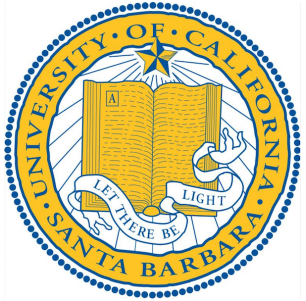
CS 290B

Scalable Internet Services

Andrew Mutz

December 4, 2014

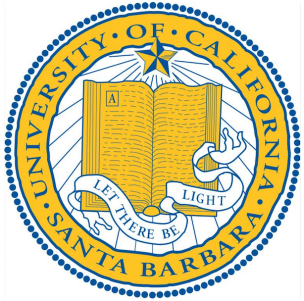


Agenda

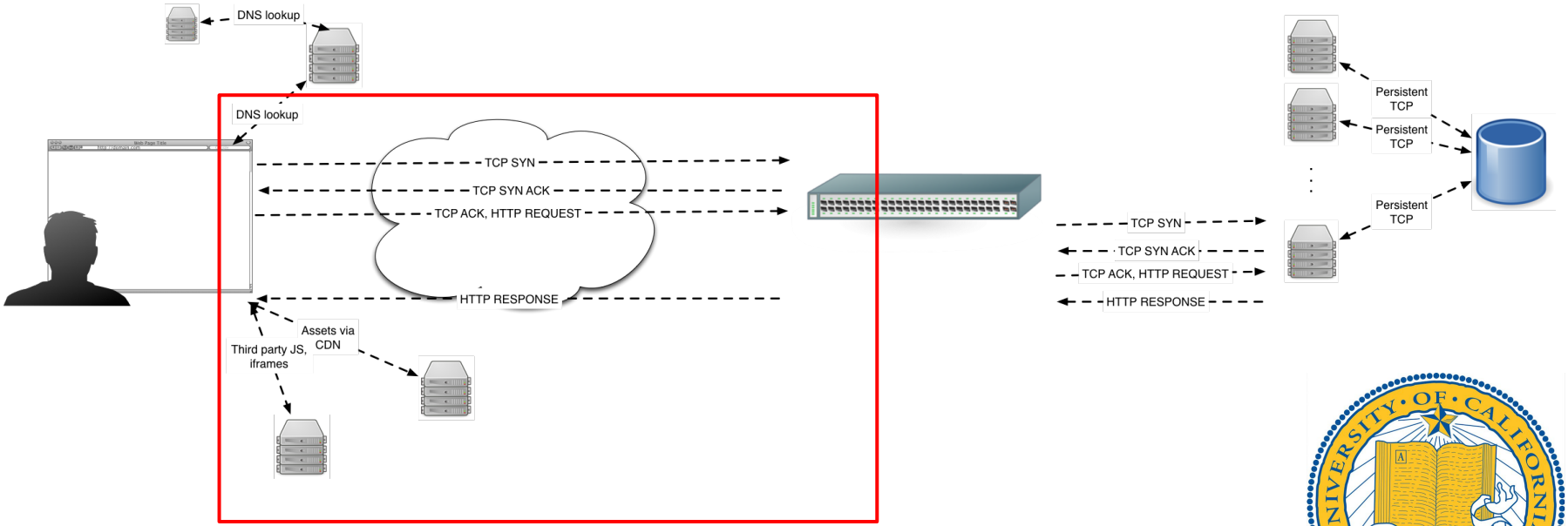
Today's performance problems

Today's performance tricks

Tomorrow's HTTP 2.0

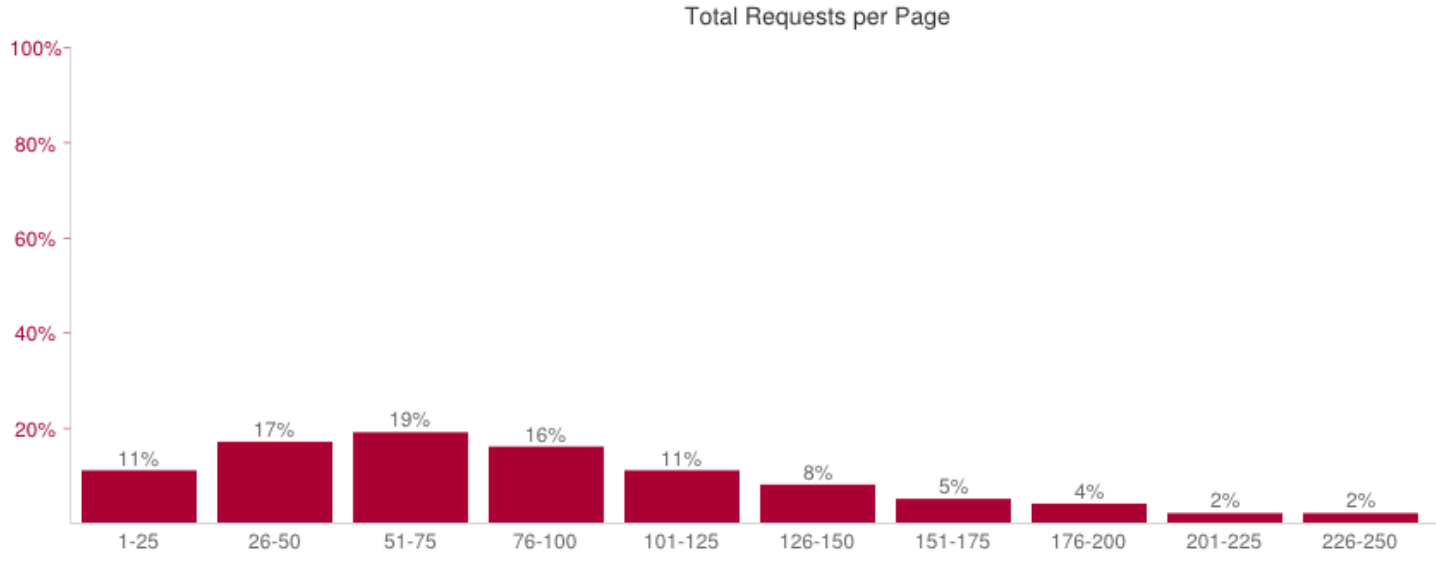


Motivation


DNS lookup
DNS lookup
TCP SYN
TCP SYN ACK
TCP ACK, HTTP REQUEST
HTTP RESPONSE
Assets via CDN
Third party JS, iframes
Persistent TCP
