
HTTP/0.9 (1991)

HTTP/1.0 (1996)

HTTP/1.1 (1999)

To reduce the load on the server, HTTP/1.1's approach was to limit its TCP connections

"A single-user client should not maintain more than 2 connections with any server or proxy."

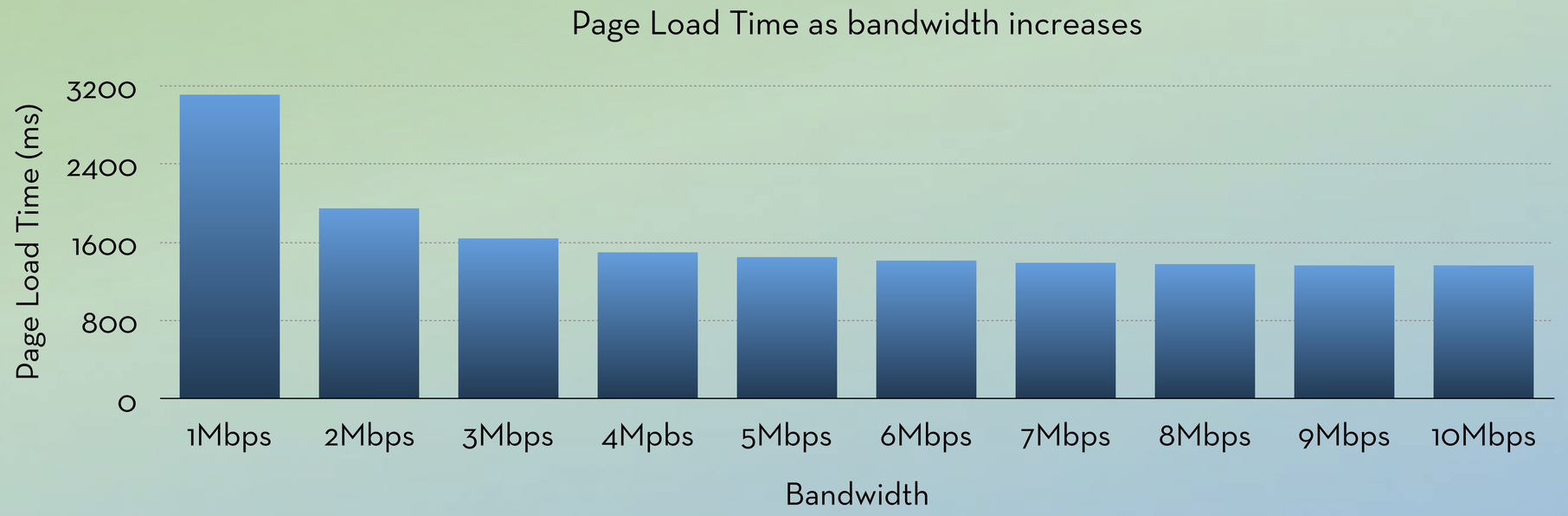
In real life, browsers hold ~6 TCP connections simultaneously per origin.

More Bandwidth Doesn’t Make a Big Difference

An increase from 5Mbps to 10Mbps results in a disappointing 5% improvement in page load times.

Source

Bandwidth & Round-Trip Time


Page Load Time as bandwidth increases
Page Load Time (ms)
3200
2400
1600
800
0
1Mbps
2Mbps
3Mbps
4Mpbs
5Mbps
6Mbps
7Mbps
8Mbps
9Mbps
10Mbps
Bandwidth

