HTTP/2: What You Need to Know

Robert Boedigheimer

@boedie

About Me

- Web developer since 1995
- Pluralsight Author
- 3rd Degree Black Belt, Tae Kwon Do
- ASP.NET MVP

- boedie@outlook.com
- @boedie
- weblogs.asp.net/boedie

HTTP 0.9

- 1991, http://tinyurl.com/5obj3z
- Sir Tim Berners-Lee, CERN

- Text based request/response

- GET (only method) and HTML (only response type)
- Closes connection after response

HTTP 1.0

- 1996, https://tools.ietf.org/html/**rfc1945**

- “Informational” RFC (not a standard)
  - Compilation of best practices

- Request/response headers
- Any type of response (images, text file, etc.)

HTTP 1.1

- **1999**, https://tools.ietf.org/html/**rfc2616**

- Persistent Connections (*Keep Alive*)
- Host Headers
- Chunked transfer encoding
- 100 Continue Status

- HUGE success!

Fiddler

- Tracing tool built specifically for HTTP
  - Shows complete request and response
  - Proxy
  - http://fiddler2.com (free)

- Eric Lawrence (@ericlaw)

- **.NET framework needs to support ALPN!!** (need for HTTP/2)

Problems with HTTP 1.1

- Wasn’t designed for todays web pages
  - 100+ requests and 2 MB+ for a single page! (Httparchive.org)

- Requires multiple connections
- Head of Line Blocking
- Lack of prioritization
- Verbose headers

Requires Multiple Connections (HTTP 1.1)

- Single active request/response on a given connection

- Most browsers use up to ~6 connections per host
  - Uses resources
  - Takes time to establish and be efficient
    - 3 way handshake
    - TCP Slow Start

Head of Line Blocking (HTTP 1.1)

- Serial request(s) and response(s)
  - Slow response blocks all other requests and responses on that connection

- *HTTP Pipelining*
  - *Submit multiple requests simultaneously*
  - *Not used*

Lack of Prioritization (HTTP 1.1)

- No direct way to specify desired order of responses
- Browsers need to decide how to best use their limited number of connections and what to request first
  - CSS
  - JavaScript
  - Images

Verbose Headers (HTTP 1.1)

