawood Alam** <salam@cs.odu.edu>

2019-10-31

Original slides by Michael L. Nelson

HTTPS Is All the Rage!

# ISPs Inject Ad In HTTP HTML Pages

The image shows the homepage of naukri.com, a popular Indian job portal. The header includes the naukri.com logo and navigation links for Jobs, Recruiters, Companies, Services, More, and Login. Below the header, there are tabs for All Jobs, Premium Jobs, Govt. Jobs, and International Jobs. A search bar is prominently displayed with fields for Skills, Designations, Companies, Location, Experience, and Salary, along with a Search button and an Advanced Search link. Below the search bar, there are links to Browse Jobs, All Jobs, Jobs by Company, Jobs by Category, Jobs by Location, Jobs by Designation, and Jobs by Skill. At the bottom of the page, there are sections for Top Employers (IBM, Tech Mahindra) and Best Places to Work (All Sectors, Information Technology, Manufacturing, Services). An advertisement for MTNL's Broadband Festive Bonanza is overlaid on the right side of the page. The ad features a yellow background with red and green text, announcing a save of ₹800 on a new broadband connection. It also mentions a ₹300/- waived off on registration & testing charges and a ₹500/- save on initial charges for Wi-Fi Modem. The ad includes a 'Hurry!' badge and a deadline for the offer.

**naukri.com**  
India's No.1 Job Site

Jobs Recruiters Companies Services More Login

Employer Zone  
Buy Online

All Jobs Premium Jobs Govt. Jobs International Jobs  
Govt. Jobs

Skills, Designations, Companies Location Experience Salary Search  
Advanced Search

Browse Jobs  
All Jobs Jobs by Company Jobs by Category Jobs by Location Jobs by Designation Jobs by Skill

JOIN naukri.com YODLEE Register Free

Top Employers  
IBM Tech Mahindra

Best Places to Work  
All Sectors Information Technology Manufacturing Services

Information Technology  
Novell Nucleus Software ChioDsa/Semicond.

\* Jindal Steel & Power  
KSB Pumps

MTNL  
Broadband Festive Bonanza

**MTNL's BROADBAND FESTIVE BONANZA**

Save **₹800** on a New Broadband Connection

₹300/- Waived Off on Registration & Testing charges

Save ₹500/- on initial charges for Wi-Fi Modem from ₹800/- to ₹300/-

**Hurry!** For booking type "mtnl" on your mobile and send it to 9868552121. Offer valid for Limited Period

This Offer is applicable on all MTNL Broadband Plans

For further details :-  
• Call 1566 / 22221590 • Email at 1566@btl.net.in • Contact your nearest Sanchal Head

For booking type "mtnl" on your mobile and send it to 9868552121. Offer valid for Limited Period

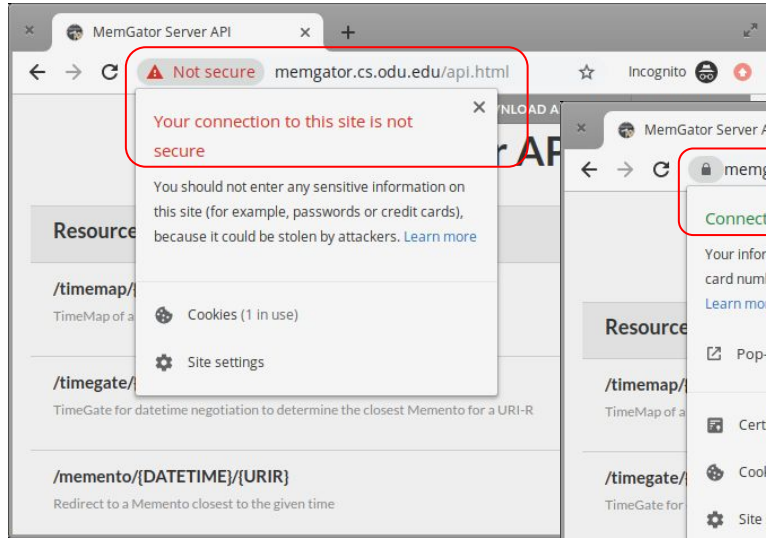
From: <https://www.medianama.com/2015/06/223-mtnl-isp-advertising-airtel/>