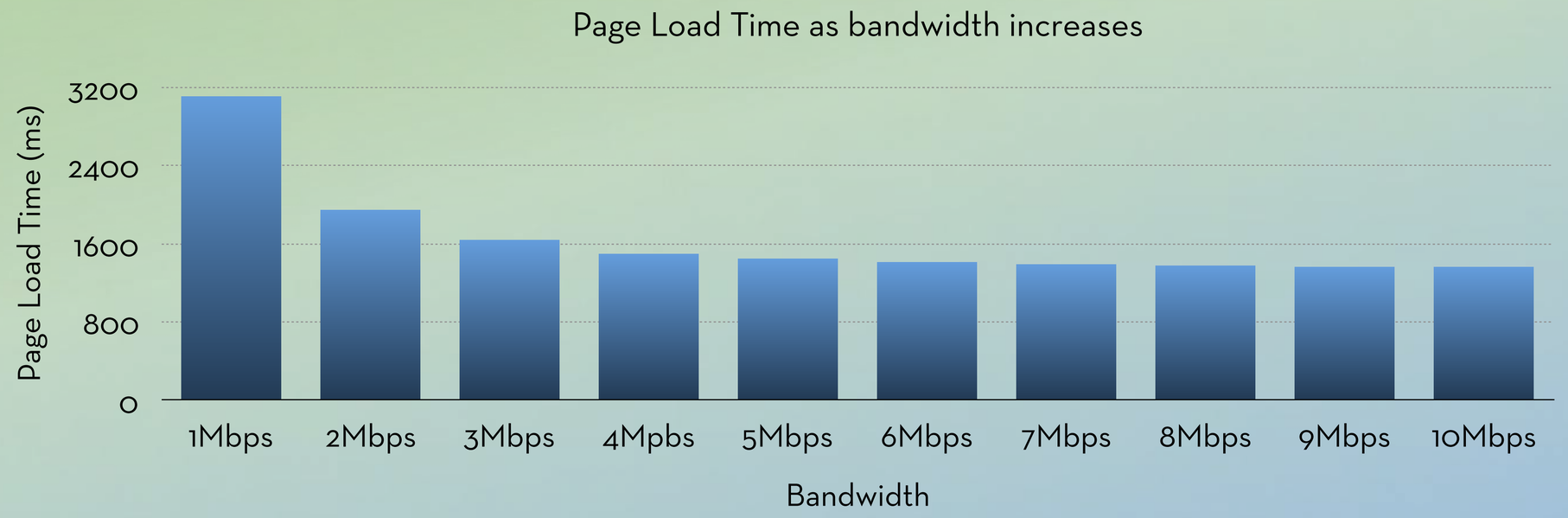
Source

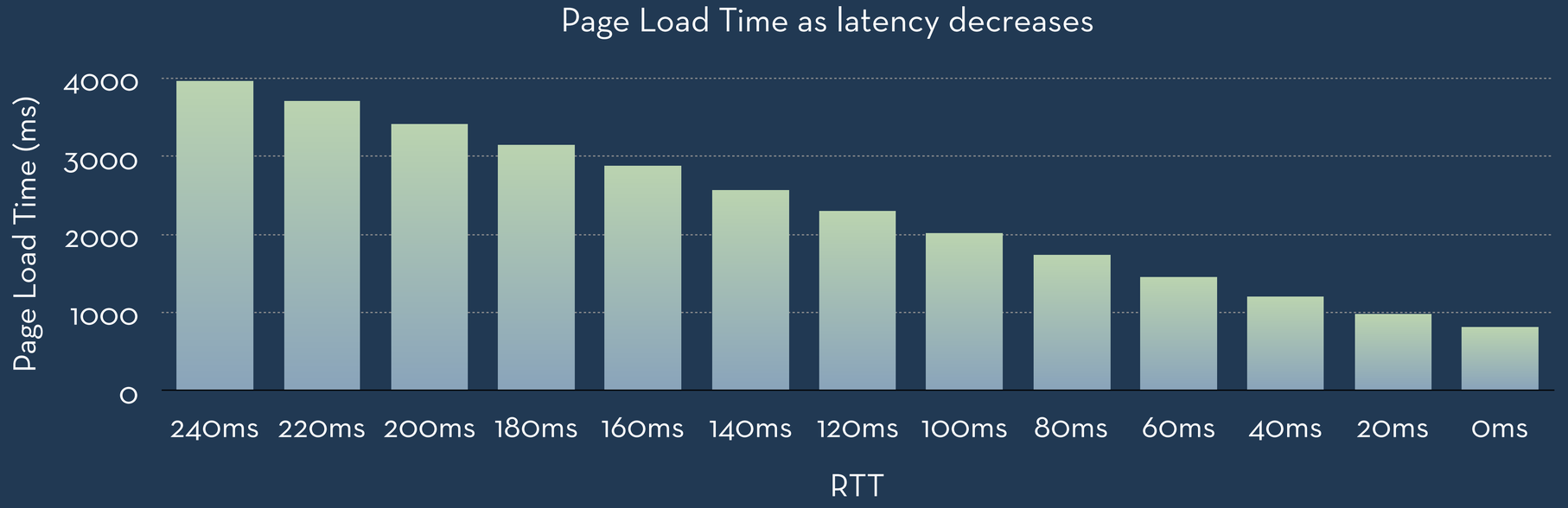
Round-trip-times (RTT) have a bigger impact on performance, more than bandwidth does.