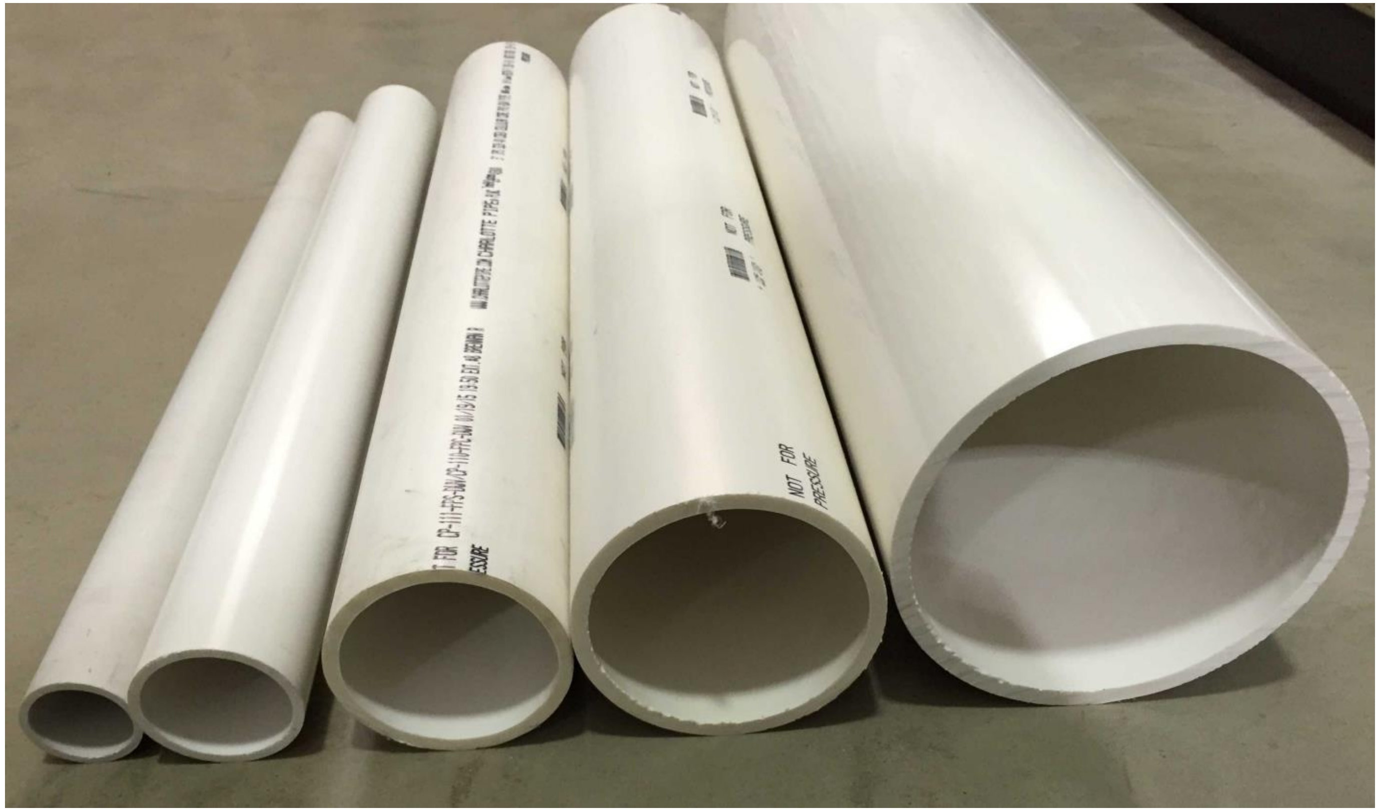
- No header compression

- Repeated headers sent for multiple requests to same host
  - **Cookie**
  - User-Agent
  - Accept-language
  - Accept-encoding
  - Referer
  - …

Bandwidth

- Measured in units of *bits per seconds* (bps)