# Percentage of Web Pages Loaded by Firefox Using HTTPS

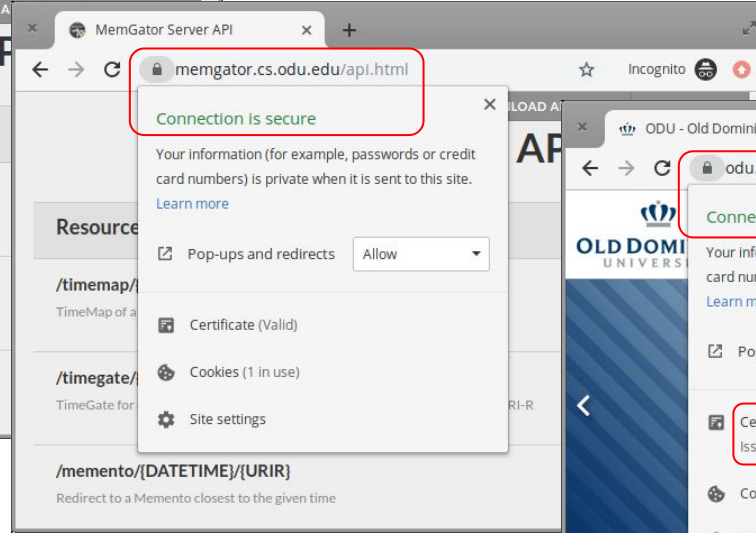


From: <https://letsencrypt.org/stats/>

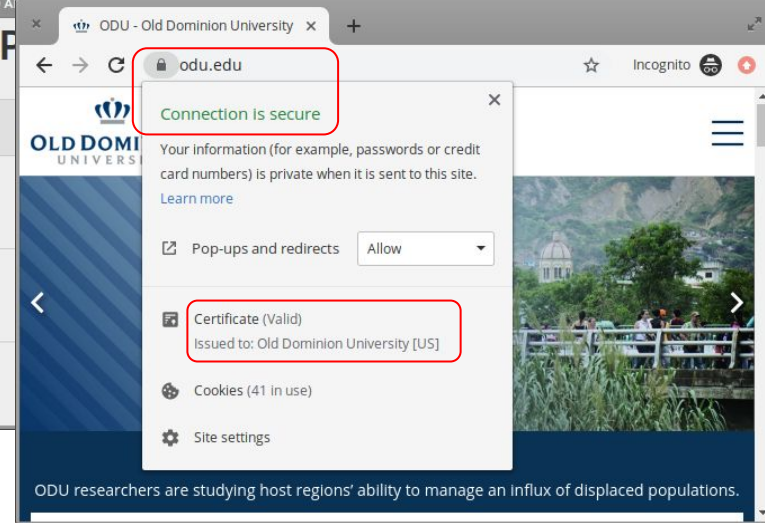
# HTTP vs. HTTPS



No Certificate



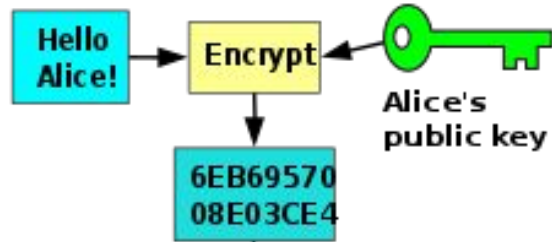
Domain Validation



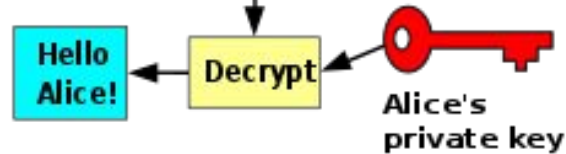
Organization/Extended Validation

# Public-key Cryptography

Bob

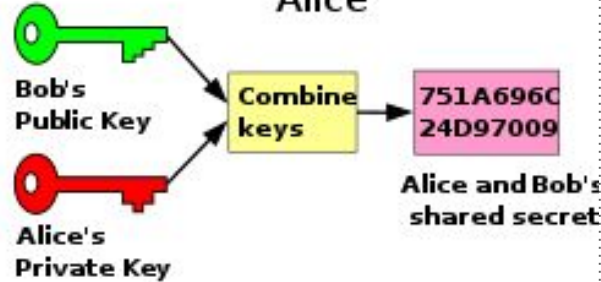


Alice

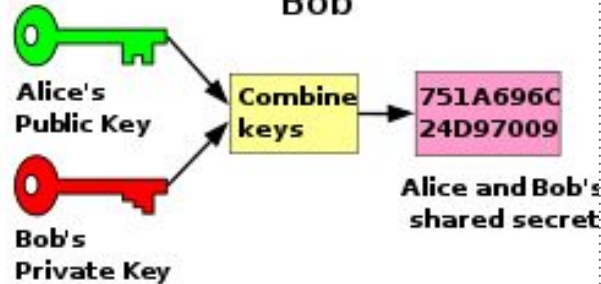


Encryption

Alice

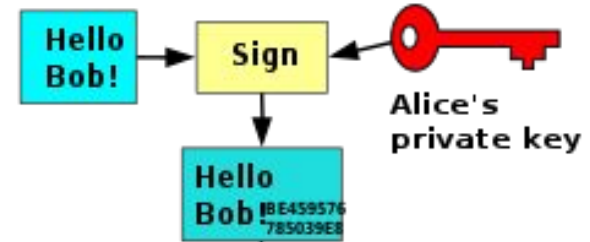


Bob

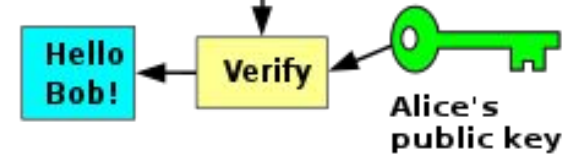


Diffie-Hellman key exchange

Alice

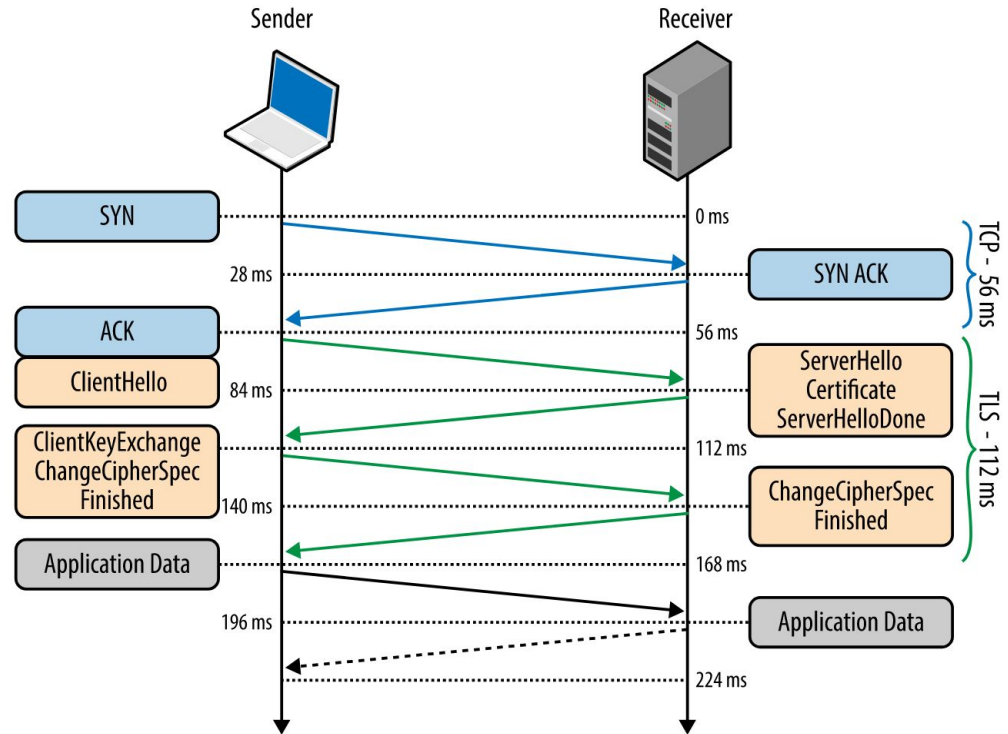
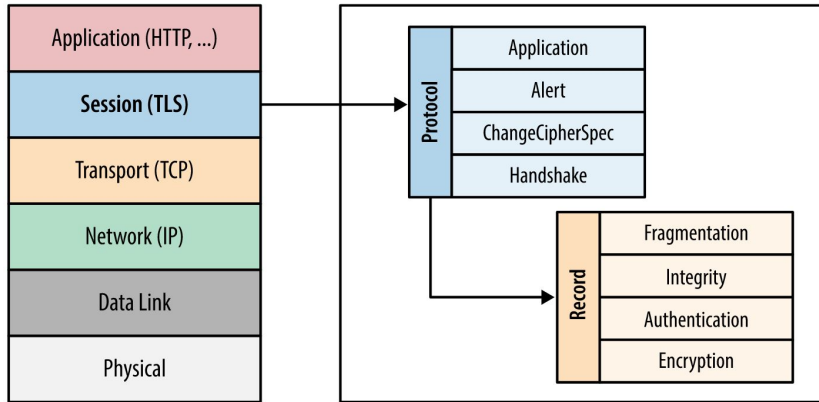


