WEB

PERFORMANCE AND FUTURE

Yan (Peter) Li / Sep 2015

http://10.197.38.188/velocity

View online

Download PDF

HELLO, I'M YAN

- Joined MSTR in July 2008
- Work on Usher Network Manager
- Love Web

TOPIC

- Web Performance Tuning
- Brief Introduction to HTTP/2
- A little on HTTPS

Hope it helps!

DO YOU KNOW WEB?

Frontend + **Backend**

PAGE SIZE (TOP 1000)

Year	Requests	Avg. Transfer Size	Growth
12/17/2010	82	655 KB	N/A
12/15/2011	90	810 KB	24%
12/15/2012	99	1163 KB	44%
12/15/2013	112	1607 KB	38%
12/15/2014	117	1834 KB	14%
08/15/2015	128	2008 KB	9%

http://httparchive.org/trends.php

PAGE SIZE

Site	Requests	Size	Time
www.163.com	434	3.6 MB	1.7 min
www.taobao.com	116	1.3 MB	40.35 s
www.jd.com	79	1.6 MB	1.2 min
www.microsoft.com	76	2.2 MB	16.79 s
www.microstrategy.com	81	1.1 MB	37.29 s

NETWORK SPEED

Desktop		Mobile		
Data rate	**Latency**	**G**	**Data rate**	**Latency**
10 Mbps 100 Mbps 1 Gbps	65-145 ms	2G	100-400 Kbps	300-1000 ms
		3G	0.5-1 Mbps	100-500 ms
		4G	1-50 Mbps	<100 ms

FASTER WEB SITES

What rules/best practices do you know?

YAHOO! BEST PRACTICES

1. Make Fewer HTTP Requests
2. Use a CDN
3. Add Expires or Cache-Control Header
4. Gzip Components
5. Put Stylesheets at Top
6. Put Scripts at Bottom
7. ...

https://developer.yahoo.com/performance/rules.html

GOOGLE PAGESPEED INSIGHTS

https://developers.google.com/speed/pagespeed/insights/

PAGE LOAD TIMELINE

redirections	DNS	TCP	request response	rendering
		image spriting	caching	css on top
minimize redirection	reduce lookups			
		minify	domain sharding	js at bottom

Similar graph: http://www.w3.org/TR/navigation-timing/#processing-model

REDIRECTIONS

| redirections | DNS | TCP | request response | rendering |

minimize redirection

MINIMIZE REDIRECTIONS

- Links/forms
- HTTP 3XX status codes
- HTML meta refresh

```html
<meta http-equiv="refresh" content="5; url=http://example.com/">
```

- JavaScript

```javascript
window.location.reload(true); // document.location?
window.location.replace("http://www.z.cn");
window.location.assign("http://www.z.cn");
window.location.href = "http://www.z.cn";
window.location = "http://www.z.cn";
window.history.forward();
window.history.back();
window.history.go(-1);
```

