




Front-End Performance

October 11, 2013

- Fred Hatfull (fhats)
-  **Yelp** Engineering
 - Infrastructure
 - Operations
- Case Western Alum
 - Computer Science ('11)
- Pittsburgh, PA Native



- Infrastructure & Operations
 - Keep the site up
 - Keep the site fast
 - Keep other developers productive
- I <3 programming & systems administration



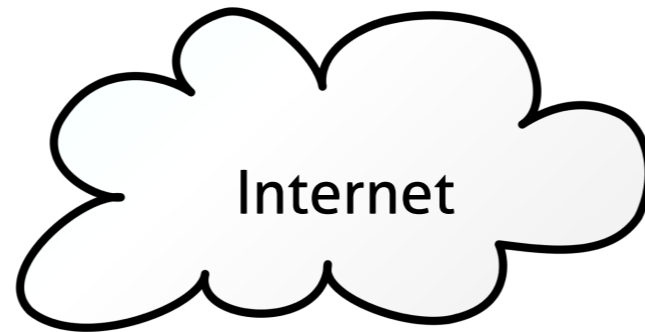
! Important !



Stop me and ask questions if you have them.

What is the Front-End?

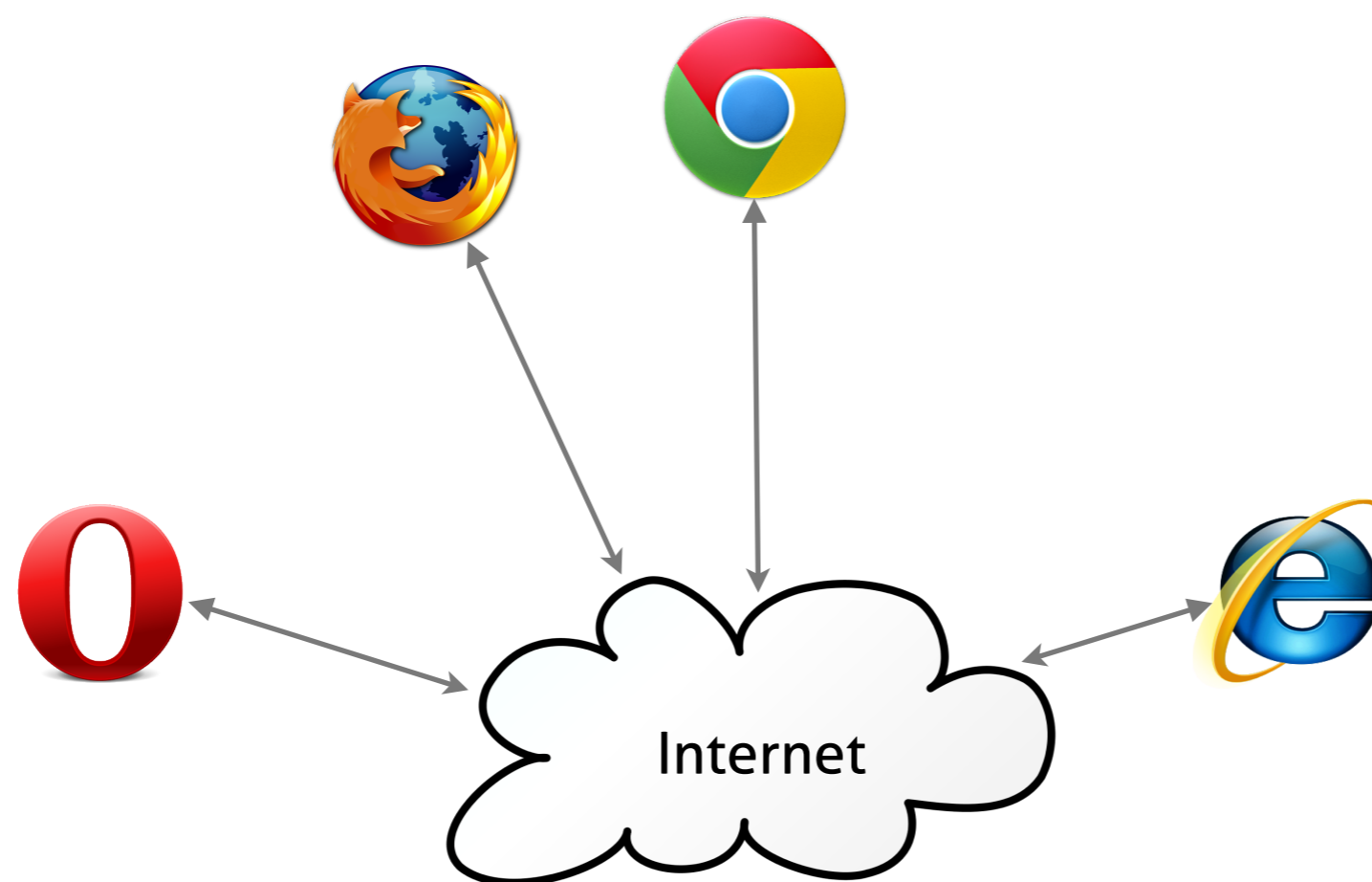
What Is The Front-End?