Bob



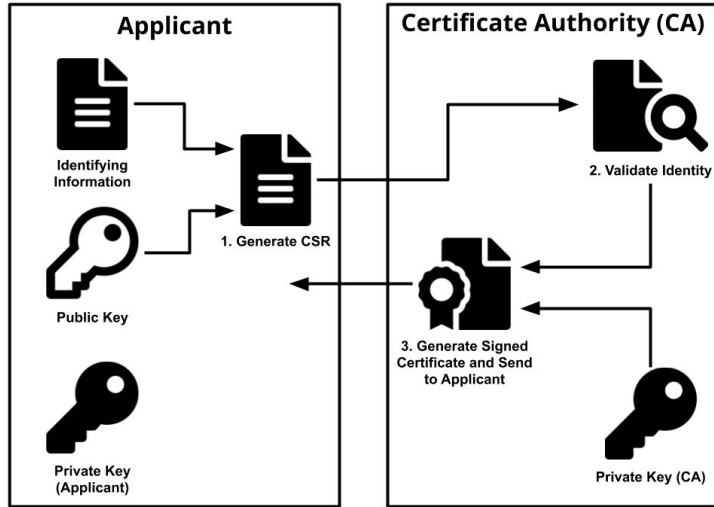
Signing

# Transport Layer Security (TLS)

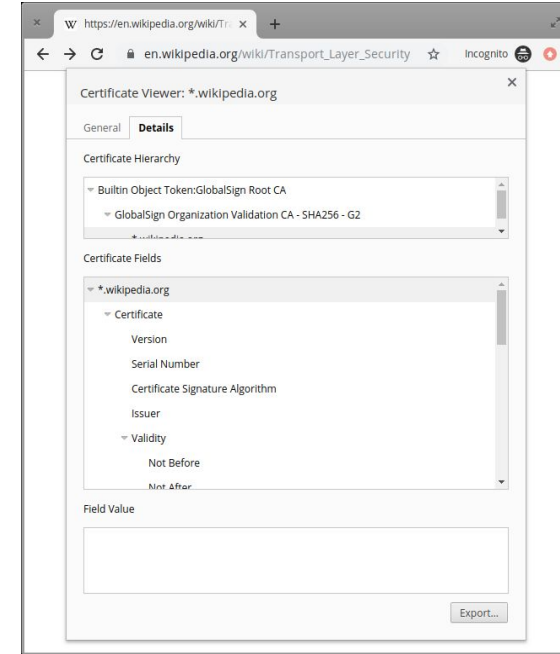
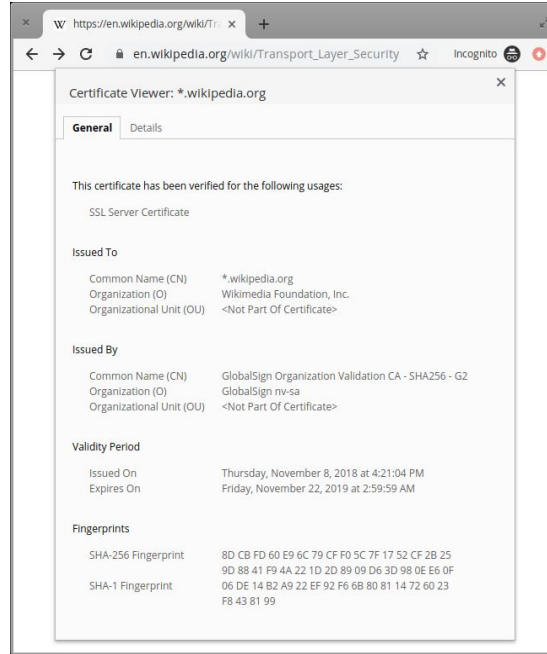


From: <https://hpbn.co/transport-layer-security-tls/>

# Anatomy of TLS Certificate



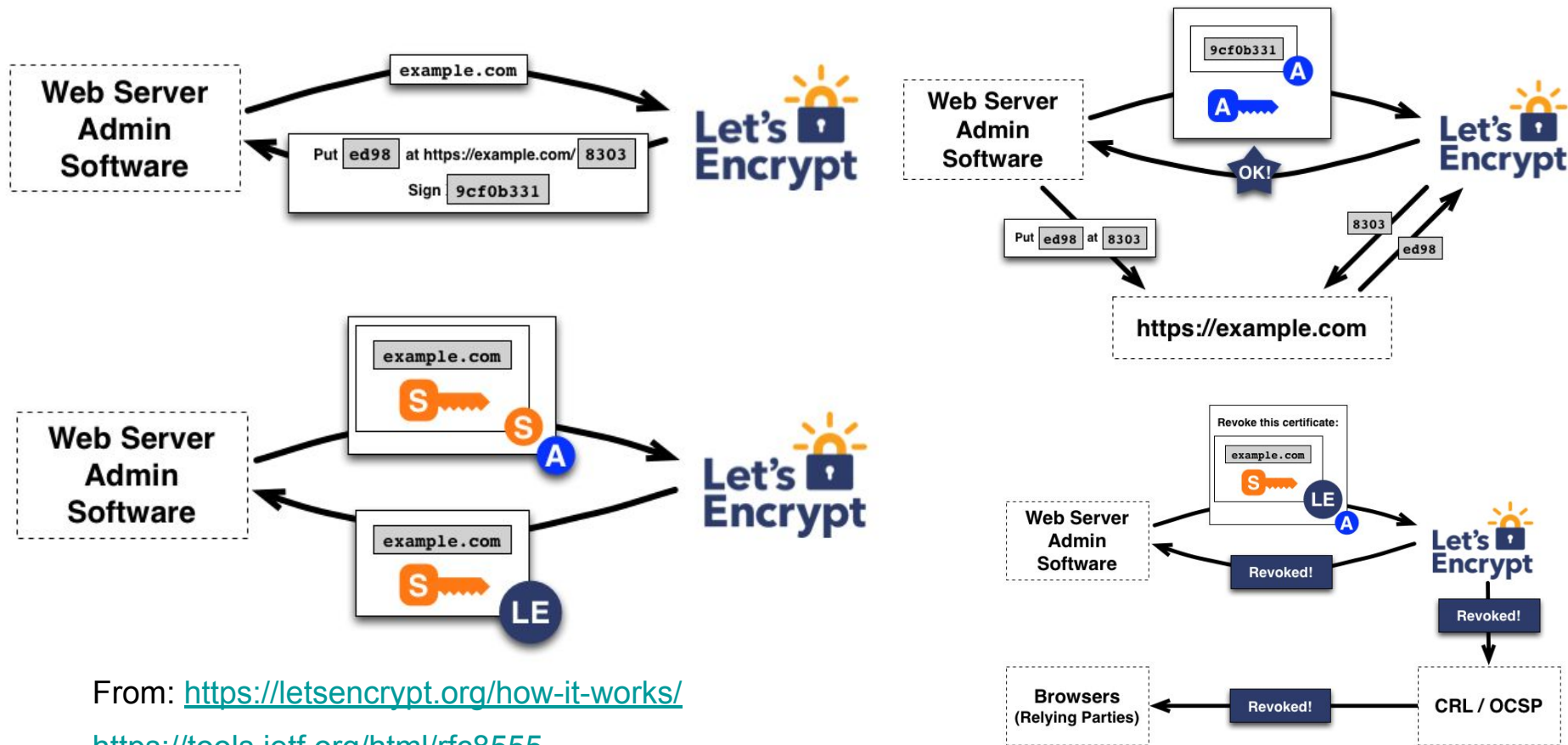
Certificate Issuance from a Certificate Authority (CA)



