Impacts of the HTTP/2 protocol for large scale web environments

Martin Leucht <martin.leucht@os3.nl>
James Gratchoff <james.gratchoff@os3.nl>

Large Infrastructure Administration
Jaap van Ginkel
Master System and Network Engineering 2015
UvA

Introduction

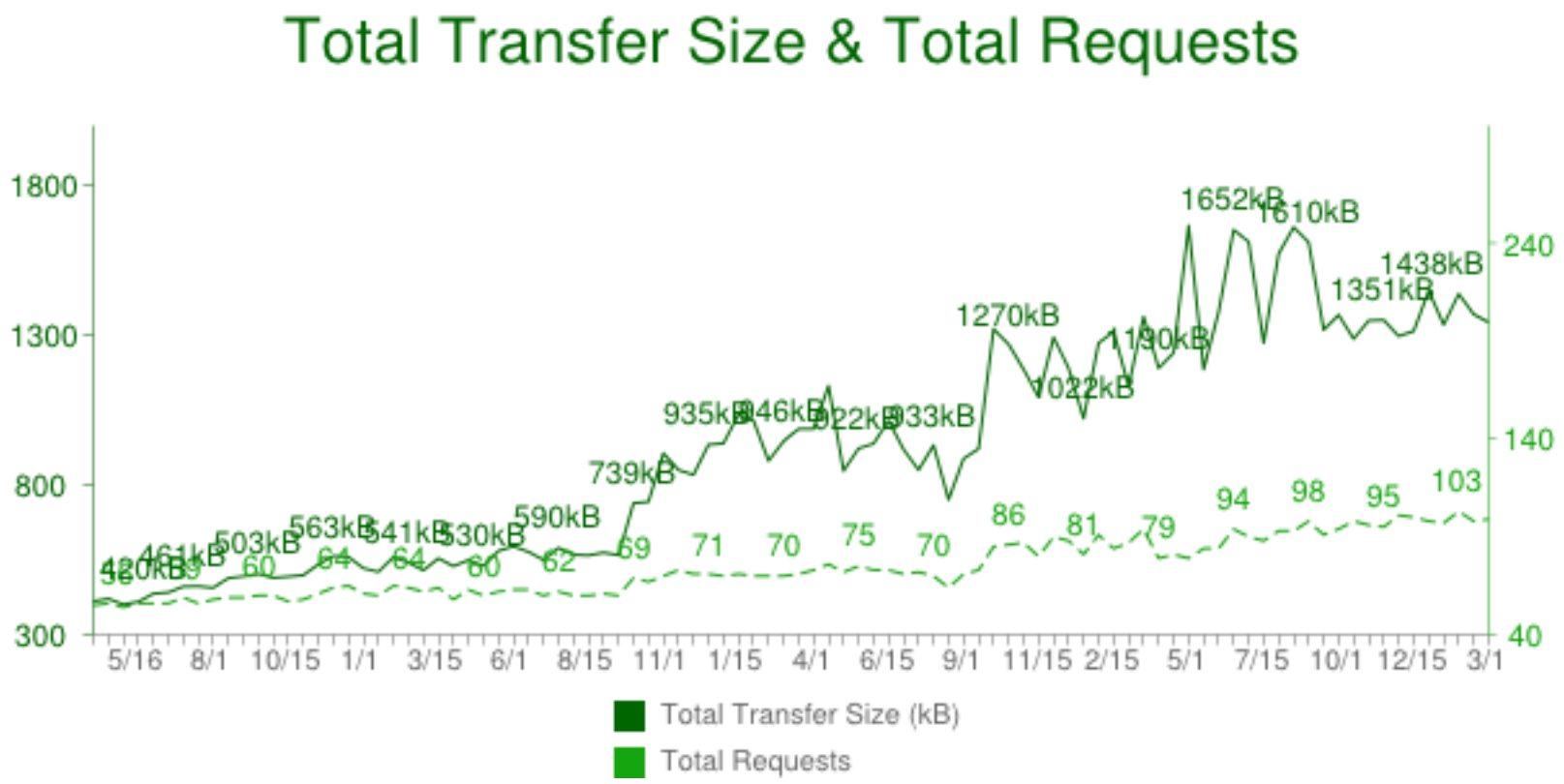
- HTTP/2 sp
  on the 18th
- On the way
- 16 years of

I E T F

SG

Engineering
Steering Group

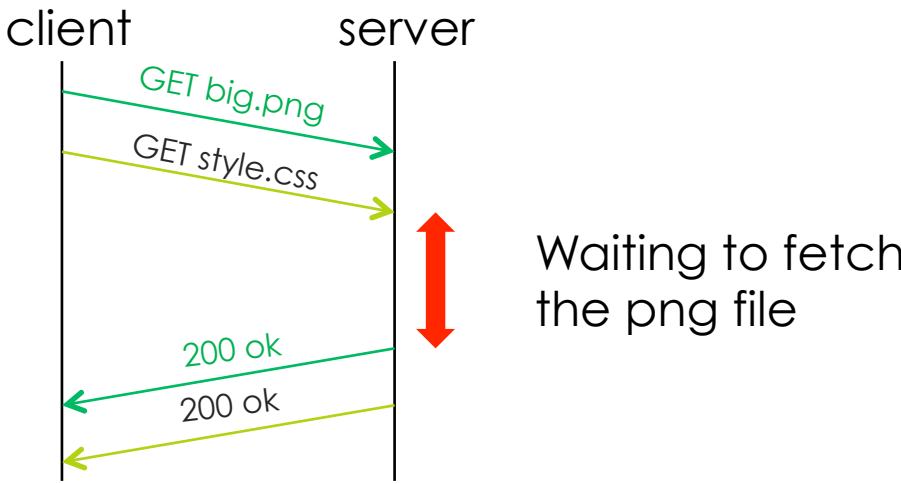
Why change?