Friday, October 11, 13

The internet we all know and love

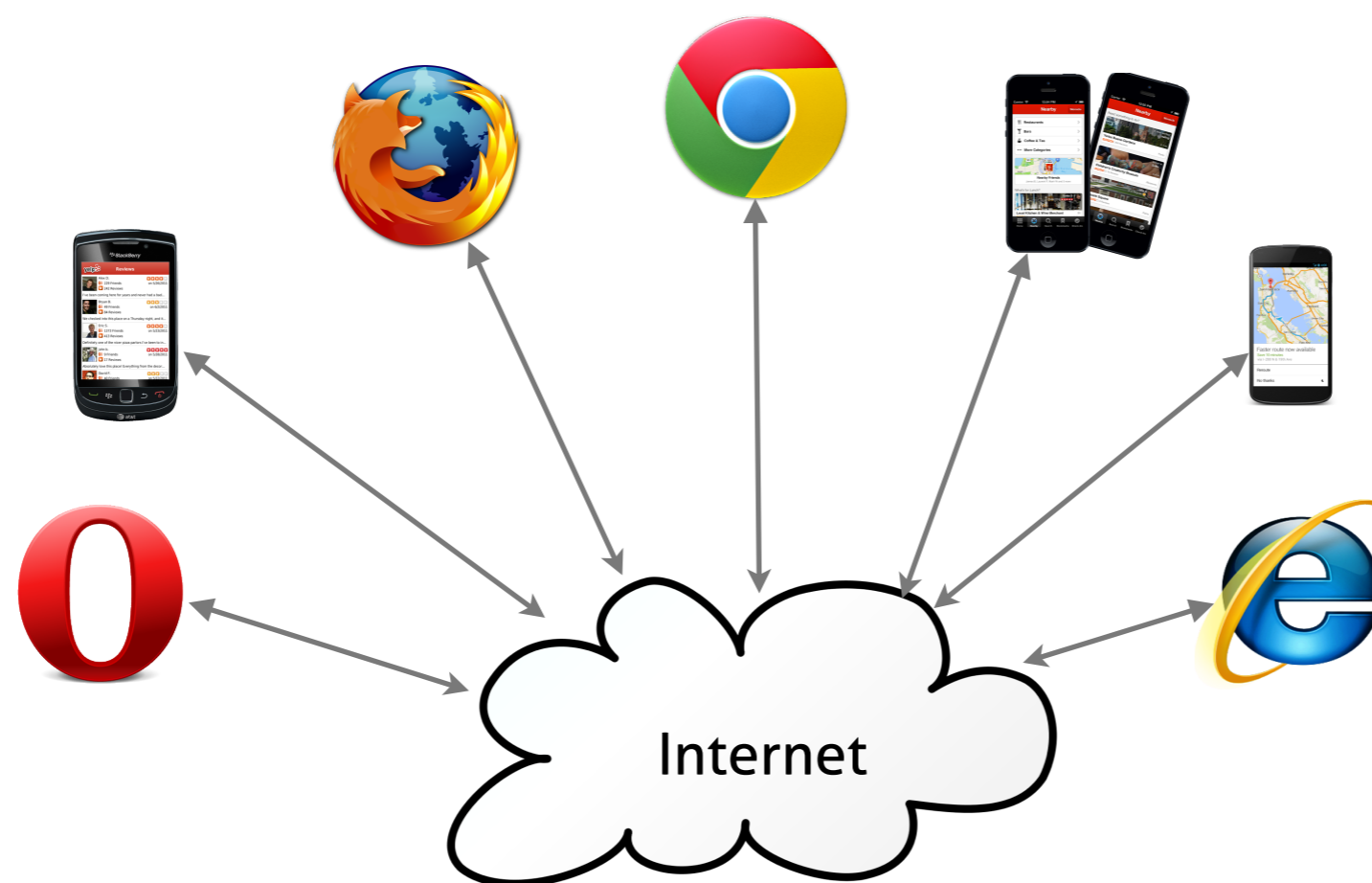
What Is The Front-End?