Certificate Viewer



# Automatic Certificate Management Environment (ACME)



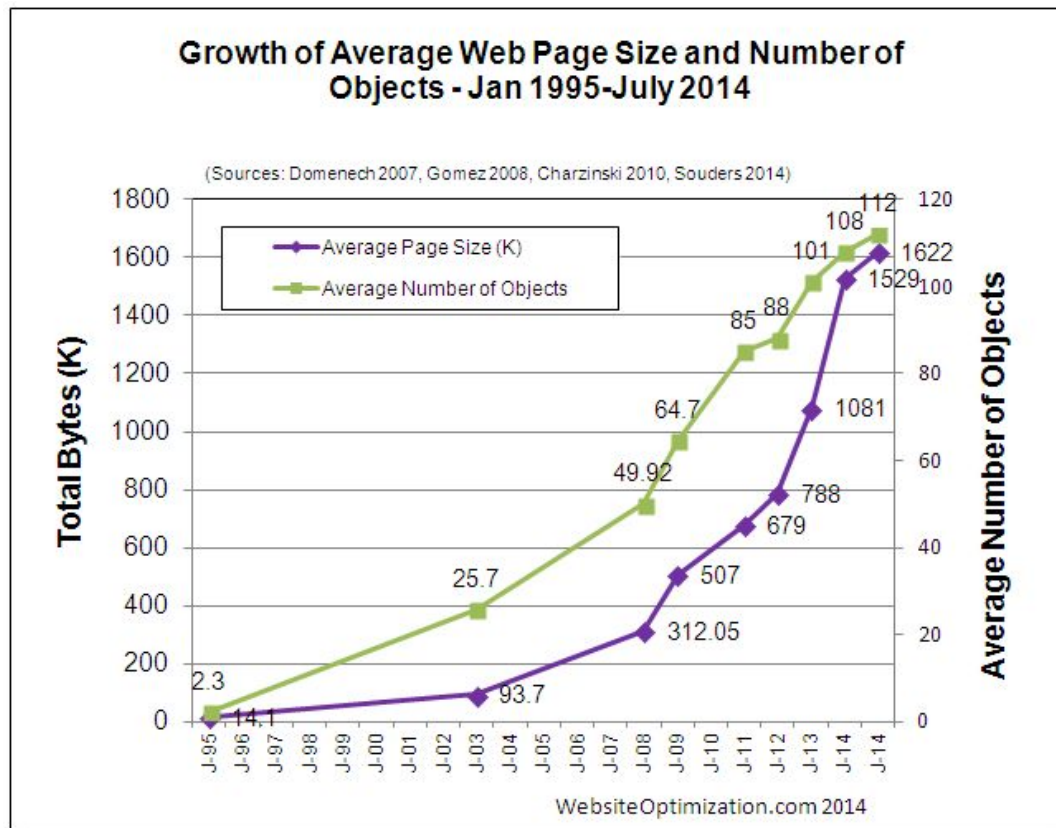
HTTP/1.1 is awesome –  
you can't argue with its deployed footprint.

But there are well-known performance  
limitations.

# HTTP is not a good fit for TCP

- TCP is designed for long-lived, bulk transfers
  - High-handshake costs, TLS adds even more to startup costs
  - HTTP requests are short and bursty
- Parallelism needed, but:
  - Pipelining has problems with head-of-line-blocking, recovering from failures
  - More TCP connections, more client+server resources to manage the sockets, bandwidth consumed by TCP overhead
  - In practice, browsers limit to six concurrent connections

# Parallelism Is Needed Because of Page Bloat



From: <https://www.webbloatscore.com/>

See also: <https://httparchive.org/reports/state-of-the-web>

# Parallelism Limits In Practice

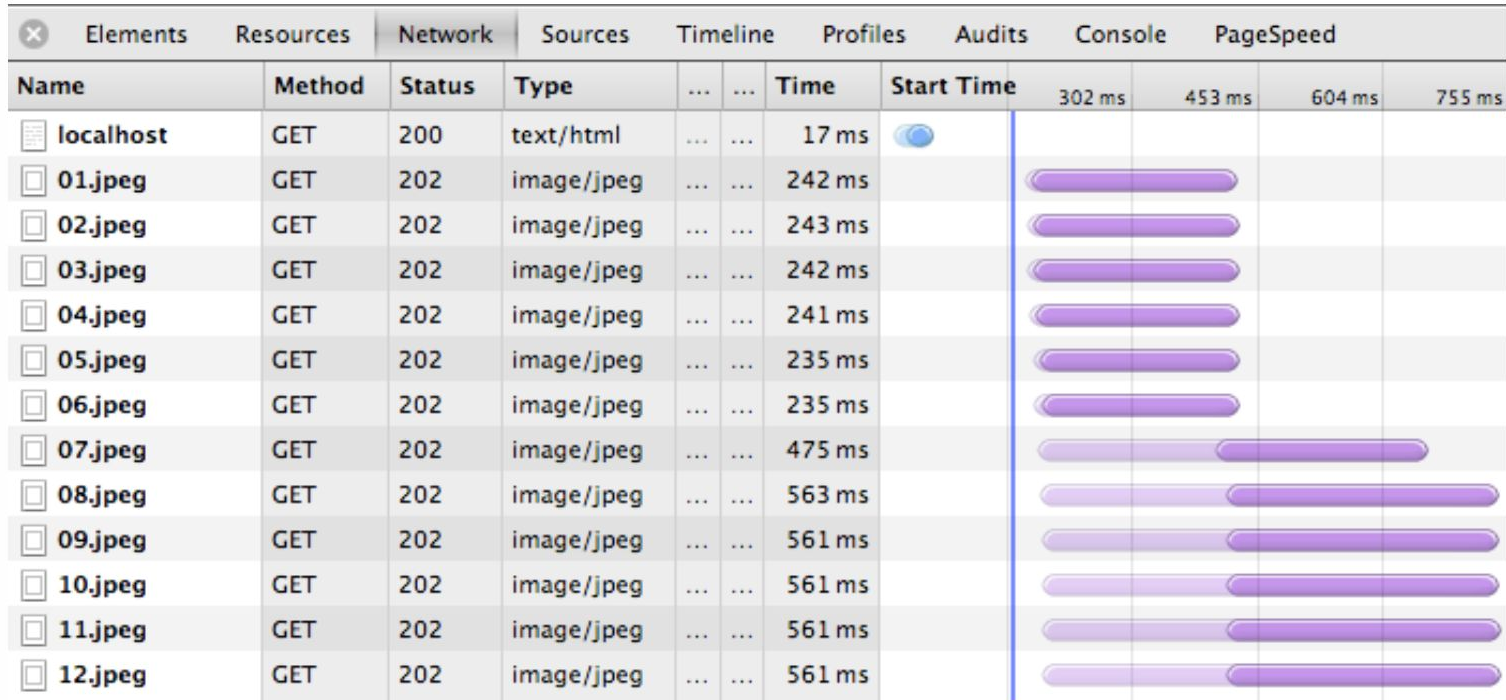


Figure 11-5. Staggered resource downloads due to six-connection limit per origin

From: <https://hpbn.co/http1x/>

# HTTP Headers: Metadata >> Data

```
$> curl --trace-ascii - -d '{"msg":"hello"}' http://www.igvita.com/api
```

```
== Info: Connected to www.igvita.com
```

```
=> Send header, 218 bytes ❶
```

```
POST /api HTTP/1.1
```

```
User-Agent: curl/7.24.0 (x86_64-apple-darwin12.0) libcurl/7.24.0 ...
```

```
Host: www.igvita.com
```

```
Accept: */*
```

```
Content-Length: 15 ❷
```

```
Content-Type: application/x-www-form-urlencoded
```

```
=> Send data, 15 bytes (0xf)
```

```
{"msg":"hello"}
```

```
<= Recv header, 134 bytes ❸
```

```
HTTP/1.1 204 No Content
```

```
Server: nginx/1.0.11
```

```
Via: HTTP/1.1 GWA
```

```
Date: Thu, 20 Sep 2012 05:41:30 GMT
```

```
Cache-Control: max-age=0, no-cache
```

Here, 15 bytes of json + 352 bytes of request and response headers

- ❶ HTTP request headers: 218 bytes
- ❷ 15-byte application payload ({ "msg": "hello" })
- ❸ 204 response from the server: 134 bytes

From: <https://hpbn.co/http2/>

# HTTP/1.1 Optimizations

# Image Sprites



Send one large image of all flags, use CSS to “cut out” the flag you need

From: <https://daniel.haxx.se/http2/>



# Inlining & Concatenation

- Inlining: send small images as base64

```

```

[https://en.wikipedia.org/wiki/Data\\_URI\\_scheme](https://en.wikipedia.org/wiki/Data_URI_scheme)

- Concatenation: put all of your .js/.css files into a single, large .js/.css file
  - Probably sends more than you need
  - Small changes in one file means changes in the entire file

# Domain Sharding

Six connections per domain,  
But with the overhead of additional  
DNS lookups.