Persistent TCP
Persistent TCP
TCP SYN
TCP SYN ACK
TCP ACK, HTTP REQUEST
HTTP RESPONSE

Today's performance problems

Web pages have many constituent resources



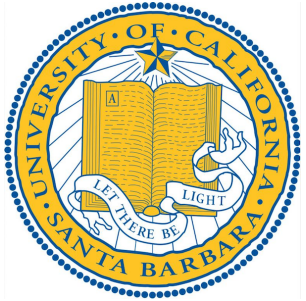
Source: http://httparchive.org/

Today's Performance Problems

Many requests are needed to present today's web pages.

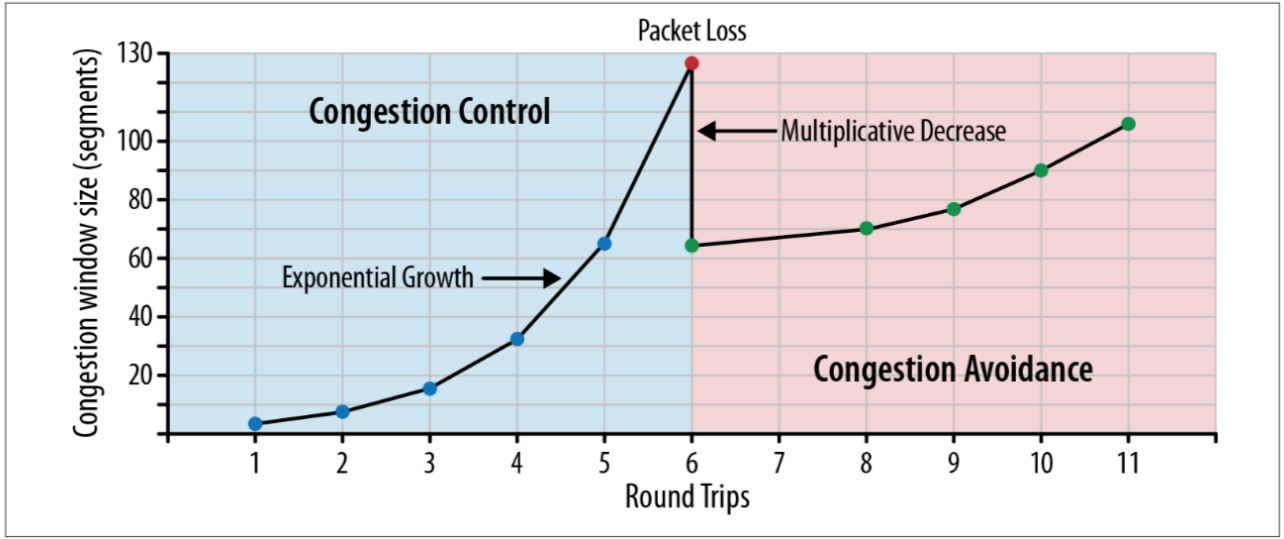
- CSS
- Javascript
- Images

Establishing many TCP connections to serve all these is very slow.