DNS

redirections	DNS	TCP	request response	rendering

reduce
lookups
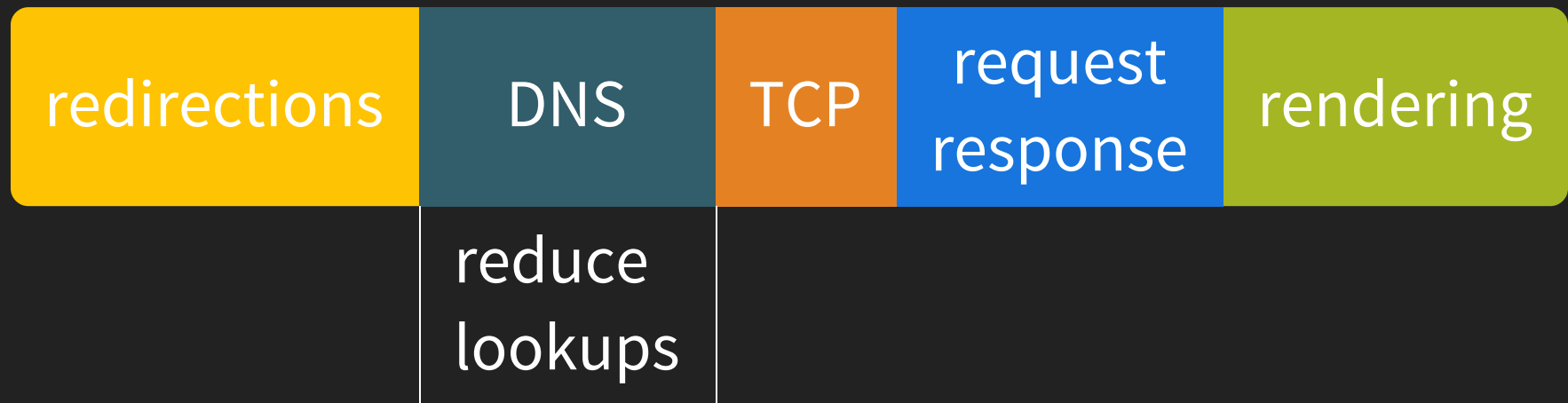
DNS LOOKUP

www.google.com

1. *Browser/OS* cache
2. *Intranet* cache
3. *ISP/local* cache
4. *Root* name server
5. *.com* name server
6. *google.com* name server

HOW FAST IS YOUR LOCAL DNS?

chrome://histograms/DNS.ResolveSuccess

- Good: <30ms
- Average: 30-100ms
- Ouch: 100ms+

HTTPDNS

- 1st request: Http gets nearest server IP
- Subsequent: nearest server IP

全局精确流量调度新思路-HttpDNS服务详解
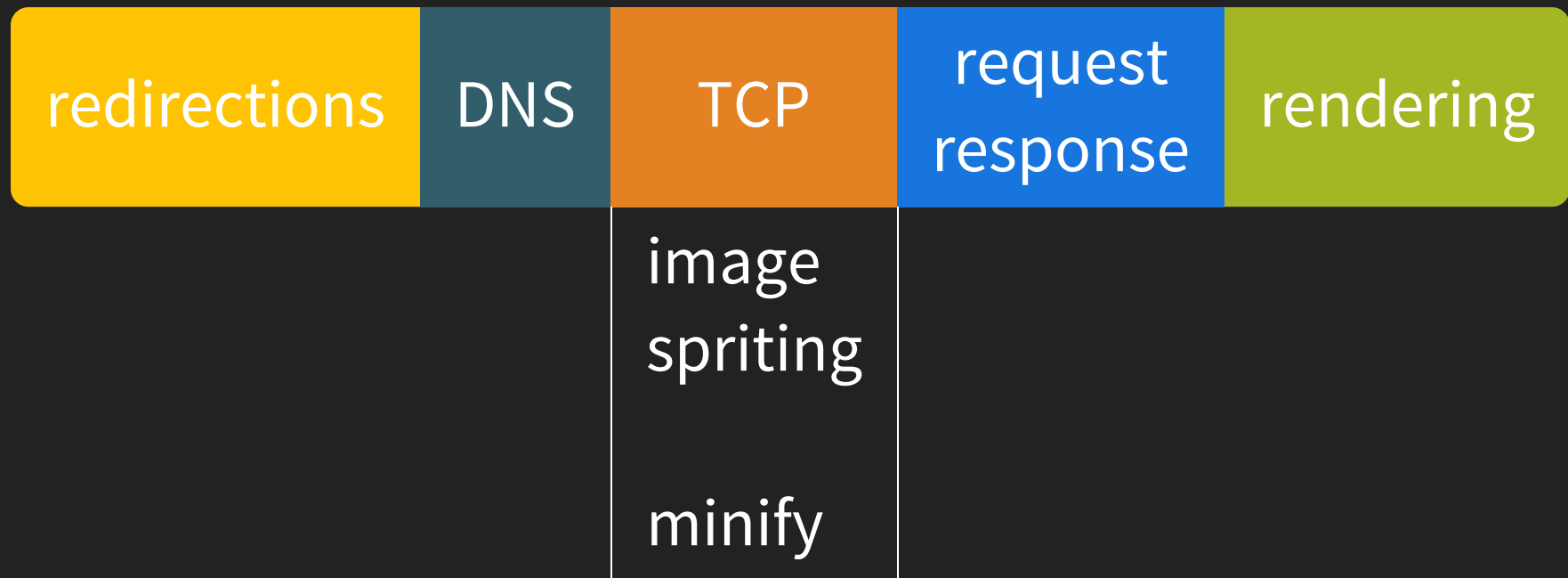App域名劫持之DNS高可用 - 开源版HttpDNS方案详解