From: <https://daniel.haxx.se/http2/>

200	GET	174.jpg	w.cdn-expressen.se	jpeg	6.14 KB	→ 105 ms
200	GET	174.jpg	y.cdn-expressen.se	jpeg	4.19 KB	→ 172 ms
200			dn-expressen.se	jpeg	4.48 KB	→ 223 ms
200			dn-expressen.se	jpeg	4.58 KB	→ 173 ms
200			dn-expressen.se	jpeg	35.18 KB	→ 56 ms
200			dn-expressen.se	jpeg	12.97 KB	→ 165 ms
200			dn-expressen.se	jpeg	4.83 KB	→ 56 ms
200			dn-expressen.se	jpeg	9.54 KB	→ 228 ms
200			dn-expressen.se	jpeg	182.50 KB	→ 285 ms
200			dn-expressen.se	jpeg	5.66 KB	→ 104 ms
200			dn-expressen.se	jpeg	12.24 KB	→ 287 ms
200			dn-expressen.se	jpeg	6.85 KB	→ 225 ms
200			dn-expressen.se	jpeg	7.50 KB	→ 173 ms
200			dn-expressen.se	gif	2.85 KB	→ 227 ms
200			dn-expressen.se	jpeg	50.87 KB	→ 188 ms
200			dn-expressen.se	jpeg	6.65 KB	→ 55 ms
200	GET	265.jpg	y.cdn-expressen.se	jpeg	6.09 KB	→ 196 ms
200	GET	540.jpg	z.cdn-expressen.se	jpeg	16.14 KB	→ 67 ms
200	GET	540.jpg	w.cdn-expressen.se	jpeg	19.89 KB	→ 112 ms
200	GET	174.jpg	z.cdn-expressen.se	jpeg	5.03 KB	→ 55 ms
200	GET	540.jpg	w.cdn-expressen.se	jpeg	21.27 KB	→ 108 ms
200	GET	540.jpg	x.cdn-expressen.se	jpeg	5.43 KB	→ 237 ms
200	GET	174.jpg	y.cdn-expressen.se	jpeg	6.08 KB	→ 169 ms
200	GET	174.jpg	w.cdn-expressen.se	jpeg	5.62 KB	→ 105 ms
200	GET	540.jpg	x.cdn-expressen.se	jpeg	20.32 KB	→ 241 ms
200	GET	174.jpg	z.cdn-expressen.se	jpeg	6.66 KB	→ 55 ms
200	GET	540.jpg	x.cdn-expressen.se	jpeg	11.13 KB	→ 237 ms
200	GET	265.jpg	w.cdn-expressen.se	jpeg	5.20 KB	→ 111 ms
200	GET	265.jpg	x.cdn-expressen.se	jpeg	6.93 KB	→ 288 ms
200	GET	265.jpg	x.cdn-expressen.se	jpeg	12.09 KB	→ 249 ms
200	GET	265.jpg	z.cdn-expressen.se	jpeg	5.92 KB	→ 167 ms
200	GET	original.jpg	y.cdn-expressen.se	jpeg	64.28 KB	→ 192 ms
200	GET	original.jpg	w.cdn-expressen.se	jpeg	21.88 KB	→ 106 ms
200	GET	540.jpg	w.cdn-expressen.se	jpeg	18.77 KB	→ 112 ms
200	GET	128.jpg	z.cdn-expressen.se	jpeg	3.34 KB	→ 55 ms
200	GET	265.jpg	x.cdn-expressen.se	jpeg	13.00 KB	→ 245 ms
200	GET	265.jpg	y.cdn-expressen.se	jpeg	9.19 KB	→ 194 ms
200	GET	540.jpg	w.cdn-expressen.se	jpeg	13.13 KB	→ 108 ms
200	GET	174.jpg	y.cdn-expressen.se	jpeg	5.66 KB	→ 197 ms
200	GET	174.jpg	z.cdn-expressen.se	jpeg	5.56 KB	→ 55 ms
200	GET	174.jpg	w.cdn-expressen.se	jpeg	5.07 KB	→ 111 ms
200	GET	174.jpg	z.cdn-expressen.se	jpeg	6.16 KB	→ 59 ms
200	GET	174.jpg	y.cdn-expressen.se	jpeg	6.57 KB	→ 210 ms
200	GET	174.jpg	y.cdn-expressen.se	jpeg	4.58 KB	→ 12 ms
200	GET	265.jpg	y.cdn-expressen.se	jpeg	11.49 KB	→ 173 ms

# Evolution from SPDY to HTTP/2

- November 2009: Google begins work on SPDY to address performance limitations of HTTP/1.1
- September 2010: SPDY supported in Chrome
- January 2011: SPDY deployed for all Google services
- March 2012: Twitter supports SPDY
- March 2012: Call for proposals for HTTP/2
- June 2012: NGINX supports SPDY
- July 2012: Facebook announces planned support for SPDY
- November 2012: First draft of HTTP/2 (based on SPDY)
- August 2014: HTTP/2 draft-17 and HPACK draft-12 are published
- August 2014: Working Group last call for HTTP/2
- February 2015: IESG approved HTTP/2 and HPACK drafts
- May 2015: RFC 7540 (HTTP/2) and RFC 7541 (HPACK) are published

Collected from: <https://en.wikipedia.org/wiki/SPDY>, <https://hpbn.co/http2/>

# Google Deprecates SPDY

“HTTP/2's primary changes from HTTP/1.1 focus on improved performance. Some key features such as multiplexing, header compression, prioritization and protocol negotiation evolved from work done in an earlier open, but non-standard protocol named SPDY. **Chrome has supported SPDY since Chrome 6, but since most of the benefits are present in HTTP/2, it's time to say goodbye. We plan to remove support for SPDY in early 2016,** and to also remove support for the TLS extension named NPN in favor of ALPN in Chrome at the same time. Server developers are strongly encouraged to move to HTTP/2 and ALPN.

We're happy to have contributed to the open standards process that led to HTTP/2, and hope to see wide adoption given the broad industry engagement on standardization and implementation.”

Quoted in: <https://hpbn.co/http2/> Original: <https://blog.chromium.org/2015/02/hello-http2-goodbye-spdy.html>

High-level semantics of HTTP  
don't change in HTTP/2,  
but the method of packaging and transport do.

# Binary Framing Layer

No more hand-crafted telnet sessions – boo!!!!