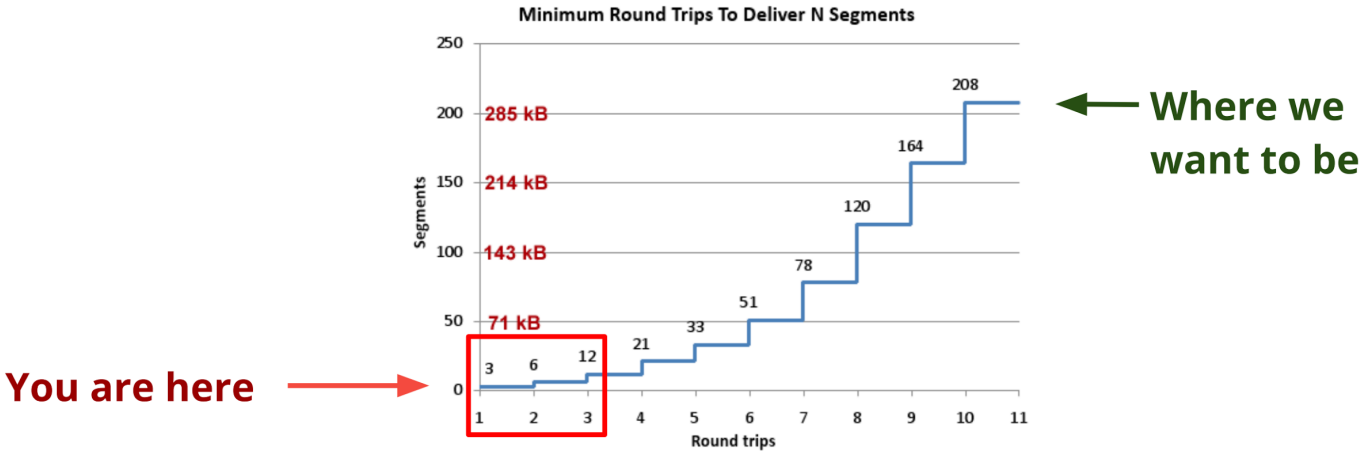
Today's Performance Problems

Establishing many TCP connections to serve all these is very slow.



Today's Performance Problems

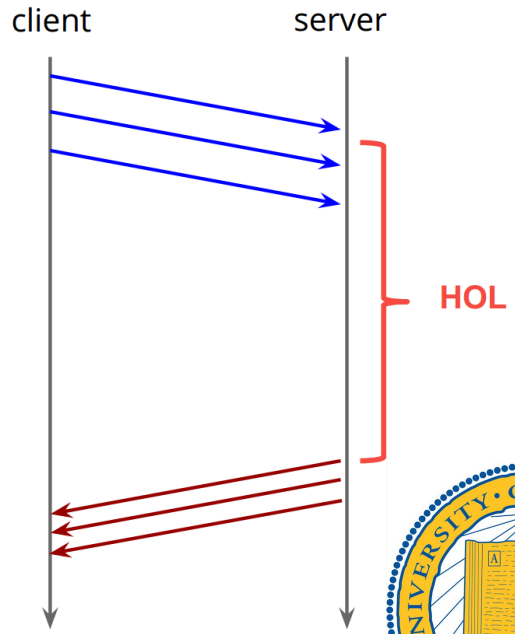
TCP was designed for long-lived flows. HTTP is short and bursty.



Today's Performance Problems

HTTP Keepalive was introduced to help (and it does), but there are problems.

- We can reuse a TCP socket for multiple HTTP requests, but one heavyweight request can affect all others
- This is called Head-of-line blocking

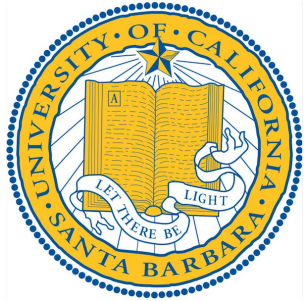


Today's Performance Problems

Additionally, if you look at the data that is being sent, there is a lot of repetition.

```
GET / HTTP/1.1
Host: www.etsy.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_2) AppleWebKit/536.26.14 (
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
DNT: 1
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Cookie: uaid=uaid%3DVdhk5W6sexG-_Y7ZBeQFa3cq7yMQ%26_now%3D1325204464%26_slt%3Ds_L(
Connection: keep-alive
```

525 bytes

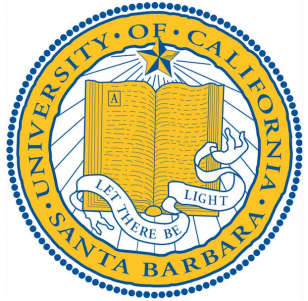


Today's Performance Problems

Additionally, if you look at the data that is being sent, there is a lot of repetition.

```
GET /assets/dist/js/etsy.recent-searches.20121001205006.js HTTP/1.1
Host: www.etsy.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_2) AppleWebKit/536.26.14 (
Accept: */*
DNT: 1
Referer: http://www.etsy.com/
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Cookie: autosuggest_split=1; etala=111461200.1476767743.1349274889.1349274889.134
Connection: keep-alive
```

226 new bytes; 690 total



Today's Performance Problems

Additionally, if you look at the data that is being sent, there is a lot of repetition.

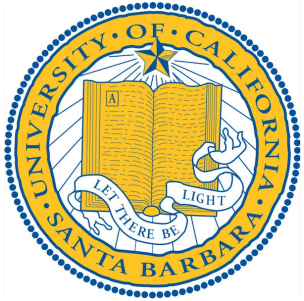
```
GET /assets/dist/js/jquery.appear.20121001205006.js HTTP/1.1
Host: www.etsy.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_2) AppleWebKit/536.26.14 (
Accept: */*
DNT: 1
Referer: http://www.etsy.com/
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Cookie: autosuggest_split=1; etala=111461200.1476767743.1349274889.1349274889.134
Connection: keep-alive
```

14 new bytes; 683 total

Today's Performance Tricks

Many techniques are used to reduce the number of requests.

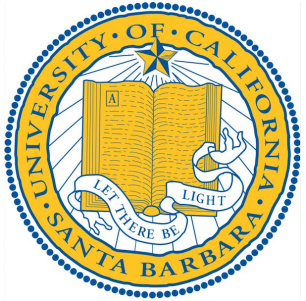
- **Why does the asset pipeline exist in Rails?**



Today's Performance Tricks

Many techniques are used to reduce the number of requests.