Friday, October 11, 13

Some users talk to the internet via a 'desktop' web browser

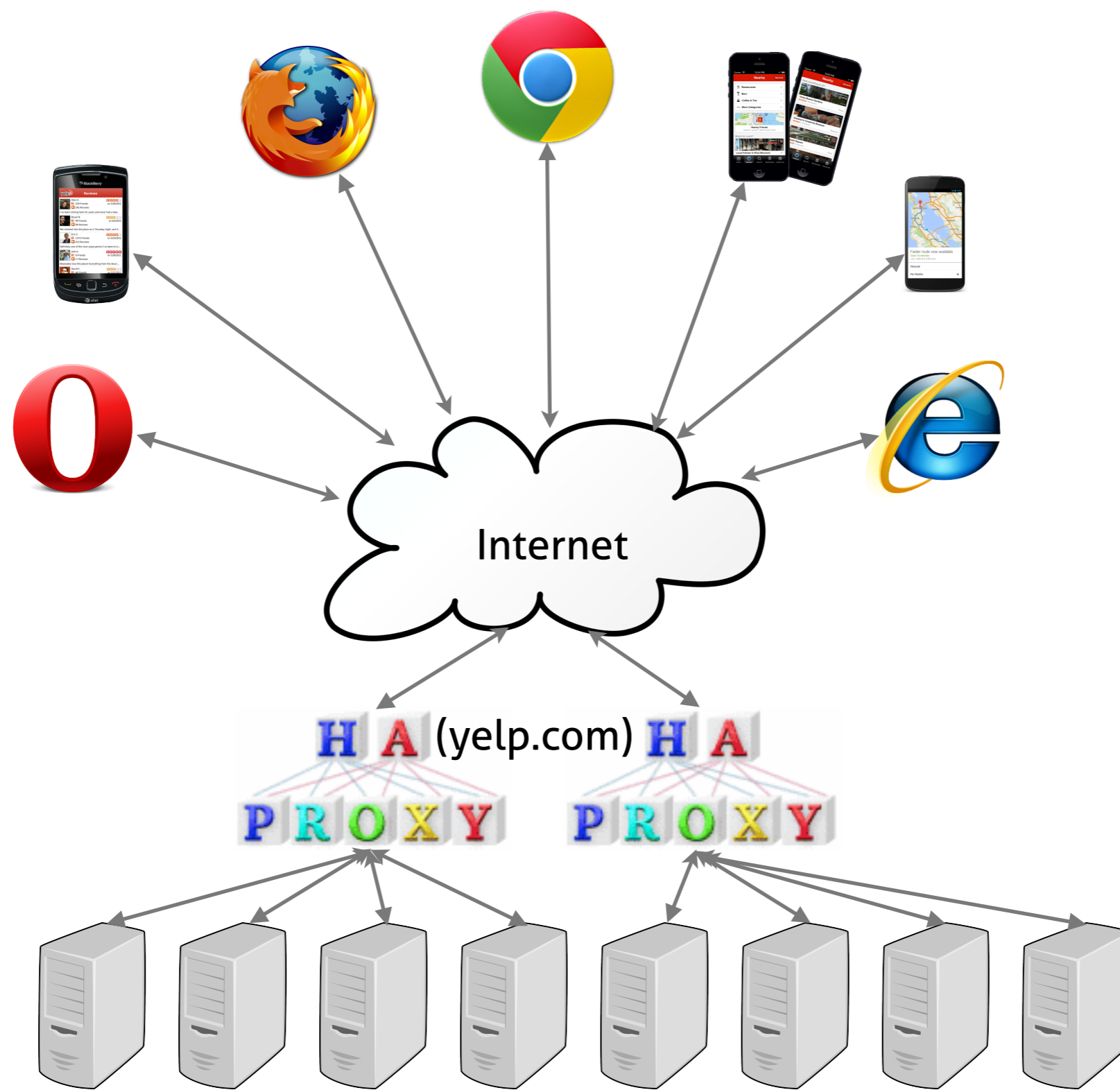
What Is The Front-End?