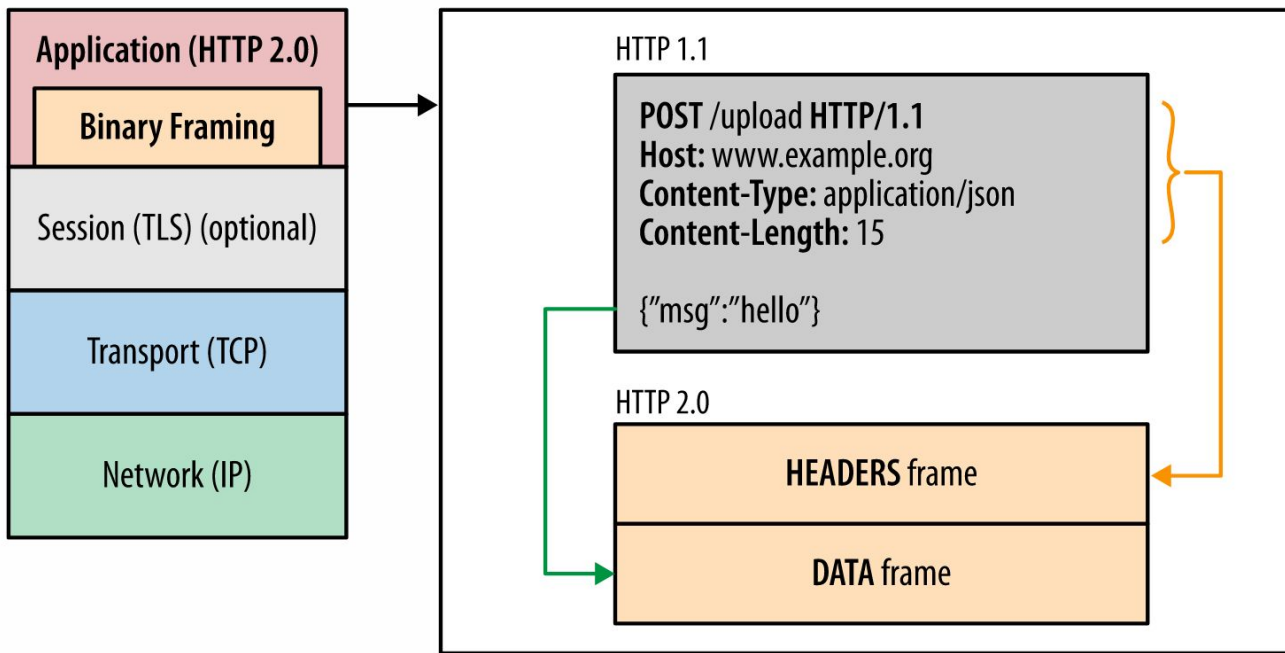
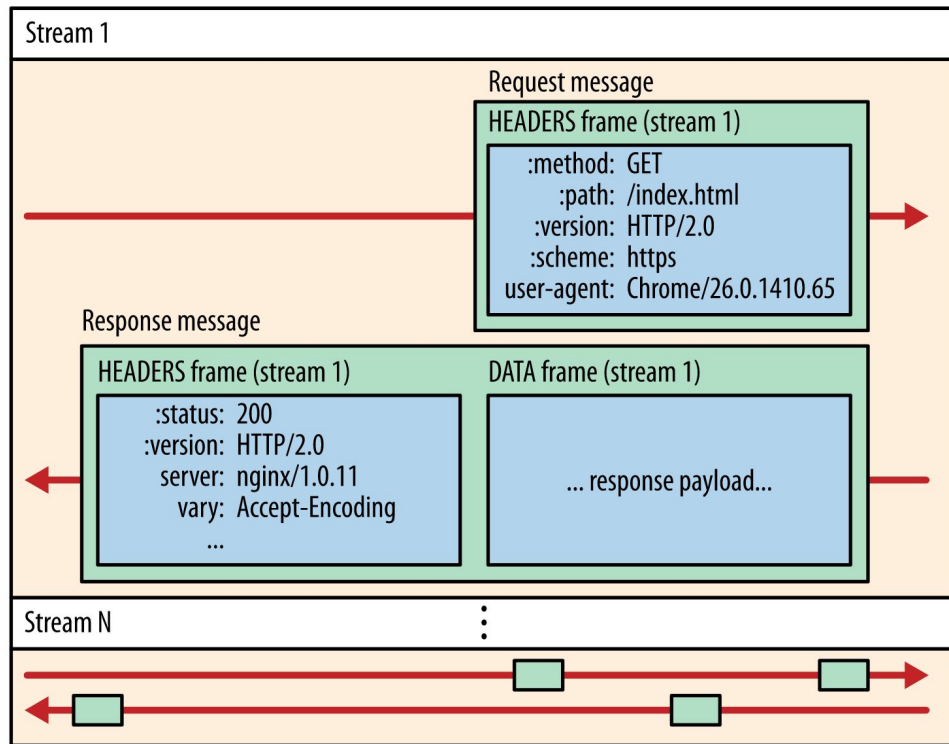


Figure 12-1. HTTP/2 binary framing layer

# Streams, Messages, Frames

Connection



**Stream:** bi-directional connection, with 1 or more messages

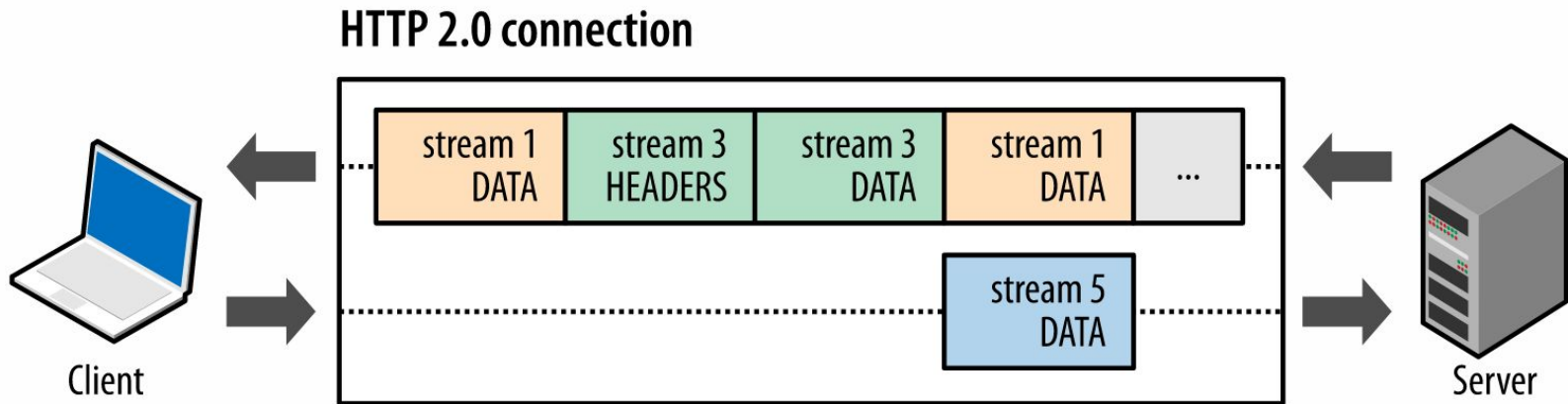
**Message:** logically complete request or response

**Frame:** typed, atomic unit of communication

Figure 12-2. HTTP/2 streams, messages, and frames

From: <https://hpbnc.co/http2/>

# Request & Response Multiplexing



*Figure 12-3. HTTP/2 request and response multiplexing within a shared connection*

- Interleave multiple requests in parallel without blocking on any one
- Interleave multiple responses in parallel without blocking on any one
- Use a single connection to deliver multiple requests and responses in parallel
- Remove unnecessary HTTP/1.x workarounds (such as concatenated files, image sprites, and domain sharding)
- Deliver lower page load times by eliminating unnecessary latency and improving utilization of available network capacity

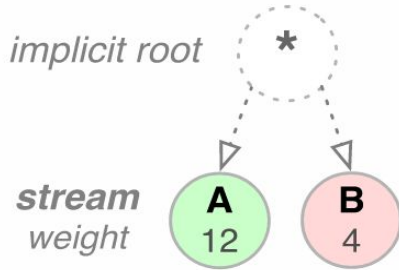
Note: frames cannot be received out of order!