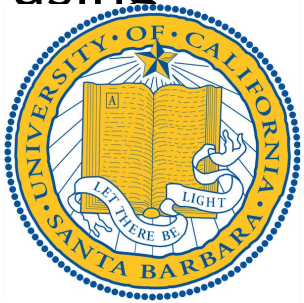
- Why does the asset pipeline exist in Rails?
  - File concatenation.
  - Having many JS, CSS files means having many requests, so we mash them together.



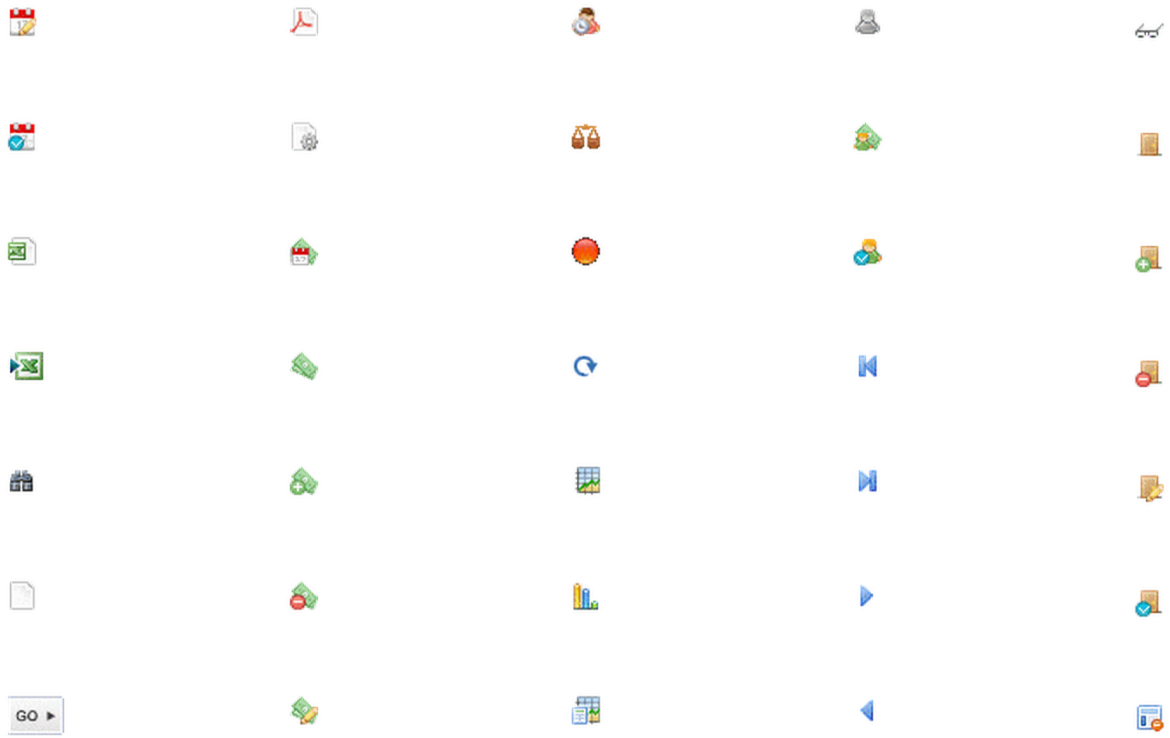
Today's Performance Tricks

Many techniques are used to reduce the number of requests.

- Image Spriting
  - The process of putting lots of smaller images in a single image, and then referring to them all using offsets.



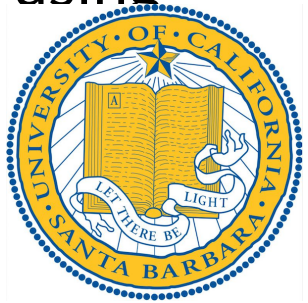
Today's Performance Tricks



Today's Performance Tricks

Many techniques are used to reduce the number of requests.

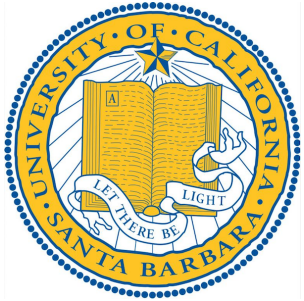
- Image Spriting
  - The process of putting lots of smaller images in a single image, and then referring to them all using offsets.
  - This spriting is cumbersome to deal with, and is really a hack.