DNS PREFETCH

```
<link rel="dns-prefetch" href="www.microstrategy.com">
```
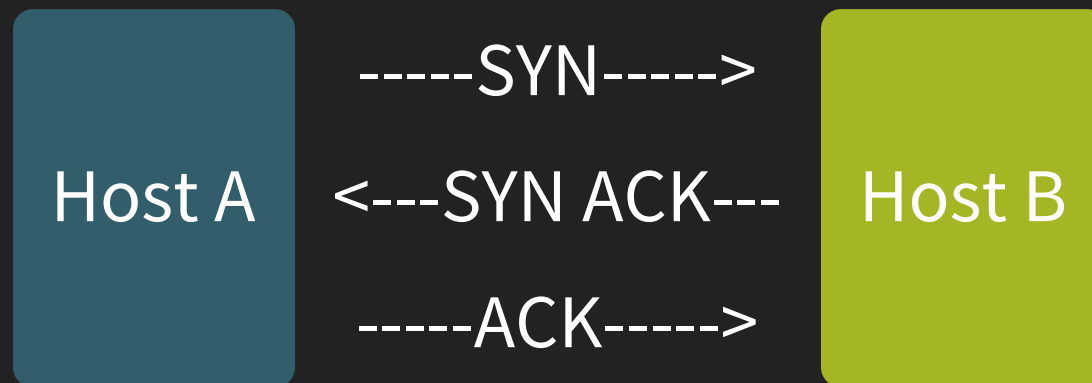
Hint browser to pre-resolve these names
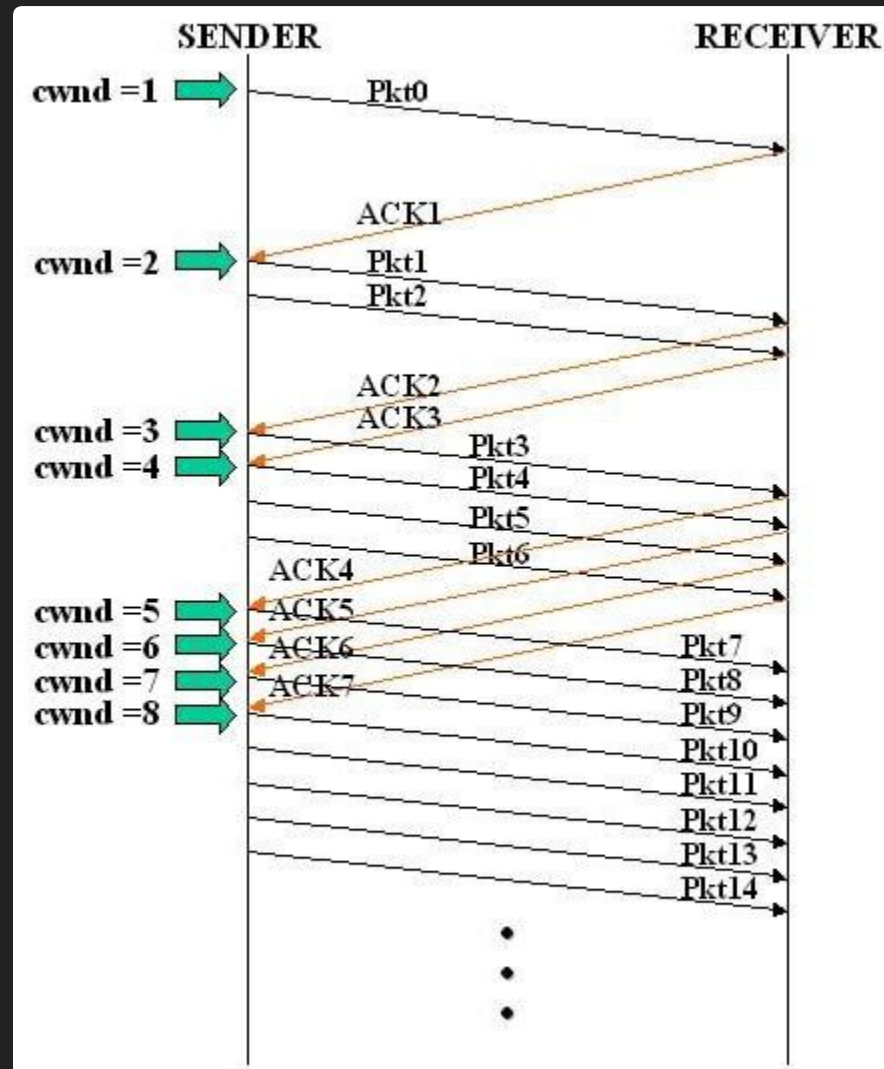
Supported: Chrome, Firefox, Safari, IE9+