Friday, October 11, 13

...and some use mobile devices

What Is The Front-End?

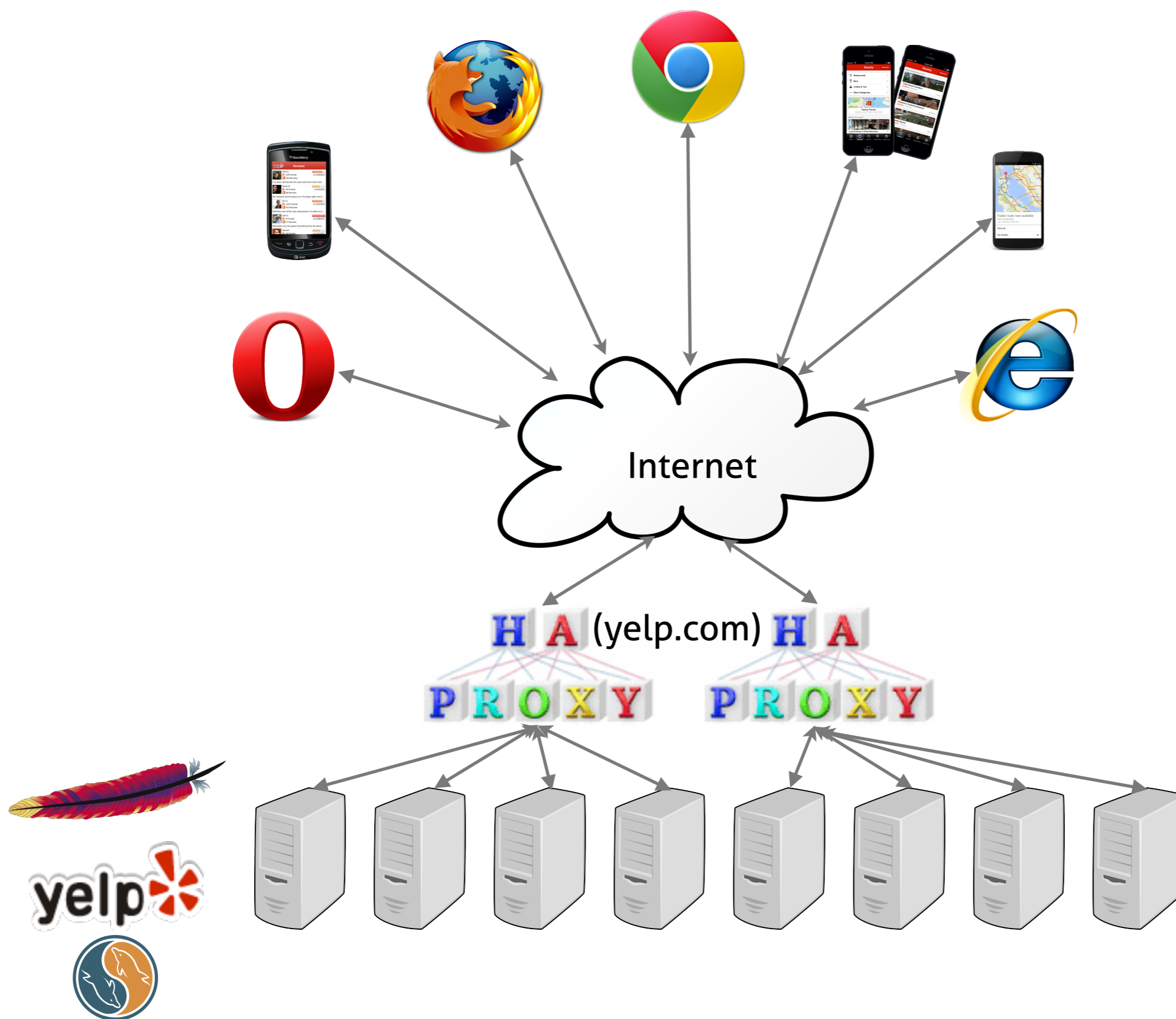


Friday, October 11, 13

and those clients all talk to our web app & infrastructure

(depicted as terminating at a load-balancer, which distributed requests to application servers)

What Is The Front-End?