Today's Performance Tricks

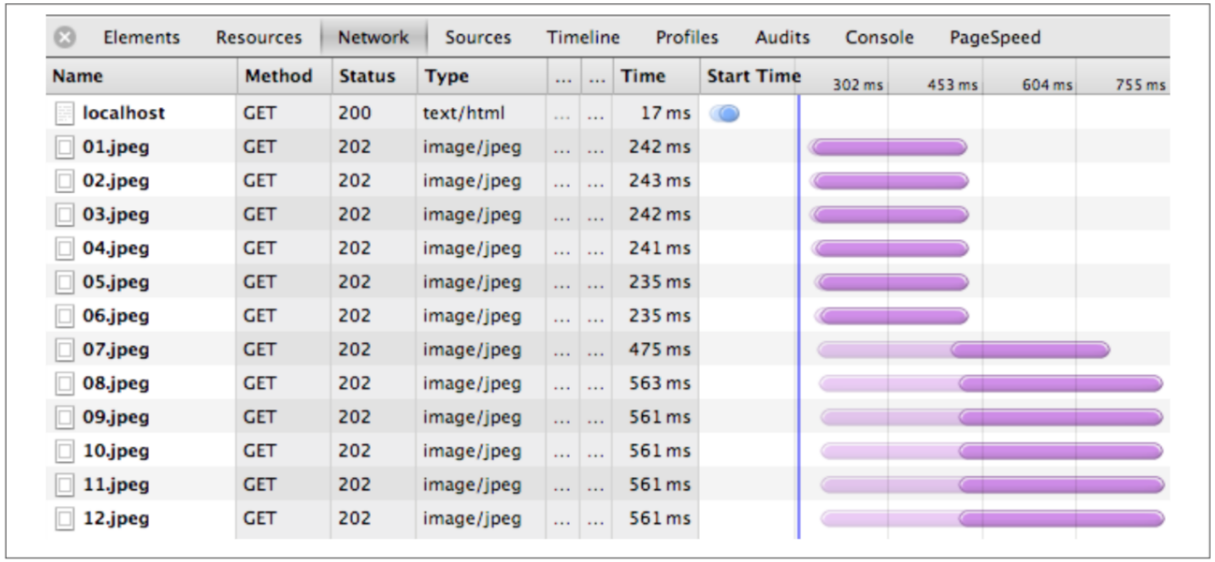
Techniques are used to increase the number of parallel requests that browsers can have to a server.

- Most browsers will only open 6 concurrent TCP connections to a single host
  - Why?



Today's Performance Tricks

This is the result:

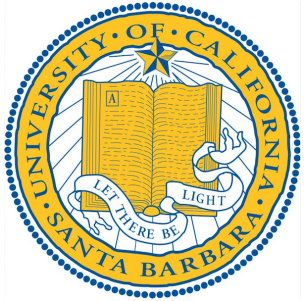


Name	Method	Status	Type	...	...	Time	Start Time
localhost	GET	200	text/html	...	...	17 ms	
01.jpeg	GET	202	image/jpeg	...	...	242 ms	
02.jpeg	GET	202	image/jpeg	...	...	243 ms	
03.jpeg	GET	202	image/jpeg	...	...	242 ms	
04.jpeg	GET	202	image/jpeg	...	...	241 ms	
05.jpeg	GET	202	image/jpeg	...	...	235 ms	
06.jpeg	GET	202	image/jpeg	...	...	235 ms	
07.jpeg	GET	202	image/jpeg	...	...	475 ms	
08.jpeg	GET	202	image/jpeg	...	...	563 ms	
09.jpeg	GET	202	image/jpeg	...	...	561 ms	
10.jpeg	GET	202	image/jpeg	...	...	561 ms	
11.jpeg	GET	202	image/jpeg	...	...	561 ms	
12.jpeg	GET	202	image/jpeg	...	...	561 ms	

Today's Performance Tricks

Techniques are used to increase the number of parallel requests that browsers can have to a server.

- Most browsers will only open 6 concurrent TCP connections to a single host
  - Why?
  - How can we work around this?

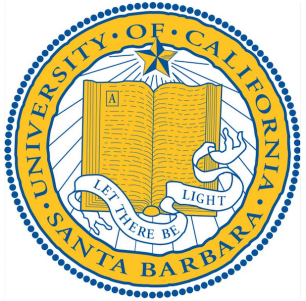


Today's Performance Tricks

Domain sharding!

- Make one host appear as many hosts
- www1.example.com, www2.example.com, www3.example.com
  - All actually point to the same host

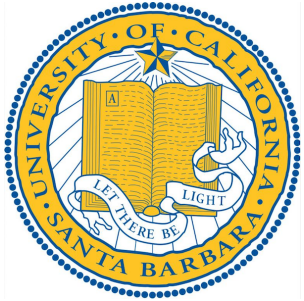
This is cumbersome to deal with and is really a hack.



Today's Performance Tricks

How do we address this?

- We want fewer TCP connections, but…
  - we don't want head-of-line blocking
  - we don't want to have to jam all our css, js artificially together
  - we don't want to have to stuff our images in one big file and deal with offsets everywhere
  - we don't want to have to do DNS tricks to fool the browser

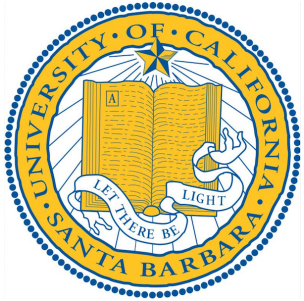


HTTP 2

We address this with HTTP 2

- Started life at Google as SPDY
  - Added to Chrome in 2009
- Pushed towards standardization starting in 2012
- Today supported by Chrome, Firefox, Safari
  - IE is working on it
- Server side support optional in Apache and Nginx

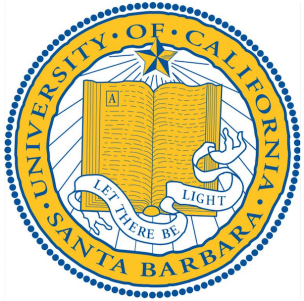
Standard is still in flux.