chrome://dns/

TCP

| redirections | DNS | TCP | request response | rendering |

image spriting

minify

TCP 3-WAY HANDSHAKE

Host A   -----SYN----->   Host B

Host A   <---SYN ACK---   Host B

-----ACK----->
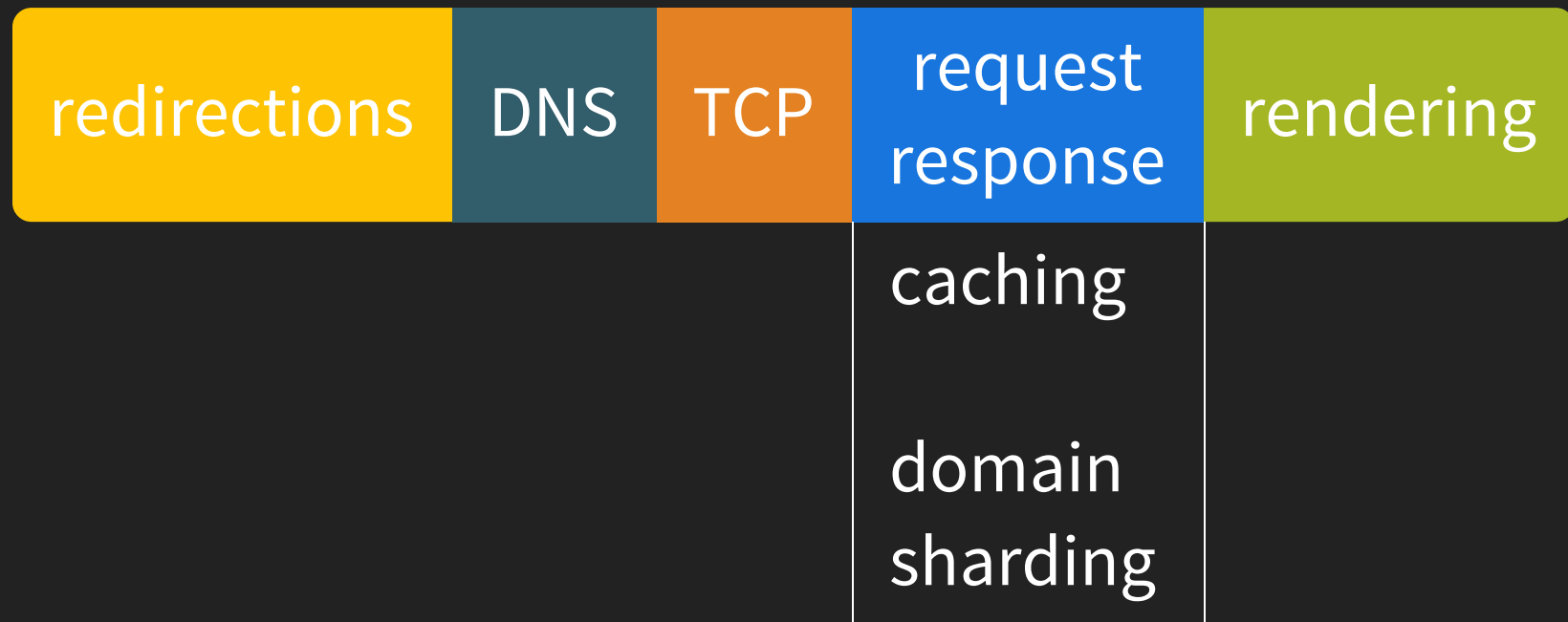
See TCP 3-Way Handshake (SYN,SYN-ACK,ACK)