- *Relatively* easy to add more



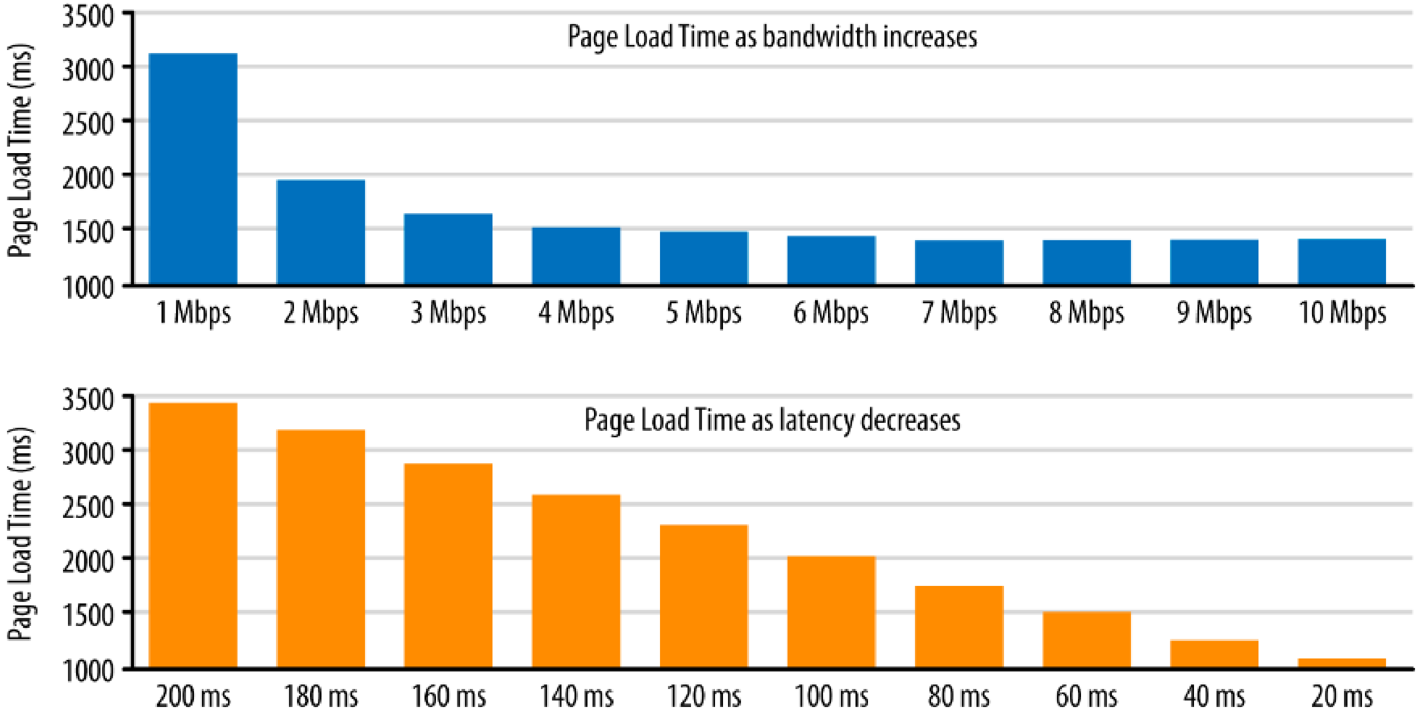
Latency

- Measured in milliseconds (ms)

- Time takes for packet to get to destination
    - **Propagation**
    - Transmission
    - Processing

- Extremely difficult to improve, try to avoid!

Latency vs Bandwidth impact on Page Load Time



Single digit % perf improvement after 5 Mbps

Linear improvement in page load time!

http://tinyurl.com/omyuh3x, Ilya Grigorik

“Bandwidth Doesn’t Matter Much” - http://tinyurl.com/btqpclr

SPDY

- 2009, Experimental…
- http://tinyurl.com/3nh7rto
- Modifies how requests and responses are sent over the wire
- Required HTTPS

- Features
  - Single connection
  - Header compression
  - Request prioritization
  - Server Push

HTTP/2 Process

- IETF (Internet Engineering Task Force) – NOT W3C
  - http://www.ietf.org/

- HTTP Working Group – HTTPbis
  - https://httpwg.github.io/
  - 2012
  - Initially based on SPDY

- HTTP/2 - May 2015, https://tools.ietf.org/html/**rfc7540**
- HPACK - May 2015, https://tools.ietf.org/html/**rfc7541**

HTTP/2 Goals

- Minimize impact of latency
- Avoid head of line blocking
- Use a single connection (per host)
- **Keep HTTP 1.1 semantics**!
  - Methods, status, headers

- DON’T NEED TO CHANGE APPLICATION CODE!!
  - Should remove some current workarounds…

HTTP/2 Major Features

- Binary framing layer
- Streams
  - Prioritization and dependencies
- Fully multiplexed on single TCP connection
- Header Compression (HPACK)
- *Server Push*

Binary Framing Layer

- Previously text based protocol
  - Very easy to review and troubleshoot

- Binary protocols are much easier to parse, less error prone
- Need tool support!