HTTP 2

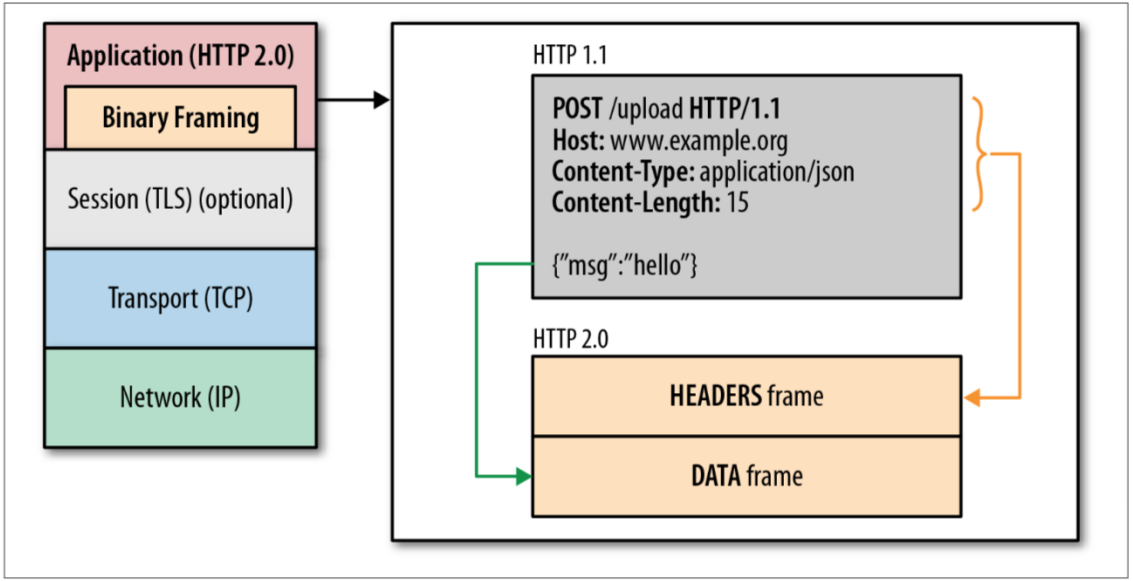
How does HTTP 2 work?

- One TCP connection, multiplex everything over that
- Header compression
- Server push
- Prioritization