TCP SLOW START

TCP SLOW START

Wikipedia: Slow Start

Tuning initcwnd for optimum performance

Linux TCP/IP tuning for scalability (tcp_slow_start_after_idle)

REQUEST & RESPONSE

redirections | DNS | TCP | request response | rendering

caching

domain sharding

CACHING

Caching Tutorial by Mark Nottingham

Check your pages with REDbot

DOMAIN SHARDING

Browser

2(1999) simultaneous connections

6-8(2010) simultaneous connections

Domain

http://www.mobify.com/blog/domain-sharding-bad-news-mobile-performance/

RESOURCE PREFETCH

```
<link rel="prefetch" href="next-page.css">
<link rel="subresource" href="critical.js">
```

- prefetch
  - Low-priority download of resources to be used on subsequent pages
  - Chrome, Firefox, Safari, IE9+
- subresource
  - Early loading of resources within the current page
  - Chrome only

RENDERING

redirections	DNS	TCP	request response	rendering

css on top

js at bottom

INLINE CRITICAL RESOURCES

Eliminate render-blocking JavaScript and CSS in above-the-fold content

Your page has 2 blocking script resources and 2 blocking CSS resources. This causes a delay in rendering your page.

None of the above-the-fold content on your page could be rendered without waiting for the following resources to load. Try to defer or asynchronously load blocking resources, or inline the critical portions of those resources directly in the HTML.

Remove render-blocking JavaScript:

https://use.typekit.net/uza8xtu.js

https://███████████████████.js

Optimize CSS Delivery of the following:

https://████████████████████████.min.css

https://use.typekit.net/...d4c9c19d9c76d37e9ffaa92b500b8c940d090388

http://www.smashingmagazine.com/2015/08/understanding-critical-css/

Alva Cheung

"Load non-critical features asynchronously"

Facebook - The technology behind preview photos

PRERENDER

```
<link rel="prerender" href="http://www.microstrategy.com">
```

Initiate background pre-render of entire page

Supported: Chrome, Firefox

chrome://net-internals/#prerender

chrome://predictors/

JAVASCRIPT RENDERING

Why care?

Static Web Pages	Interactive
CSS	CSS
HTML	JavaScript
JavaScript	HTML

AngularJS, React

INLINE CACHING

Speed up runtime method binding by remembering the results of a previous method lookup directly at the call site

https://en.wikipedia.org/wiki/Inline_caching

INLINE CACHING CAN BE EASILY BROKEN

```javascript
function addTwoThings(thing1, thing2) {
    return thing1 + thing2;
}

addTwoThings(1, 2);
addTwoThings(100, 200); /* fast */
addTwoThings('a', 'b'); /* slow */
```

```javascript
function Dog(name) {
    this.name = name;
}

var dog1 = new Dog('Jim'); /* hidden class C1 */
var dog2 = new Dog('Bin'); /* hidden class C2 */
dog2.gender = 'male';
```

CSS RENDERING

- Reflow
  - relocate elements
  - change content
  - window resize
- Repaint
  - change color
  - change background color
  - change visibility

EVENT DEBOUNCING/THROTTING

onresize, onscroll...etc

USE THE DOM CHANGE QUEUE

```
var $div = $('#content')
var elements = ['a', 'b', 'c', 'd'];
for(var i=0,l=elements.length; i<l; i++) {
    $div.append(elements[i]);
}
```

```
var $div = $('#content')
var elements = ['a', 'b', 'c', 'd'];
for(var i=0,l=elements.length; i<l; i++) {
    // one line of evilness
    var a = $(window).scrollTop;
    $div.append(elements[i]);
}
```