Source: http://httparchive.org

Problems

Bad usage of TCP:
Head of line blocking



Headers size:
Repetition of header

```
Hypertext Transfer Protocol
  GET /images/tri-rt-t-14x28.gif HTTP/1.1\r\n
  Host: httparchive.org\r\n
  Connection: keep-alive\r\n
  Pragma: no-cache\r\n
  Cache-Control: no-cache\r\n
  Accept: image/webp,*/*;q=0.8\r\n
  User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
  Referer: http://httparchive.org/\r\n
  Accept-Encoding: gzip, deflate, sdch\r\n
  Accept-Language: en-US,en;q=0.8,fr;q=0.6\r\n
  Cookie: __utmt=1; __utma=108599965.1226105352.1426776704.142677670
  \r\n
```

```
Hypertext Transfer Protocol
  GET /images/tri-lft-t-14x28.gif HTTP/1.1\r\n
  Host: httparchive.org\r\n
  Connection: keep-alive\r\n
  Pragma: no-cache\r\n
  Cache-Control: no-cache\r\n
  Accept: image/webp,*/*;q=0.8\r\n
  User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
  Referer: http://httparchive.org/\r\n
  Accept-Encoding: gzip, deflate, sdch\r\n
  Accept-Language: en-US,en;q=0.8,fr;q=0.6\r\n
  Cookie: __utmt=1; __utma=108599965.1226105352.1426776704.1426776704
  \r\n
```

Workarounds

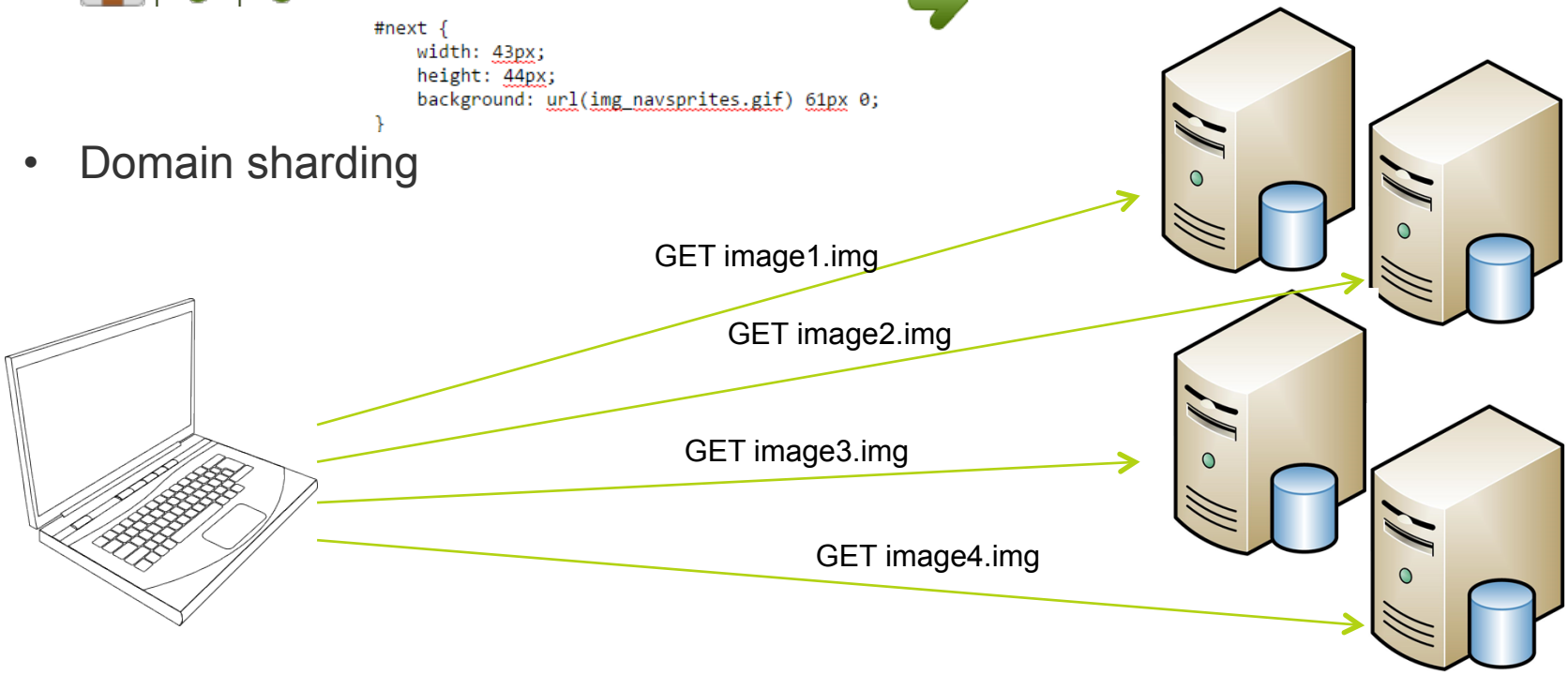
- Spriting

TCP works better for larger files. Use one single image for different one.

```
#home {
    width: 46px;
    height: 44px;
    background: url(img_navsprites.gif) 30px 0;
}

#next {
    width: 43px;
    height: 44px;
    background: url(img_navsprites.gif) 61px 0;
}
```