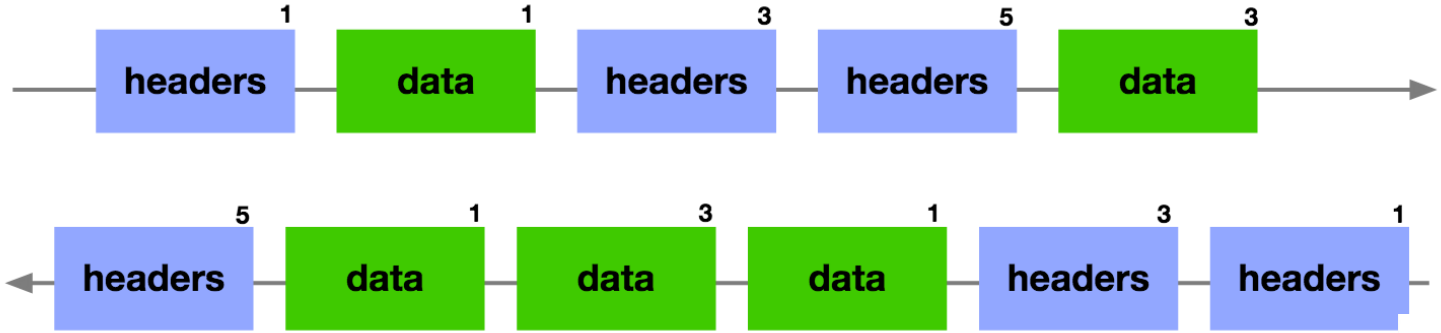
HTTP 2

Single TCP stream: Binary Framed connection



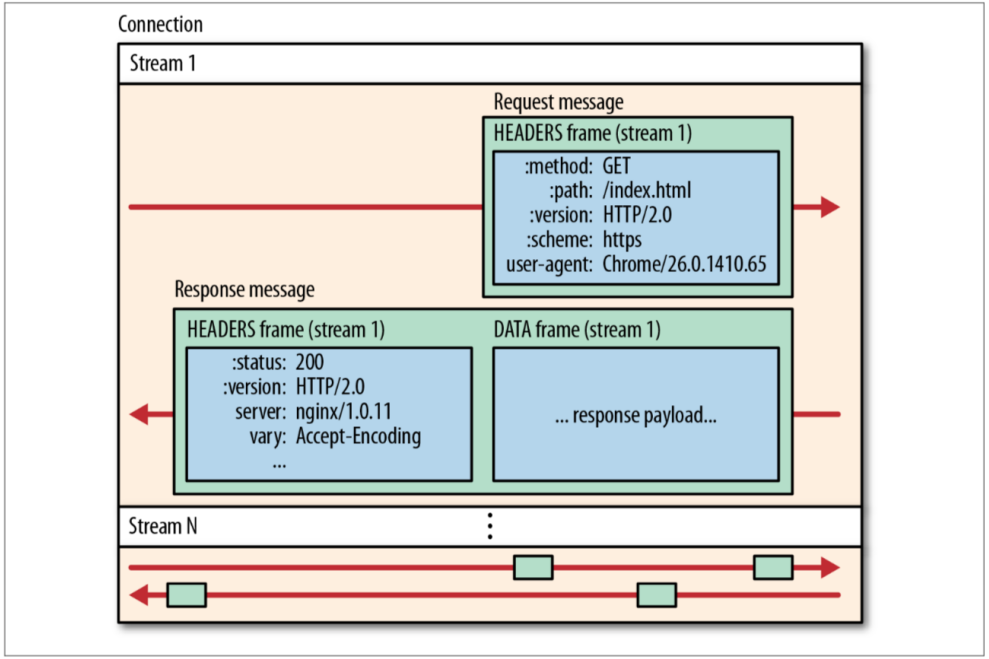
HTTP 2

Single TCP stream: Binary Framed connection



HTTP 2

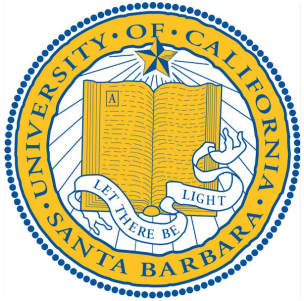
Single TCP stream: Binary Framed connection



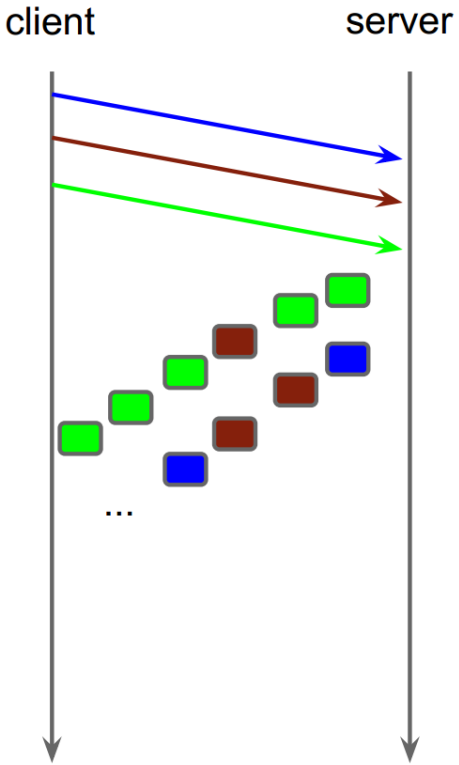
HTTP 2

With binary framing, header compression becomes easy.

- Initially implemented with GZIP, but CRIME attack revealed weaknesses
  - If an attacker can inject data, compressed size can reveal information
- Now uses HPACK
  - More coarse-grained



HTTP 2



Binary framing means ordering of resources is flexible.

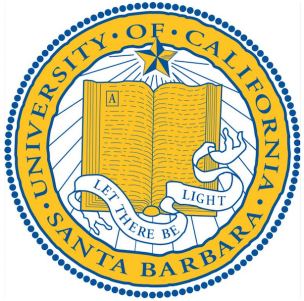
- Handling of many small resources is efficient
- Headers are compressed, so they are lightweight
- Head of line blocking no longer exists
- No TCP setup burden

HTTP 2

Prioritization & Flow Control

- Since order is no longer mandated, we can implement prioritization
  - DOM highest priority, followed by CSS, Javascript
  - Images lowest

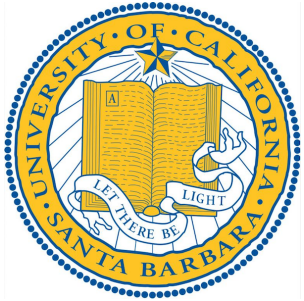
WINDOW_UPDATE flag exists to control number of frames “in flight”



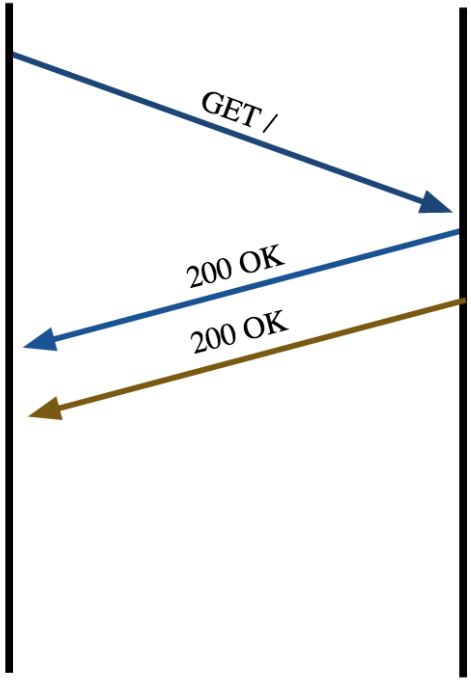
HTTP 2

The hacks of the last 20 years can be thrown out:

- Spriting & asset compliation not needed since we can handle many small resources
- Domain sharding irrelevant since all content is delivered over a single connection



HTTP 2



Server push is now possible

- When a resource is requested, the server can proactively send additional resources using PUSH_PROMISE
- Client can indicate it doesn't want the additional content (e.g. it's cached)