Source

Round-trip-times (RTT) have a bigger impact on performance, more than bandwidth does.

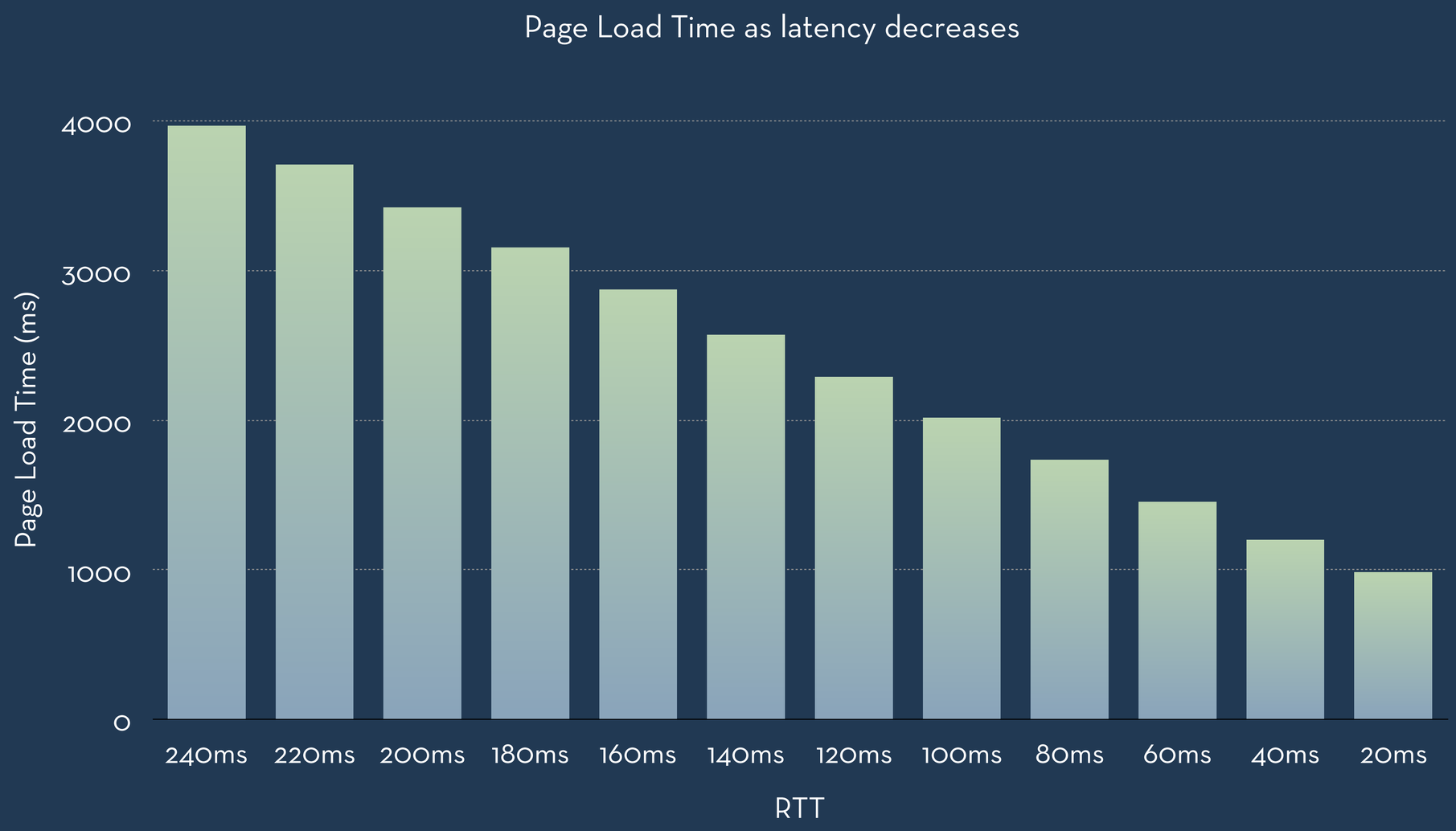
Source


Page Load Time as bandwidth increases
Page Load Time (ms)
3200
2400
1600
800
0
1Mbps
2Mbps
3Mbps
4Mpbs
5Mbps
6Mbps
7Mbps
8Mbps
9Mbps
10Mbps
Bandwidth


Page Load Time as latency decreases
Page Load Time (ms)
4000
3000
2000
1000
0
240ms
220ms
200ms
180ms
160ms
140ms
120ms
100ms
80ms
60ms
40ms
20ms
0ms
RTT

Source

“HTTP2 produces the biggest performance gains on mobile bc it remedies high latency”
— @patrickhamann #smashingconf


Page Load Time as latency decreases
Page Load Time (ms)
4000
3000
2000
1000
0
240ms
220ms
200ms
180ms
160ms
140ms
120ms
100ms
80ms
60ms
40ms
20ms
RTT

Source


SPDY (2010)

Multiplexing: allow concurrent requests across a single TCP connection;

Allow browsers to **prioritise assets** so that vital resources of a page could are sent first;

Compress and reduce HTTP **headers**;

Server push: A server can push important resources to the browser before being asked for them.


HTTP/2 (2015)

Networking protocol for **low-latency transport** of content over the web.

Originally started out from the SPDY protocol, now **standardised as HTTP version 2**.

- Multiplexing
- Compressed headers
- Asset Prioritisation & Dependencies
- Server Push
  (saves the time it takes the client to ask for the resources)

HTTP/2 is backwards-compatible
with HTTP/1.1

Building for Performance with HTTP/2

What do you need to enable HTTP/2?

SSL/TLS required

Leveraging additional benefits of SSL

Google* uses secure connections as a ranking signal, and browsers are starting to flag non-https websites as ‘not secure’.