- Domain sharding



Concatenation, Inlining (base 64)…

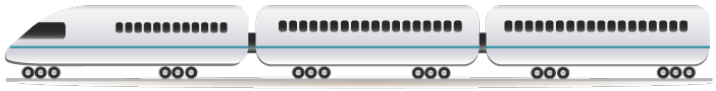
HTTP/2

Advantages:

- Binary protocol
- one TCP connection
- Streams
- Header compression (HPACK)
- Multiplexing with prioritization and flow control
- Server push
- TLS mandatory (Google Chrome, Firefox)

HTTP/1.1

HTTP/2



Research questions

- How do the new features of the HTTP/2 protocol improve the performance for frequently visited webpages/webserver?

- What are possible drawbacks that can occur for large web service providers when switching from HTTP/1.1 to HTTP/2? And what is the impact on the infrastructure?

- What is the difference between HTTP/1.1 and HTTP/2 in terms of:
  - Bandwidth and CPU utilization
  - Header size
  - mean time taken for request and response (per location/RTT)

Benchmark setup

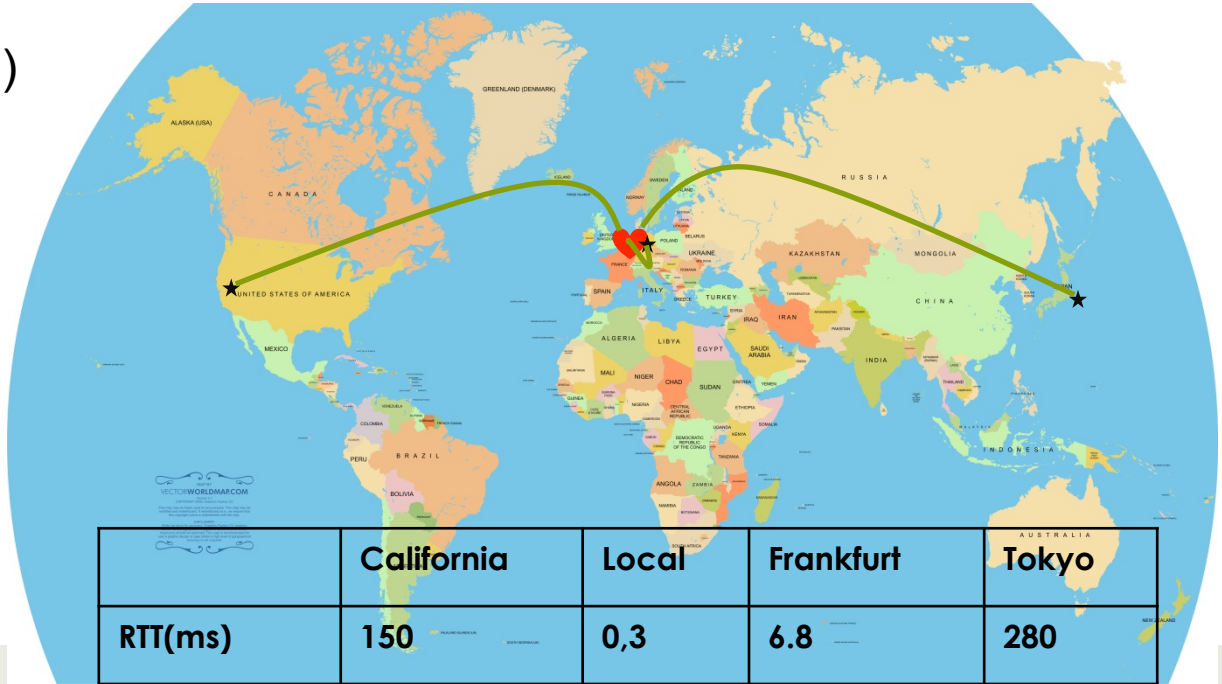
- Webserver:

Ubuntu 14 (8CPUx1,8GHz, 8GB RAM)
Servers:

HTTP/2	Nghttp2
HTTP/1.1	Apache2
Reverse Proxy	Nghttpx

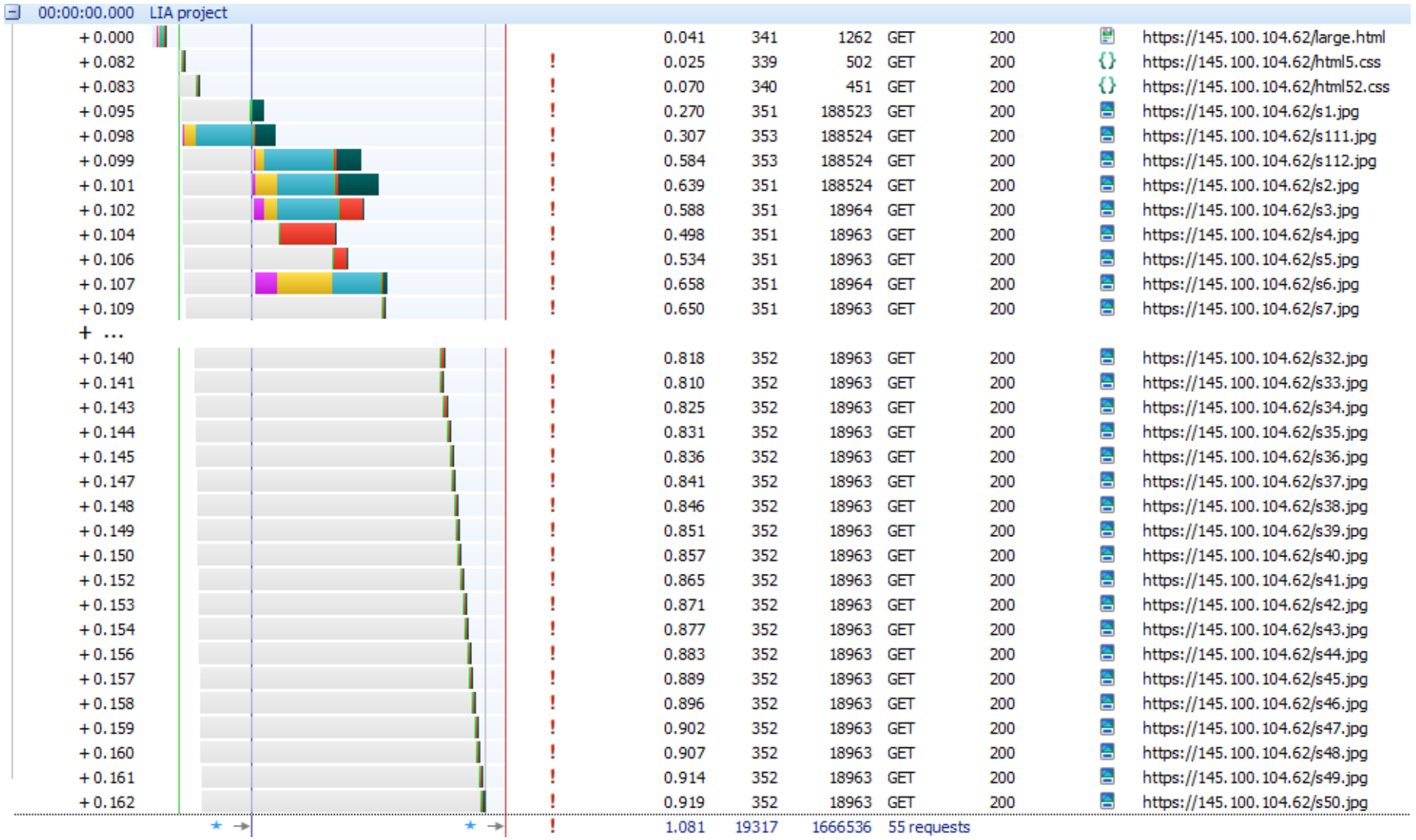
- 4-Clients:

Ubuntu 14 (ec2-Amazon)
4vCPU, 2GB
Benchmark Tool:
H2load



	California	Local	Frankfurt	Tokyo
RTT(ms)	150	0,3	6.8	280

Httpwatch screenshot



00:00:00.000	LIA project							
+ 0.000		0.041	341	1262	GET	200		https://145.100.104.62/large.html
+ 0.082	!	0.025	339	502	GET	200	{}	https://145.100.104.62/html5.css
+ 0.083	!	0.070	340	451	GET	200	{}	https://145.100.104.62/html52.css
+ 0.095	!	0.270	351	188523	GET	200		https://145.100.104.62/s1.jpg
+ 0.098	!	0.307	353	188524	GET	200		https://145.100.104.62/s111.jpg
+ 0.099	!	0.584	353	188524	GET	200		https://145.100.104.62/s112.jpg
+ 0.101	!	0.639	351	188524	GET	200		https://145.100.104.62/s2.jpg
+ 0.102	!	0.588	351	18964	GET	200		https://145.100.104.62/s3.jpg
+ 0.104	!	0.498	351	18963	GET	200		https://145.100.104.62/s4.jpg
+ 0.106	!	0.534	351	18963	GET	200		https://145.100.104.62/s5.jpg
+ 0.107	!	0.658	351	18964	GET	200		https://145.100.104.62/s6.jpg
+ 0.109	!	0.650	351	18963	GET	200		https://145.100.104.62/s7.jpg
+ ...								
+ 0.140	!	0.818	352	18963	GET	200		https://145.100.104.62/s32.jpg
+ 0.141	!	0.810	352	18963	GET	200		https://145.100.104.62/s33.jpg
+ 0.143	!	0.825	352	18963	GET	200		https://145.100.104.62/s34.jpg
+ 0.144	!	0.831	352	18963	GET	200		https://145.100.104.62/s35.jpg
+ 0.145	!	0.836	352	18963	GET	200		https://145.100.104.62/s36.jpg
+ 0.147	!	0.841	352	18963	GET	200		https://145.100.104.62/s37.jpg
+ 0.148	!	0.846	352	18963	GET	200		https://145.100.104.62/s38.jpg
+ 0.149	!	0.851	352	18963	GET	200		https://145.100.104.62/s39.jpg
+ 0.150	!	0.857	352	18963	GET	200		https://145.100.104.62/s40.jpg
+ 0.152	!	0.865	352	18963	GET	200		https://145.100.104.62/s41.jpg
+ 0.153	!	0.871	352	18963	GET	200		https://145.100.104.62/s42.jpg
+ 0.154	!	0.877	352	18963	GET	200		https://145.100.104.62/s43.jpg
+ 0.156	!	0.883	352	18963	GET	200		https://145.100.104.62/s44.jpg
+ 0.157	!	0.889	352	18963	GET	200		https://145.100.104.62/s45.jpg
+ 0.158	!	0.896	352	18963	GET	200		https://145.100.104.62/s46.jpg
+ 0.159	!	0.902	352	18963	GET	200		https://145.100.104.62/s47.jpg
+ 0.160	!	0.907	352	18963	GET	200		https://145.100.104.62/s48.jpg
+ 0.161	!	0.914	352	18963	GET	200		https://145.100.104.62/s49.jpg
+ 0.162	!	0.919	352	18963	GET	200		https://145.100.104.62/s50.jpg
	!	1.081	19317	1666536	55 requests			