HTTP 2

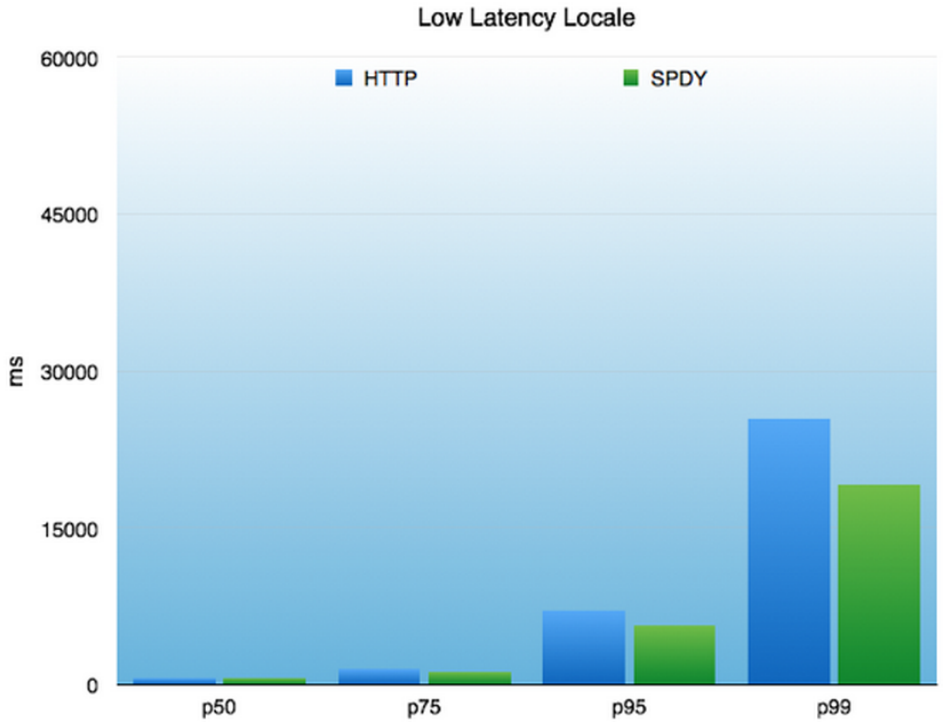
Results: much faster!

	Google News	Google Sites	Google Drive	Google Maps
Median	-43%	-27%	-23%	-24%
5th percentile (fast connections)	-32%	-30%	-15%	-20%
95th percentile (slow connections)	-44%	-33%	-36%	-28%

time from first request byte to onload event in the browser

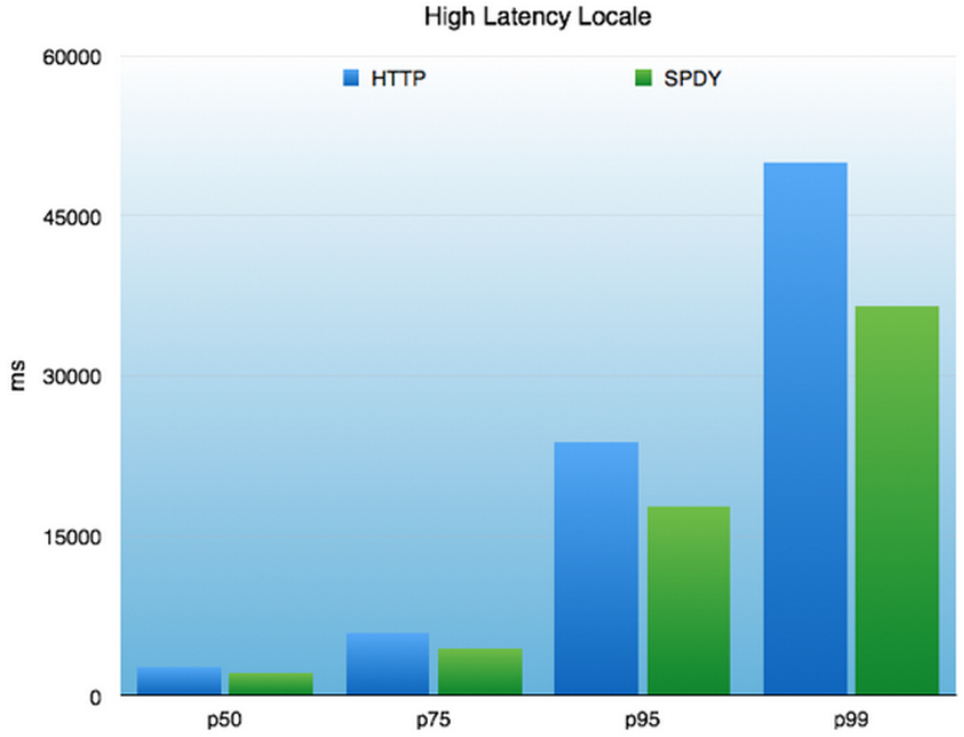
HTTP 2

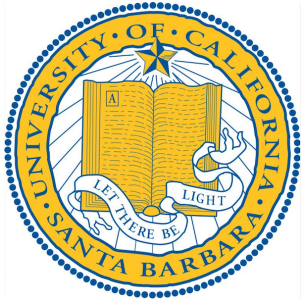
Results:
much faster!



HTTP 2

Results:
much faster!





For next time...

- Load & scaling testing
- Post to Piazza with questions/problems