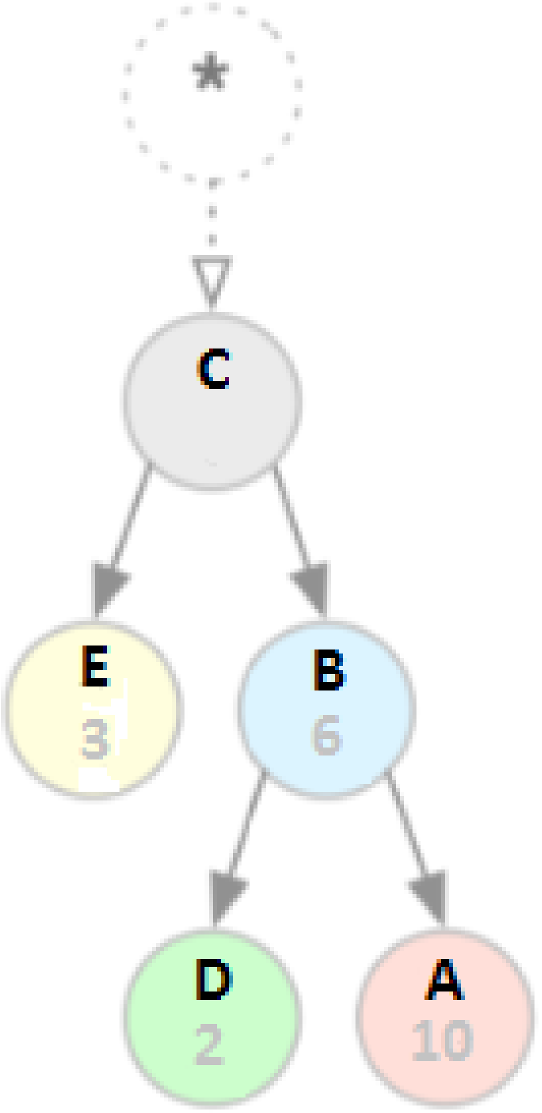
- **Frames**
  - Header
  - Data
  - …

Streams

- Single request/response
- Bidirectional series of **frames**
    - Order of frames is significant
    - Integer identifier

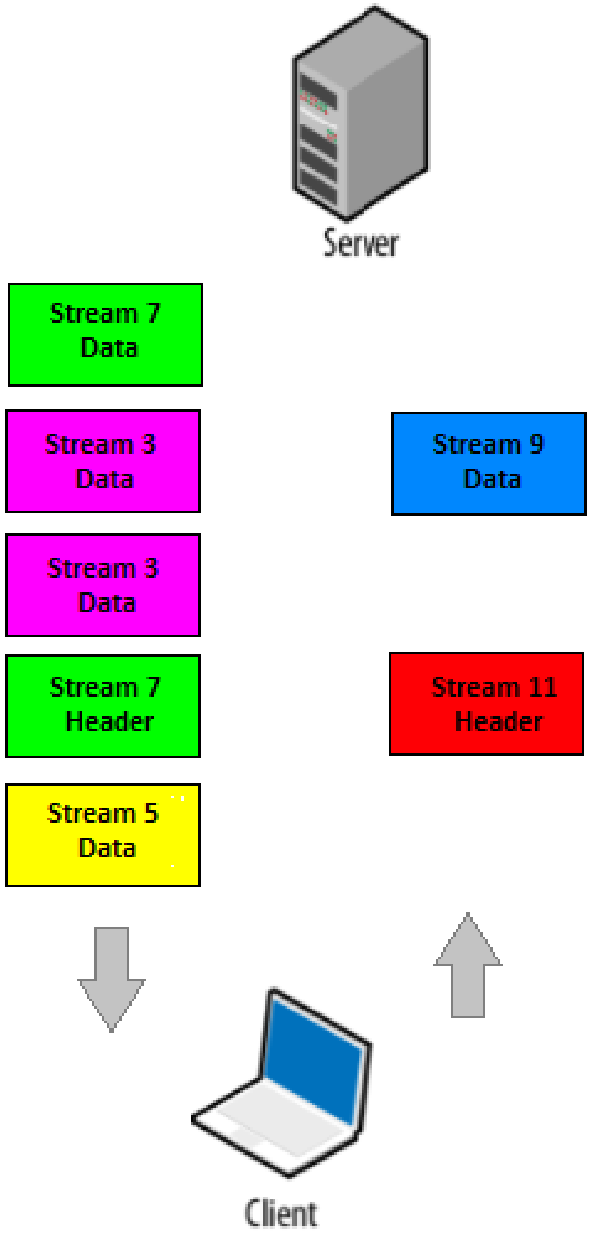
- Client "priority hints"
    - Dependencies
    - Weights