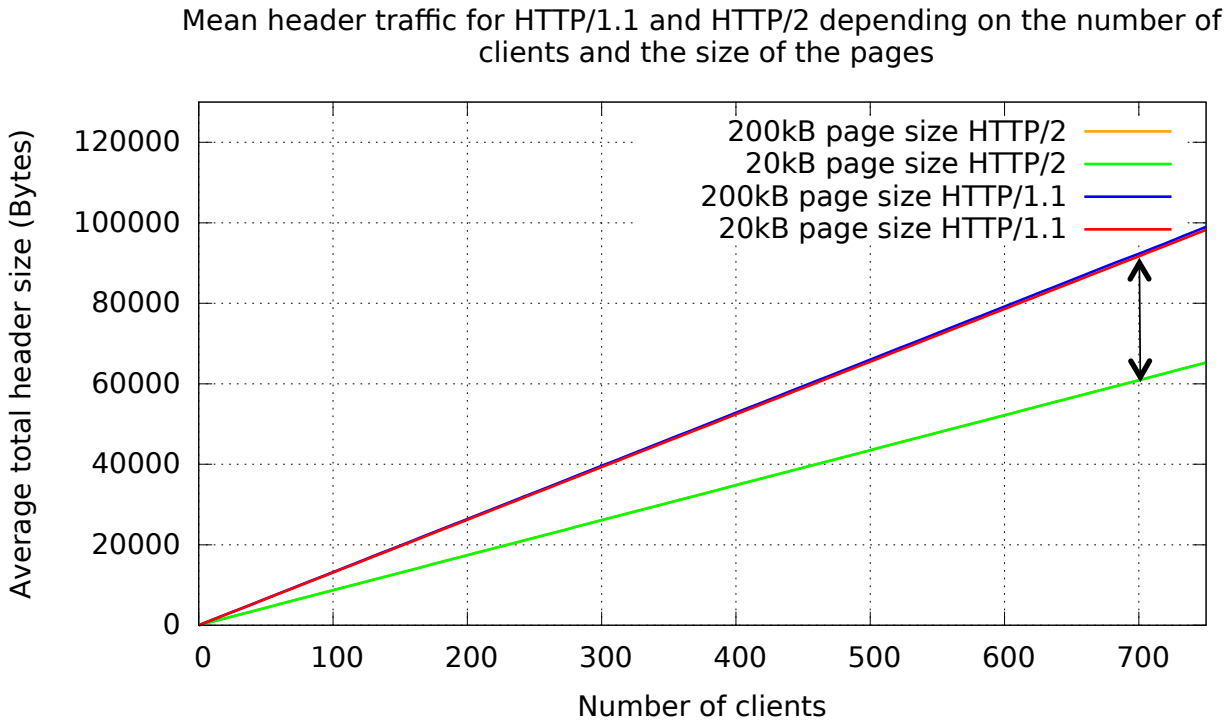
Methods - Measurements

- Test Parameters:
  - HTTP/1.1 and/or HTTP/2 protocol used
  - different locations/distances (RTT)
  - 2 different webpages (size, number of URLs included)
  - increment number of clients/requests in parallel (1…750)
    - recursion depth/repetitions 20