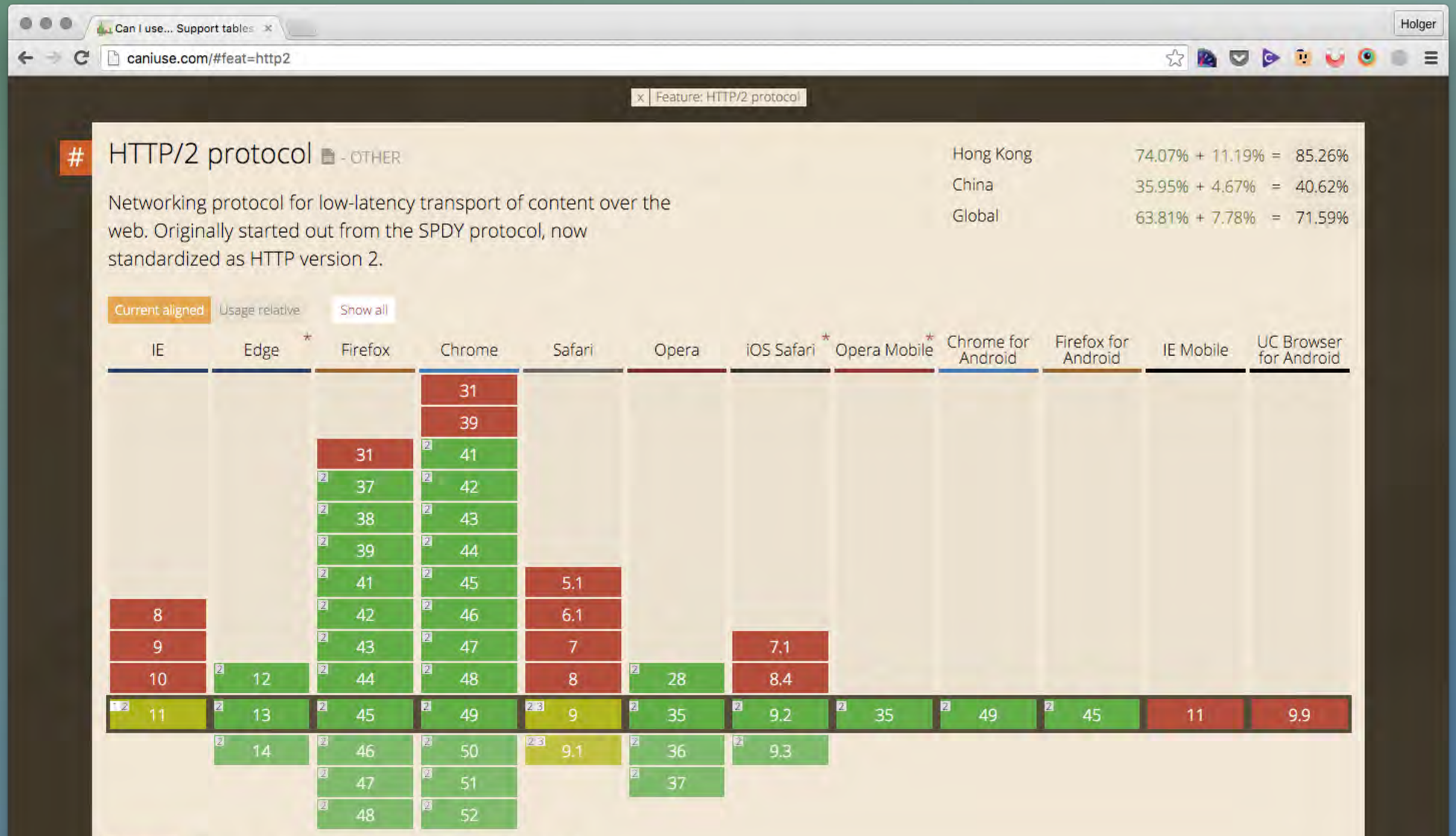
Some HTML5 APIs will also require secure connections in the future (e.g. Geolocation).

** Baidu Analytics includes a site speed section, so they might follow this trend in the future.*

Serving HTTP/2

Apache Module mod_http2

HTTP/2 supporting browsers



Under HTTP/2, some of our current best practices might impact performance negatively.

Let's look at the new anti-patterns.

HTTP/1.x  HTTP/2

What has Changed?

Concatenation of Files

This was a workaround for the lack of parallelism in HTTP/1.x to reduce requests;

Combining multiple files into one and fetch with one request.

Need to wait of the entire file/response to arrive

Requests are cheap!

- Structure code to only deliver what's needed
- No need for additional build process steps
- Optimise caching policies depending on change frequency of files