    - Can be updated at any point



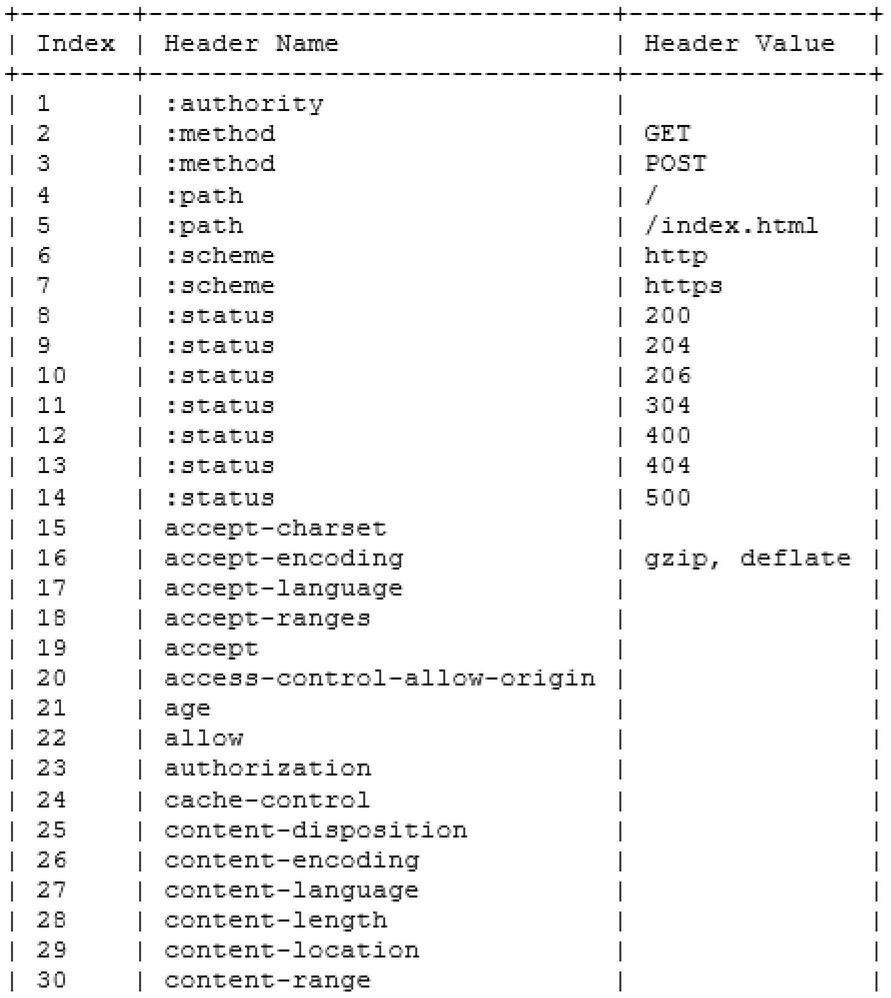
Single TCP Connection (per host)

- HTTP 1.1 browsers use ~6 connections per host
  - Serial requests and responses
  - Need to decide which requests to make first (HOL blocking)
- Multiplexing of request and response frames from various streams
- Uses less resources, more efficient



Header Compression (HPACK)

- https://tools.ietf.org/html/**rfc7541**
- Techniques
  - Index value for common headers/values
  - Indexed list of previously sent headers
  - Huffman encoding to compress a value
- Static table
  - Predefined common headers (values)
- Dynamic table
  - Maximum size



Index	Header Name	Header Value
1	:authority	
2	:method	GET
3	:method	POST
4	:path	/
5	:path	/index.html
6	:scheme	http
7	:scheme	https
8	:status	200
9	:status	204
10	:status	206
11	:status	304
12	:status	400
13	:status	404
14	:status	500
15	accept-charset	
16	accept-encoding	gzip, deflate
17	accept-language	
18	accept-ranges	
19	accept	
20	access-control-allow-origin	
21	age	
22	allow	
23	authorization	
24	cache-control	
25	content-disposition	
26	content-encoding	
27	content-language	
28	content-length	
29	content-location	
30	content-range	

Header Compression (HPACK) (cont.)

:method	GET
:scheme	HTTP
:path	/
:user-agent	…Edge/12.10240
:accept-encoding	gzip, deflate
:host	twitter.com
:accept-language	en-US
:rjb-hdr	14534

→

2	
6	
4	
58	*…Edge/12.10240*
16	
38	*twitter.com*
17	*en-US*
63	*rjb-hdr*
64	*14534*

- Future requests the compressed values would not be sent if the same

Server Push

- Server can anticipate what client will need next
    - How?
- Same origin restrictions

- “**Better Inlining**”
    - Resources are cacheable
    - No added page weight
    - Client can reject (RST_STREAM)