Methods – Data

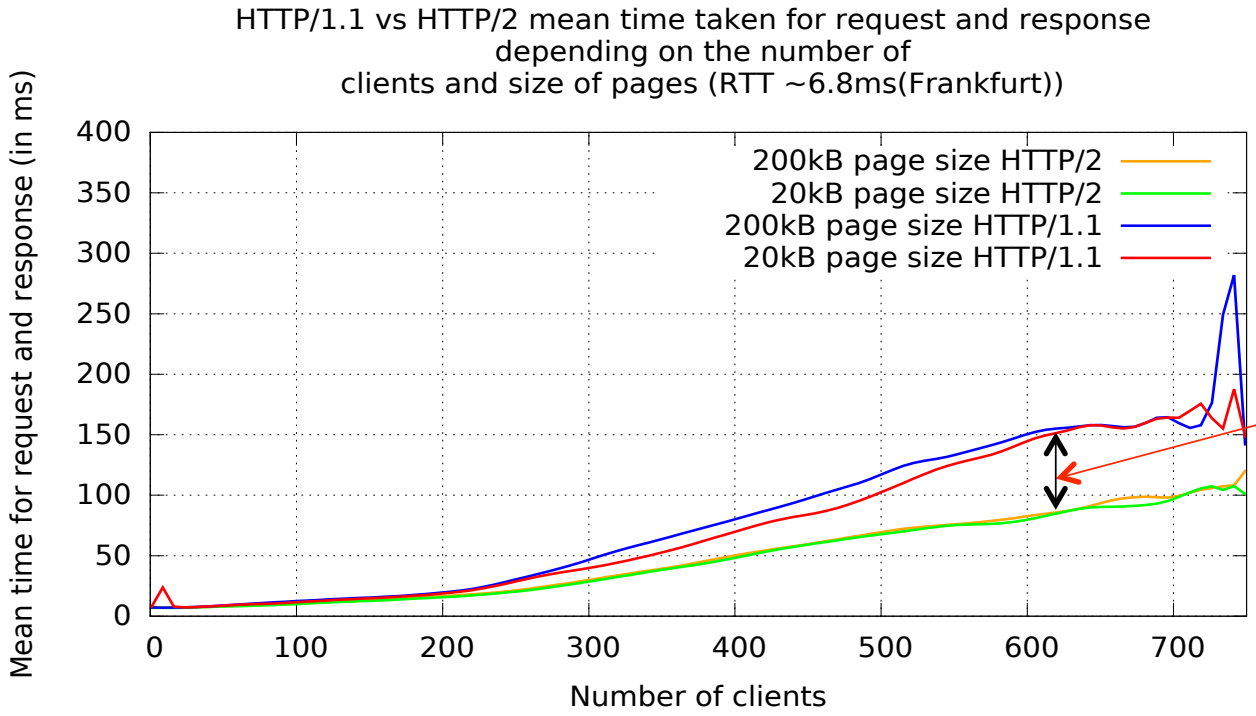
- Mean of request + response time seen from clients (in ms)
- Header sizes (bytes)
- Failed requests
- Resource Monitoring of web server/reverse proxy:
  - traffic (bps) inbound/outbound
  - CPU utilization
  - number of TCP sockets/connections

Results (HTTP/1.1 vs. HTTP/2 Header Traffic)

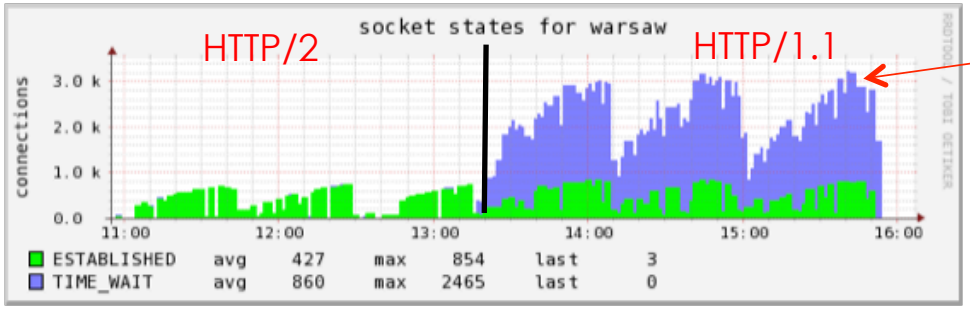


measured ca. 33% less header traffic in HTTP/2 (header compression)

Results (Frankfurt/Germany – RTT ~7ms)