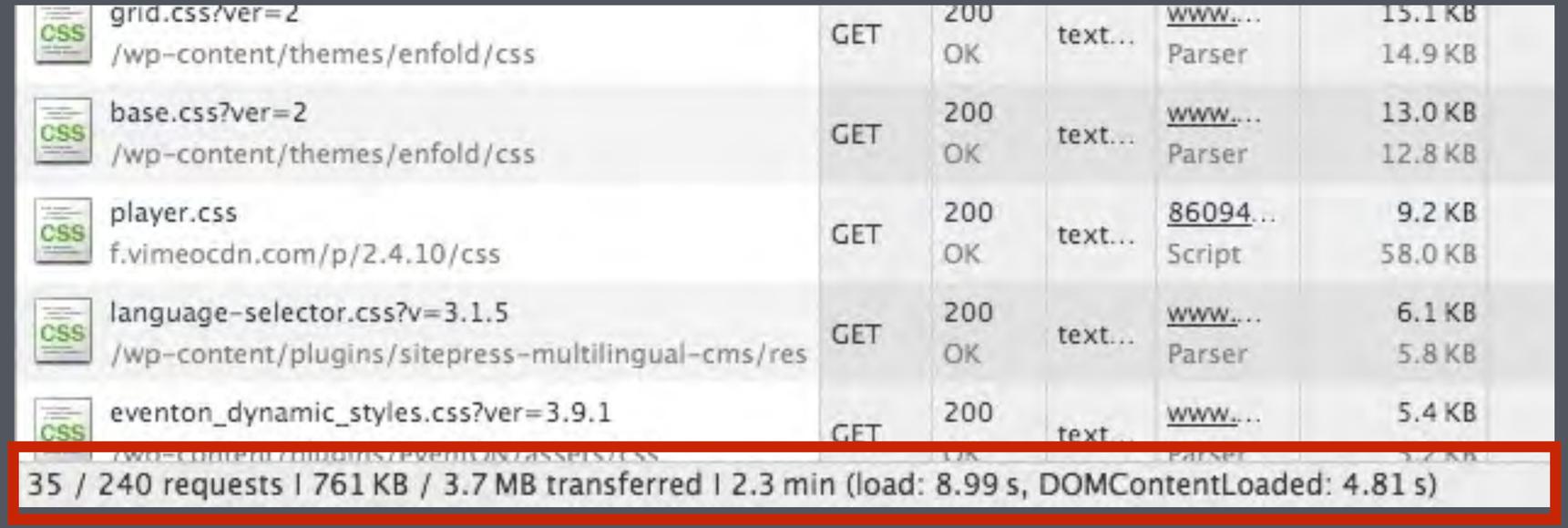

grid.css?ver=2
/wp-content/themes/enfold/css
GET
200
OK
text...
www...
Parser
15.1 KB
14.9 KB
base.css?ver=2
/wp-content/themes/enfold/css
GET
200
OK
text...
www...
Parser
13.0 KB
12.8 KB
player.css
f.vimeocdn.com/p/2.4.10/css
GET
200
OK
text...
86094...
Script
9.2 KB
58.0 KB
language-selector.css?v=3.1.5
/wp-content/plugins/sitepress-multilingual-cms/res
GET
200
OK
text...
www...
Parser
6.1 KB
5.8 KB
eventon_dynamic_styles.css?ver=3.9.1
GET
200
text...
www...
5.4 KB
35 / 240 requests | 761 KB / 3.7 MB transferred | 2.3 min (load: 8.99 s, DOMContentLoaded: 4.81 s)

Image Sprites

Thanks to the new multiplexing ability of HTTP/2 resources don't need to be queued anymore.

Nevertheless, depending on the kind of image, and how they are used, spriting can still be the better option in regards to compression and file size.

Inline Images

Another workaround for the lack of parallelism in HTTP/1.x

Besides increasing the file size of stylesheets etc., the resource can’t be cached and asset re-use will create unnecessary overhead

Prioritisation features of HTTP/2 can’t be used

Domain Sharding

And one more workaround for the lack of multiplexing in HTTP/1.x

Browsers can handle ~6 connections per origin, but domain sharding allows us to (theoretically) extend this to an unlimited amount of connections.

Domain sharding will have a negative impact when used with HTTP/2.

Source

	HTTP1.x	HTTP2
Reduce DNS lookups	✓	✓
Reuse TCP connections	✓	✓
Use a Content Delivery Network	✓	✓
Minimize number of HTTP redirects	✓	✓
Eliminate unnecessary request bytes	✓	✓
Compress assets during transfer	✓	✓
Cache resources on the client	✓	✓
Eliminate unnecessary resources	✓	✓
Apply domain sharding	Revisit (max 2)	Remove
Concatenate resources	Careful & consider caching	Remove
Inline resources	Careful & consider caching	Remove (Server Push)

Source

Getting to HTTP/2

Make the move to TLS & add a secure connection to your site

(This can be done at any time and brings some additional benefits, even without HTTP/2)

Make sure your server supports HTTP/2

(Confirm with your hosting provider, roll your own or use a HTTP/2 supporting CDN service)

Prepare your assets & adjust the build process for HTTP/2

(Adjust to output the required files that best suit your needs and test your choices)

Check Analytics & confirm your user's browser support

(This could affect users with older browsers negatively, and check for majority support)

Implement your favourite HTTP/2 best practices and adjust your caching policies

(Measure your performance before and after the update and share your results with the world!!)

谢谢!