From: <https://hpbn.co/http2/>

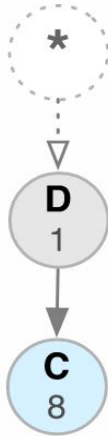


# Stream Dependencies & Weights

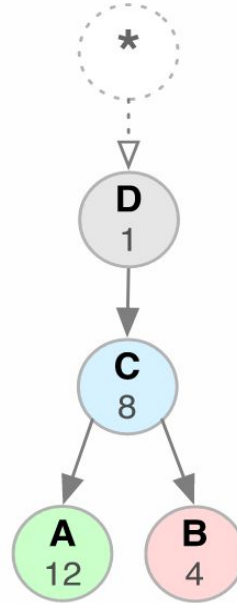
A gets  $\frac{3}{4}$  of bandwidth, B gets  $\frac{1}{4}$   
A & B are dependent on the “root”  
stream (i.e., no dependencies)



C depends on D, service D  
first (weights trumped by  
dependency)



D before C, C before  
A & B, weight A & B  
as before



D before C, C & E  
equally Before A & B,  
weight A & B as before

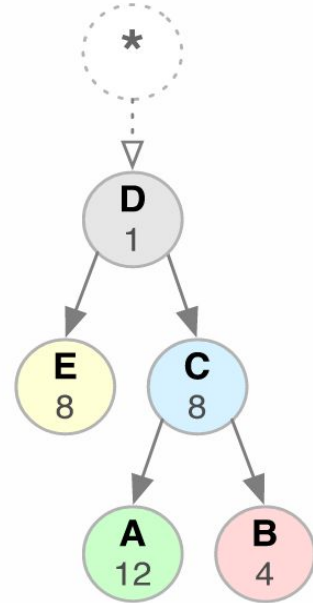


Figure 12-4. HTTP/2 stream dependencies and weights

# Server Push: 1 Request, N Responses

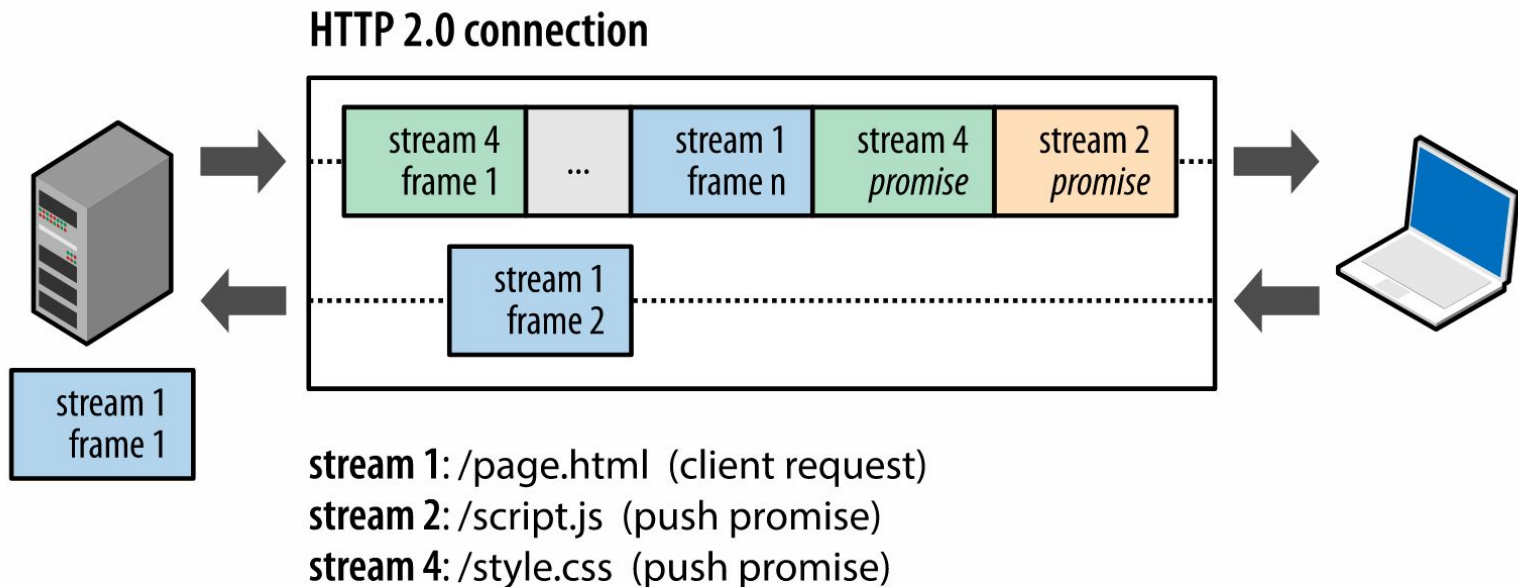


Figure 12-5. Server initiates new streams (promises) for push resources

See discussion of HTTP/2 push in:  
<https://daniel.haxx.se/blog/2018/11/11/http-3/>

Conceptually similar to inlining, rel="preload", rel="prefetch", etc.  
Can only push with same-origin policy.

From: <https://hpbn.co/http2/>

# Header Repetitiveness Allows Compression

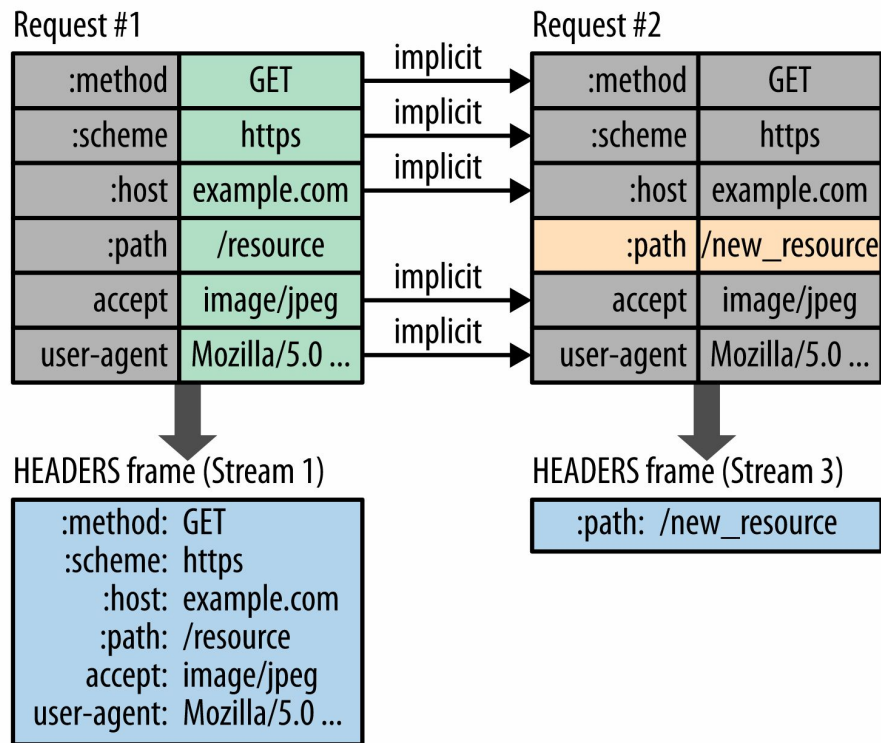


Figure 12-6. HPACK: Header Compression for HTTP/2

Note: headers beginning with “:” are “pseudo-headers” (RFC 7540, 8.1.2.1); or “things-that-should-have-been-headers-in-HTTP/1.1”  
Pseudo-headers have to be listed before real headers.