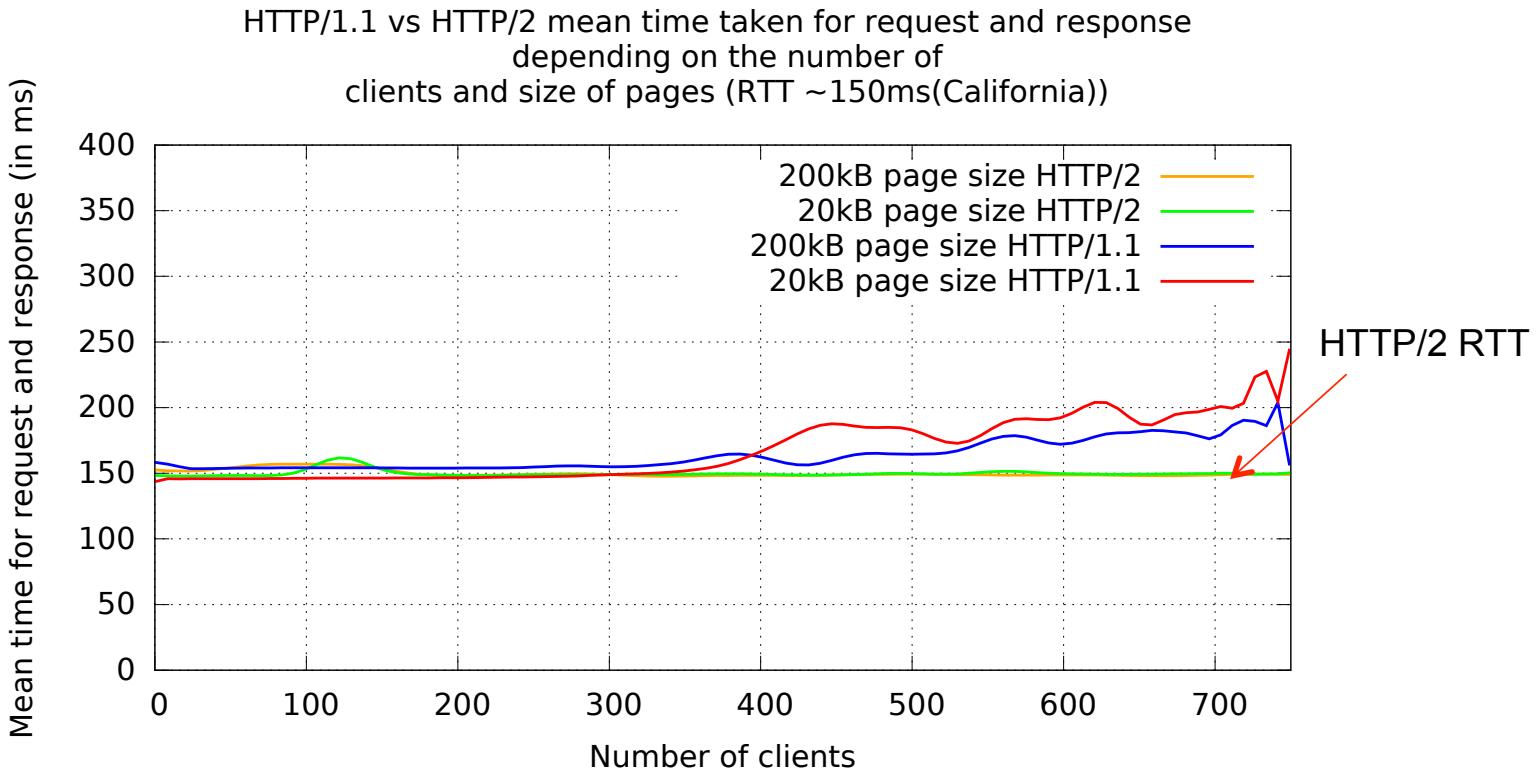


HTTP/2 RTT significantly faster



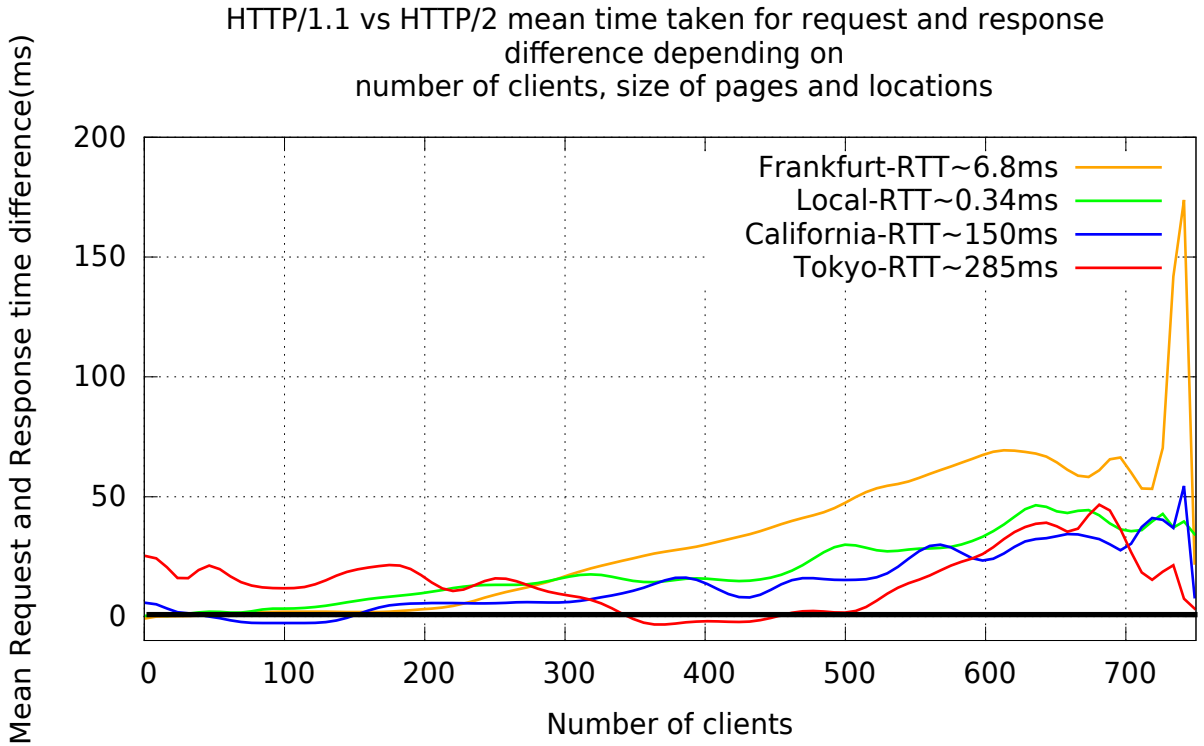
TCP TIME_WAIT connections HTTP/1.1

Results (North Carolina/US – RTT ~150ms)