PAINTING PERFORMANCE

- Minimize DOM depth
- Minimize z-index depth
- Minimize CSS rules
- Use faster selectors
  - Avoid descendant selector
  - Structure selector right to left

```
.class ul li a { color: blue; }
```
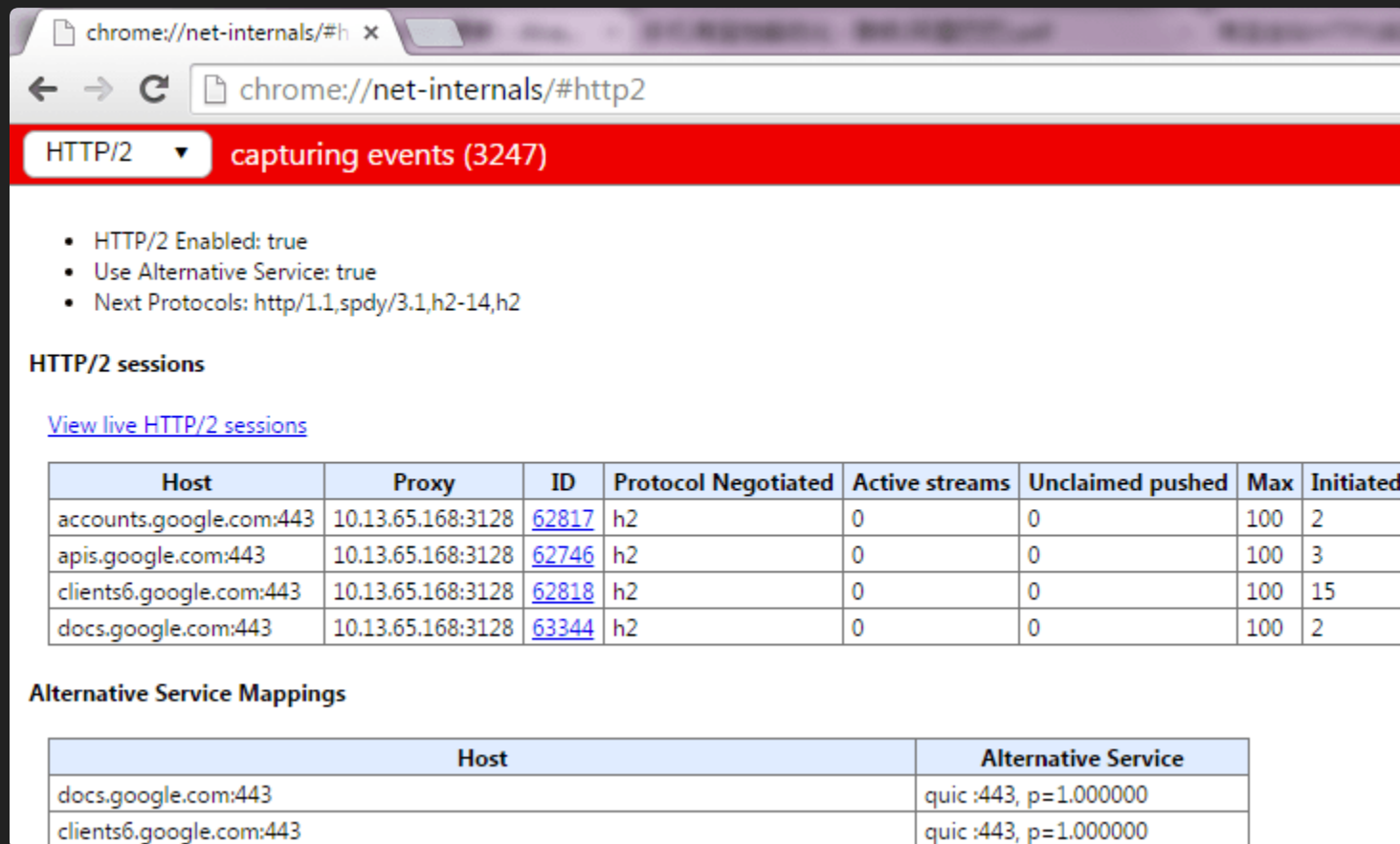
- Use expensive properties sparingly

```
border-radius; box-shadow; transform,filter; :nth-child; position:fixed;
```

- Don't use universal selectors

```
*; [disabled]; [type="text"]
```

Improving CSS Performance

PREPARE FOR HTTP/2

IETF RFC 7540

HTTP/2 AIMS

1. Reduce latency
2. Reduce total number of TCP connections
   i.e., reduce number of open sockets
3. Better web security
4. Maintain compatibility with HTTP/1.1
   clients and server
5. Maintain same usability as HTTP/1.1
   i.e., can be used wherever we use HTTP/1.1
6. Better web security

HTTP/2 FEATURES

1. ## Multiplexing
   Multiple asynchronous HTTP requests over a single TCP connection.