From: <https://hpbnc.co/http2/>

# HTTP/1.1 HTTP/2 Upgrade

```
GET /page HTTP/1.1
Host: server.example.com
Connection: Upgrade, HTTP2-Settings
Upgrade: h2c ❶
HTTP2-Settings: (SETTINGS payload) ❷
```

```
HTTP/1.1 200 OK ❸
Content-length: 243
Content-type: text/html
```

(... HTTP/1.1 response ...)

(or)

```
HTTP/1.1 101 Switching Protocols ❹
Connection: Upgrade
Upgrade: h2c
```

(... HTTP/2 response ...)

Note:

“h2” = HTTP/2 over TLS

“h2c” = HTTP/2 over clear text TCP

- ❶ Initial HTTP/1.1 request with HTTP/2 upgrade header
- ❷ Base64 URL encoding of HTTP/2 SETTINGS payload
- ❸ Server declines upgrade, returns response via HTTP/1.1
- ❹ Server accepts HTTP/2 upgrade, switches to new framing

From: <https://hpbnc.co/http2/>

# 9 Byte Frame Header

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

Figure 12-7. Common 9-byte frame header

Note: frames cannot be received out of order! Stream id, but not frame id.

## Note

Technically, the Length field allows payloads of up to  $2^{24}$  bytes (~16MB) per frame. However, the HTTP/2 standard sets the default maximum payload size of DATA frames to  $2^{14}$  bytes (~16KB) per frame and allows the client and server to negotiate the higher value. Bigger is not always better: smaller frame size enables efficient multiplexing and minimizes head-of-line blocking.

## Header Types:

- DATA - Used to transport HTTP message bodies
- HEADERS - Used to communicate header fields for a stream
- PRIORITY - Used to communicate sender-advised priority of a stream
- RST\_STREAM - Used to signal termination of a stream
- SETTINGS - Used to communicate configuration parameters for the connection
- PUSH\_PROMISE - Used to signal a promise to serve the referenced resource
- PING - Used to measure the roundtrip time and perform "liveness" checks
- GOAWAY - Used to inform the peer to stop creating streams for current connection
- WINDOW\_UPDATE - Used to implement flow stream and connection flow control
- CONTINUATION - Used to continue a sequence of header block fragments

From: <https://hpbnc.co/http2/>

# Example Binary HTTP/2 Request

▼ HyperText Transfer Protocol 2

▼ Stream: HEADERS, Stream ID: 1, Length 20

Length: 20  
Type: HEADERS (1)  
▼ Flags: 0x05

- .... ...1 = End Stream: True
- .... .1.. = End Headers: True
- .... 0... = Padded: False
- ..0. .... = Priority: False
- 00.0 ..0. = Unused: 0x00

0... .. = Reserved: 0x00000000  
.000 0000 0000 0000 0000 0000 0000 0001 = Stream Identifier: 1  
[Pad Length: 0]  
Header Block Fragment: 8682418aa0e41d139d09b8f01e078453032a2f2a  
[Header Length: 100]

common frame header

HPACK encoded headers

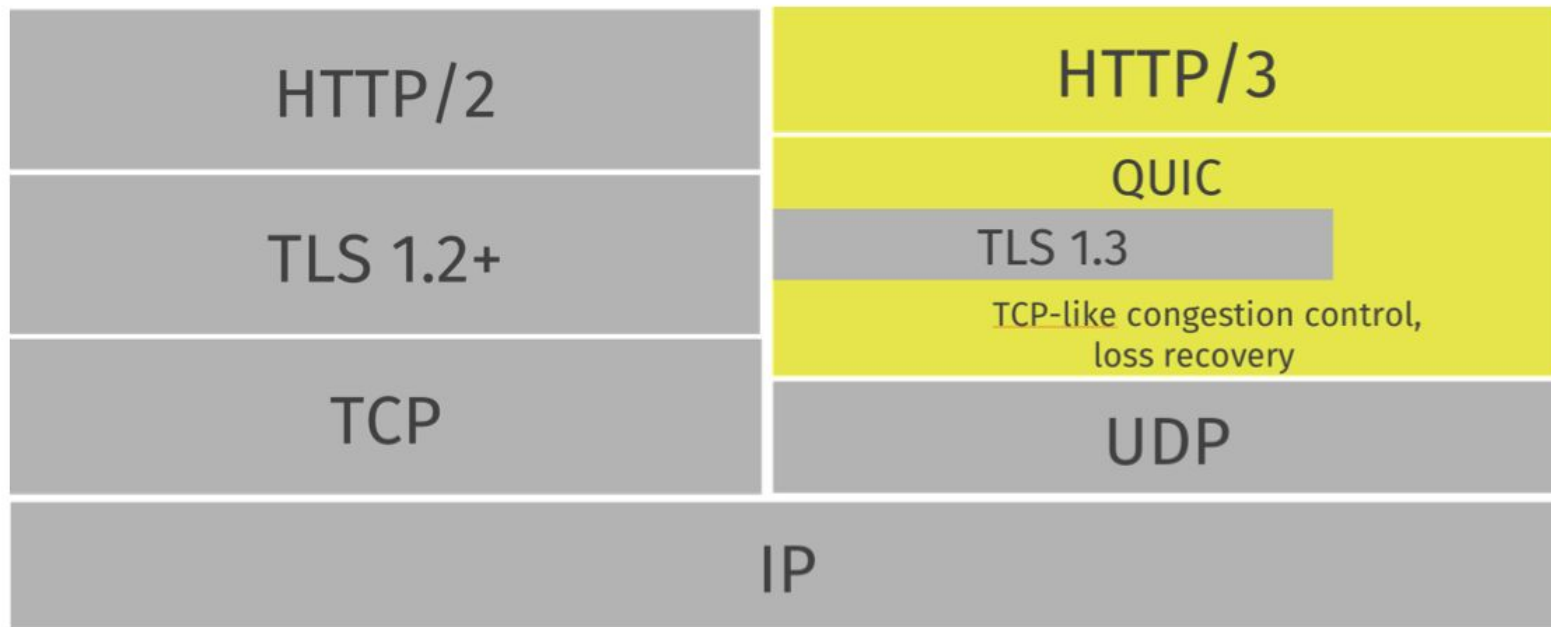
- ▶ Header: :scheme: http
- ▶ Header: :method: GET
- ▶ Header: :authority: localhost:8080
- ▶ Header: :path: /
- ▼ Header: accept: \*/\*
  - Name Length: 6
  - Name: accept
  - Value Length: 3
  - Value: \*/\*
  - Representation: Literal Header Field with Incremental Indexing – Indexed Name
  - Index: 19

Figure 12-8. Decoded HEADERS frame in Wireshark

From: <https://hpbnc.co/http2/>

# HTTP/3 Network Stack

HTTP/2 optimizes within TCP context (e.g., binary, streams & frames),  
HTTP/3 *replaces* TCP



From: <https://daniel.haxx.se/blog/2018/11/26/http3-explained/>

# HTTP/3

- “HTTP-over-QUIC” was just recently renamed “HTTP/3” (Nov 2018)
  - <https://daniel.haxx.se/blog/2018/11/11/http-3/>
  - Not really deployed yet, still in development
- Major changes:
  - Streams are moved from the HTTP layer to the QUIC layer
    - HTTP/2 fixes HTTP head-of-line blocking, but not TCP head-of-line blocking (i.e., streams in TCP can still be held up by dropped TCP packets)
  - Since streams are independent, header compression changes
  - There is no clear-text version of HTTP/3 (integral TLS 1.3)
  - QUIC has faster handshakes than TCP + TLS