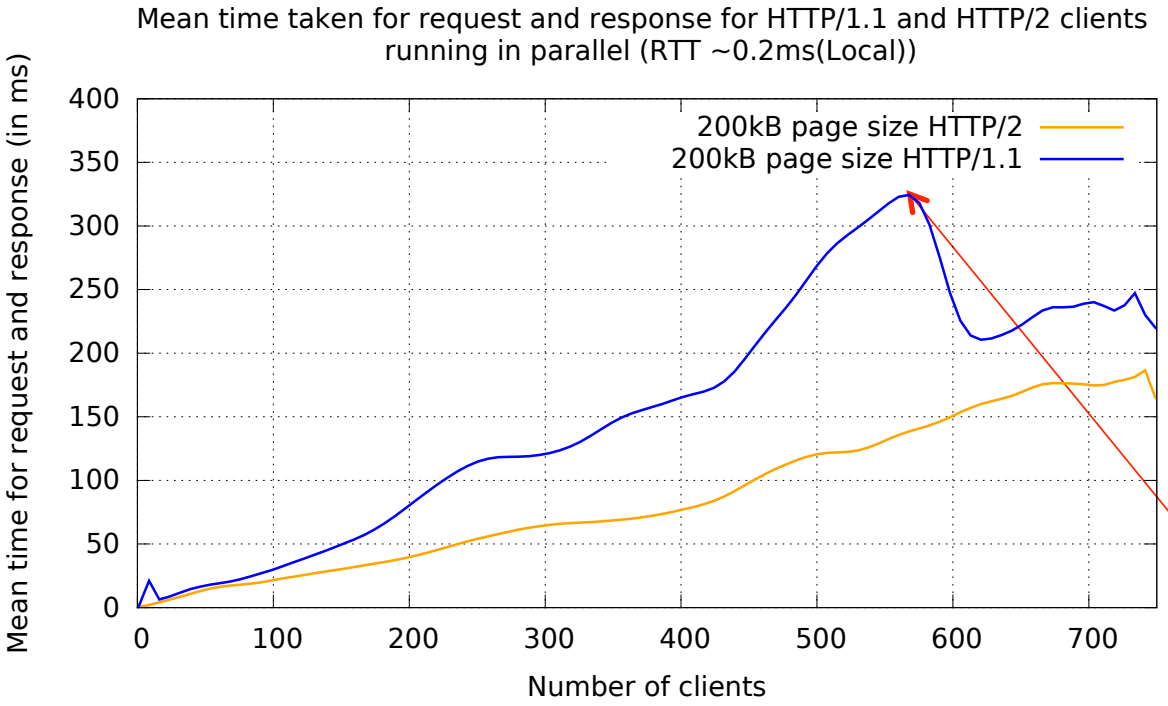
HTTP/2 mean RTT constant (no jitter/packet delay variation)

Results (request time – difference HTTP/2 – HTTP/1.1)

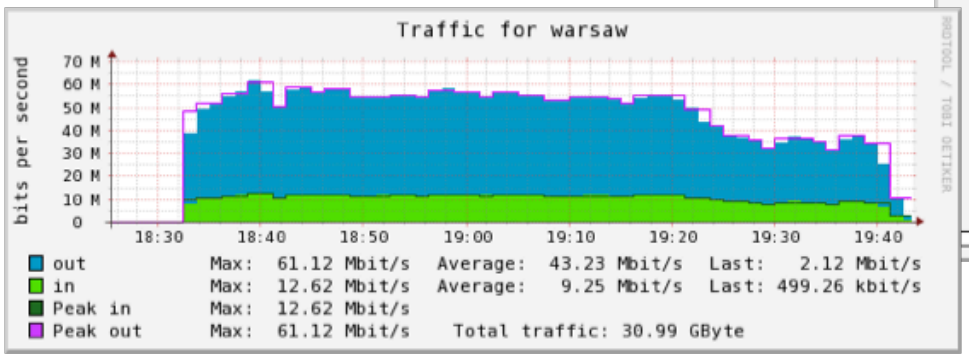


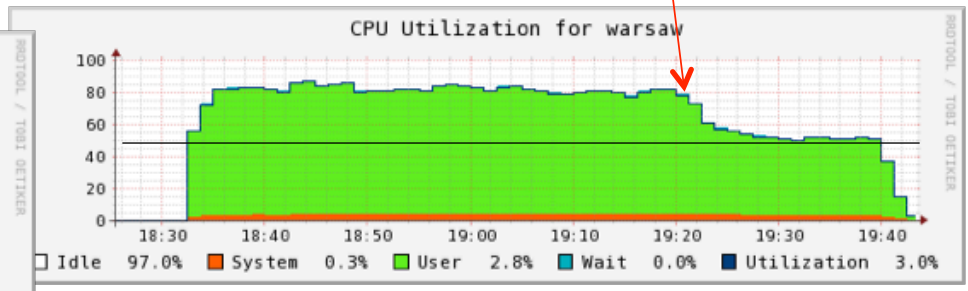
HTTP/2 is in almost all measurements **significantly faster** than HTTP/1.1

Results (HTTP/1.1 and HTTP/2 simultaneous - local)



HTTP/2 tests finished –





Conclusion

- Performance improvements HTTP/2 (compared to HTTP/1.1)
  - significantly less amount header traffic (ca. 33%)
  - significantly faster request/response time – only one TCP session per page request
  - significantly less TCP TIME_WAIT connections on server
    - endpoint (server) blocks current connection before it closes it

Conclusion

- Performance improvements HTTP/2 (compared to HTTP/1.1)
  - significantly less amount header traffic (ca. 33%)
  - significantly faster request/response time – only one TCP session per page request
  - significantly less TCP TIME_WAIT connections on server
    - endpoint (server) blocks current connection before it closes it

- Large Scale Webcontent provider should consider when switching to HTTP/2:
  - Use no longer tweaks like ‘Spriting’ or ‘Sharding’ for HTTP/2
  - Adapt deep packet inspection firewalls rules (data now binary) in order to allow HTTP/2 requests to path through

Results (Request failures)