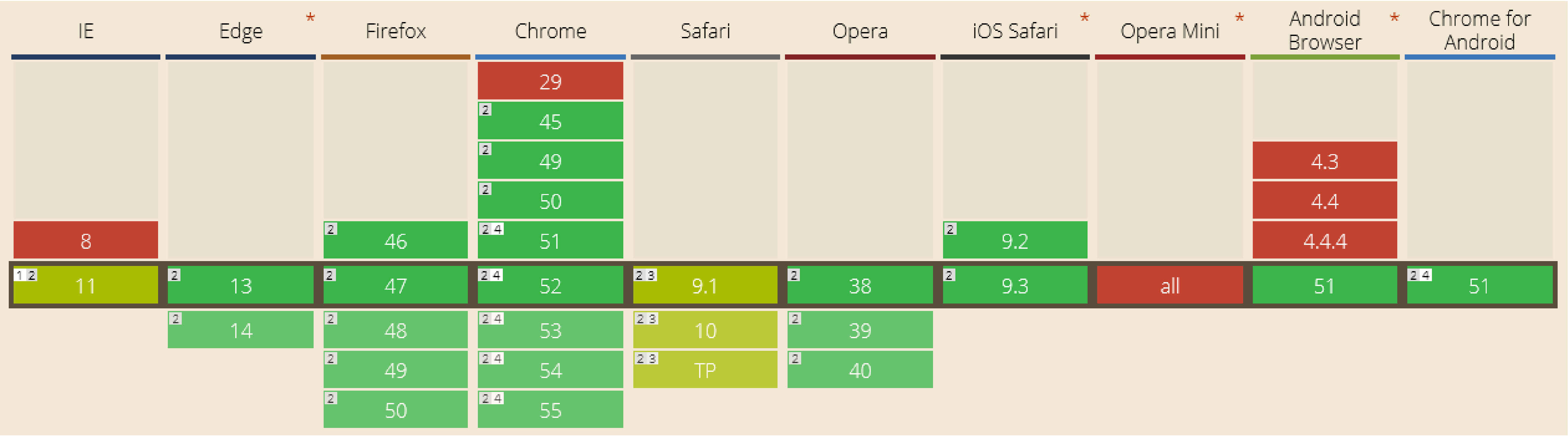
- *Experimental…*

Require HTTPS?

- NOT required in HTTP/2 RFC
  - TLS 1.2+
  - Blacklist of cipher suites

- Most browsers will only implement with HTTPS
  - Avoid problems with new protocol and “middleboxes”
    - Proxy servers
    - Firewalls
  - Improve security

Browser Support

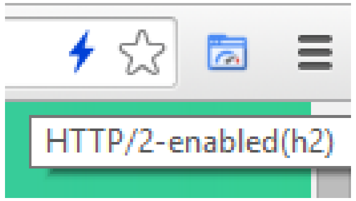


IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			29						
			[2] 45						
			[2] 49					4.3	
			[2] 50					4.4	
8		[2] 46	[2 4] 51			[2] 9.2		4.4.4	
[1 2] 11	[2] 13	[2] 47	[2 4] 52	[2 3] 9.1	[2] 38	[2] 9.3	all	51	[2 4] 51
	[2] 14	[2] 48	[2 4] 53	[2 3] 10	[2] 39				
		[2] 49	[2 4] 54	[2 3] TP	[2] 40				
		[2] 50	[2 4] 55						

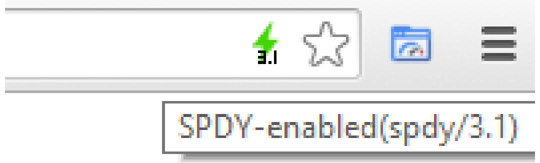
http://caniuse.com/#feat=http2

Implementations

- http://tinyurl.com/mgbmq5c
- IIS 10 (Windows 10 and Windows Server 2016)
- Indicators
  - Chrome and Firefox extensions



HTTP/2-enabled(h2)



SPDY-enabled(spdy/3.1)

Expectations

- “HTTP/2 isn’t magic Web performance pixie dust; you can’t drop it in and expect your page load times to decrease by 50%”
    - Mark Nottingham

- Should help the most in high latency networks or lots of requests to same hosts

- ~5-15% performance improvement (no changes to the site)