2. ## Server Push
   Multiple responses for single request

3. ## Header Compression
   Compress HTTP headers along with content.

4. ## Request prioritization
   While making multiple HTTP requests to a same domain they can be prioritized.

5. ## Binary Protocol
   HTTP/2 is binary protocol whereas HTTP/1.1 is text protocol.

HTTP/2 Complete Tutorial

HTTP/2 COMPATIBILITY

Request	Response	
HTTP/2 Client	**HTTP/1.1 Server**	**HTTP/2 Server**

`GET / HTTP/1.1`
`Host: server.example.com`
`Connection: Upgrade, HTTP2-Settings`
`Upgrade: h2c`
`HTTP2-Settings: <payload>`
	`HTTP/1.1 200 OK`
	`Content-Length: 243`
	`Content-Type: text/html`
	
	`...`
		`HTTP/1.1 101 Switching Protocols`
		`Connection: Upgrade`
		`Upgrade: h2c`
		
		`[ HTTP/2 connection ...`

See IETF RFC 7540

MULPLEXING IN HTTP/2

- HTTP/1.1 Pipelining
  - Multiple HTTP requests are sent on a single TCP connection asynchronously
  - Server responses synchronously
  - First-in-first-out
  - Head-of-line (HOL) blocking, see Wiki
- HTTP/2 Mulplexing
  - Streams and frames
    Every HTTP/2 request and response is given a unique id called as stream id

  - Server also responses asynchronously
  - The request and response both happen parallelly

SERVER PUSH IN HTTP/2

- Server sends multiple responses for a single request
- A client can request that server push be disabled
- A client cannot push

IETF RFC 7540, Section 8.2 Server Push

HEADER COMPRESSION IN HTTP/2

- A technique of not sending the same headers again
- Client and server maintain a headers table containing the last response and request headers
- For the first request or response they send all the required header
- For subsequent requests client and server omit headers which are same as the previous request or response

IETF RFC 7541, HPACK: Header Compression for HTTP/2

HTTPS

HTTP + SSL/TLS = HTTPS

- **Authentication**
  Am I talking to who they claim to be?

- **Data integrity**
  Has anyone tampered with the data?

- **Encryption**
  Can anyone see my conversation?

Equally Important!

ENCYPTION

- Handshake: asymmetric crypto
- Application data transfer: symmetric crypto

Transport Layer Security

IS HTTPS SLOW?

- Extra CPU costs
- Extra roundtrips

IsTlsFastYet.com:

> *"TLS has exactly one performance problem: it is not used widely enough. Everything else can be optimized."*

Dive deeper: YouTube - Is TLS Fast Yet?

HTTPS

- Get a 2048-bit TLS certificate
- Eliminate both *Yello Triangle* and *Shield*
- Use latest version of Kernal, OpenSSl and Apache/Nginx
- Use SPDY3.1 & HTTP/2
- Use protocol relative URIs

```
<script src="//code.jquery.com/jquery-1.11.3.min.js"></script>
```

- HTTP URL --> 301 --> HTTPS URL
- Apply HSTS (see next slide)

Mozilla Wiki

Qualys SSL Labs

HTTP STRICT TRANSPORT SECURITY (HSTS)

```
Strict-Transport-Security: max-age=10886400; includeSubDomains
```

Browser remembers (for specified max-age period) that it should automatically request HTTPS resources for this site and its subdomains.

HSTS eliminates HTTP --> HTTPS redirects.

WEB DEV RESOURCES

- Web Development Reading List

REFERENCES

1. The Quest to Delight Our Users by Alva Cheung(Google)
2. eBay对页面性能的监控和调优 by 施尉霁(eBay)
3. 淘宝全站HTTPS实践 by 李振宇(阿里巴巴)
4. 手机淘宝性能优化 by 黎明(阿里巴巴)
5. Google I/O 2014 - HTTPS Everywhere
6. Preconnect, prefetch, prerender by Ilya Grigorik
7. Resource Hints
8. https://http2.github.io/

THANKS!

WeChat ID: peterleepersonal