Performance Techniques to Avoid

- **Bundling JavaScript and CSS files**
- CSS Sprites
- Domain Sharding
  - Using multiple host names so browsers uses more connections
- Inlining (*Server Push*)
  - Data URIs, CSS, JavaScript

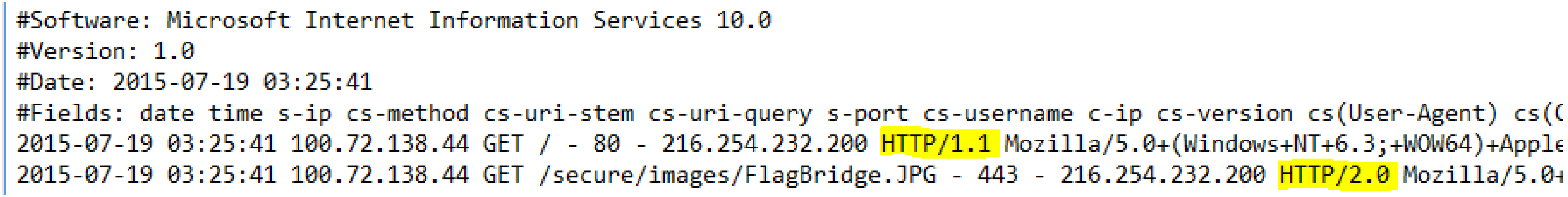
Performance Techniques to Continue

- Golden Rules
    - *Make fewer HTTP requests*
    - Send as little as possible
    - Send it as infrequently as possible

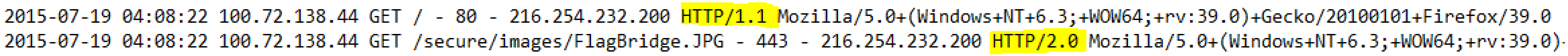
- Minification
- Compression
- Expirations
- CDN (Content Delivery Network)

Strategy

- **CDN** (latency)
  - All static resources (JavaScript, CSS, images, Web Fonts)
    - Minified
    - Bundled (HTTP 1.1) and non-bundled (HTTP/2)
    - HTTPS



```
#Software: Microsoft Internet Information Services 10.0
#Version: 1.0
#Date: 2015-07-19 03:25:41
#Fields: date time s-ip cs-method cs-uri-stem cs-uri-query s-port cs-username c-ip cs-version cs(User-Agent) cs(C
2015-07-19 03:25:41 100.72.138.44 GET / - 80 - 216.254.232.200 HTTP/1.1 Mozilla/5.0+(Windows+NT+6.3;+WOW64)+Apple
2015-07-19 03:25:41 100.72.138.44 GET /secure/images/FlagBridge.JPG - 443 - 216.254.232.200 HTTP/2.0 Mozilla/5.0+
```



```
2015-07-19 04:08:22 100.72.138.44 GET / - 80 - 216.254.232.200 HTTP/1.1 Mozilla/5.0+(Windows+NT+6.3;+WOW64;+rv:39.0)+Gecko/20100101+Firefox/39.0
2015-07-19 04:08:22 100.72.138.44 GET /secure/images/FlagBridge.JPG - 443 - 216.254.232.200 HTTP/2.0 Mozilla/5.0+(Windows+NT+6.3;+WOW64;+rv:39.0)
```

Strategy (cont.)

- Optimize for each HTTP version
  - *Detect protocol version*

- Options for detection
  - Load balancer detect HTTP/2 and pass custom header
  - UA sniffing
  - Web Server support HTTP/2
    - Upgrade web server (Windows Server 2016)
    - Use HTTPS everywhere

Summary

- Ready for production

- HTTP/2 Major Features
  - Binary framing layer
  - Streams
  - Fully multiplexed on single TCP connection
  - Header Compression (HPACK)
  - *Server Push*

Resources

- https://http2.github.io/
- https://httpwg.github.io/
- https://www.mnot.net/blog/

- “High Performance Browser Networking” by Ilya Grigorik
  - Hpbn.co/http2
- “HTTP The Definitive Guide” by David Gourley and Brian Totty (HTTP 1.1)

Questions

- boedie@outlook.com
- @boedie
- weblogs.asp.net/boedie