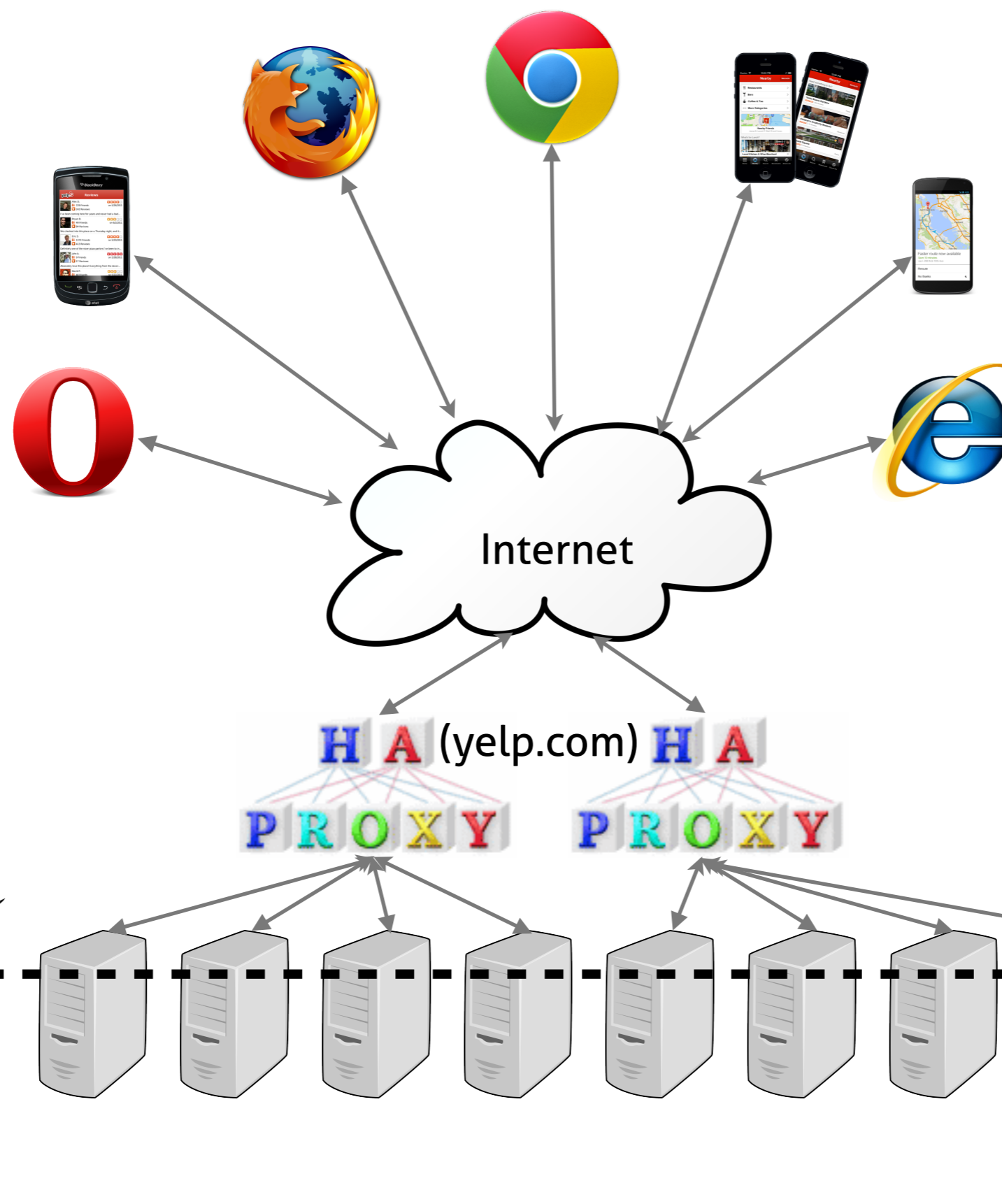


Friday, October 11, 13

typically those apps leverage some software that supports the application, like

- * HTTP servers
- * application containers (WSGI interfaces, etc)
- * databases (mySQL, mongo, etc)

What Is The Front-End?



Friday, October 11, 13

In this broader picture, we typically separate the sides of the site into front-end and back-end

What Is The Front-End?

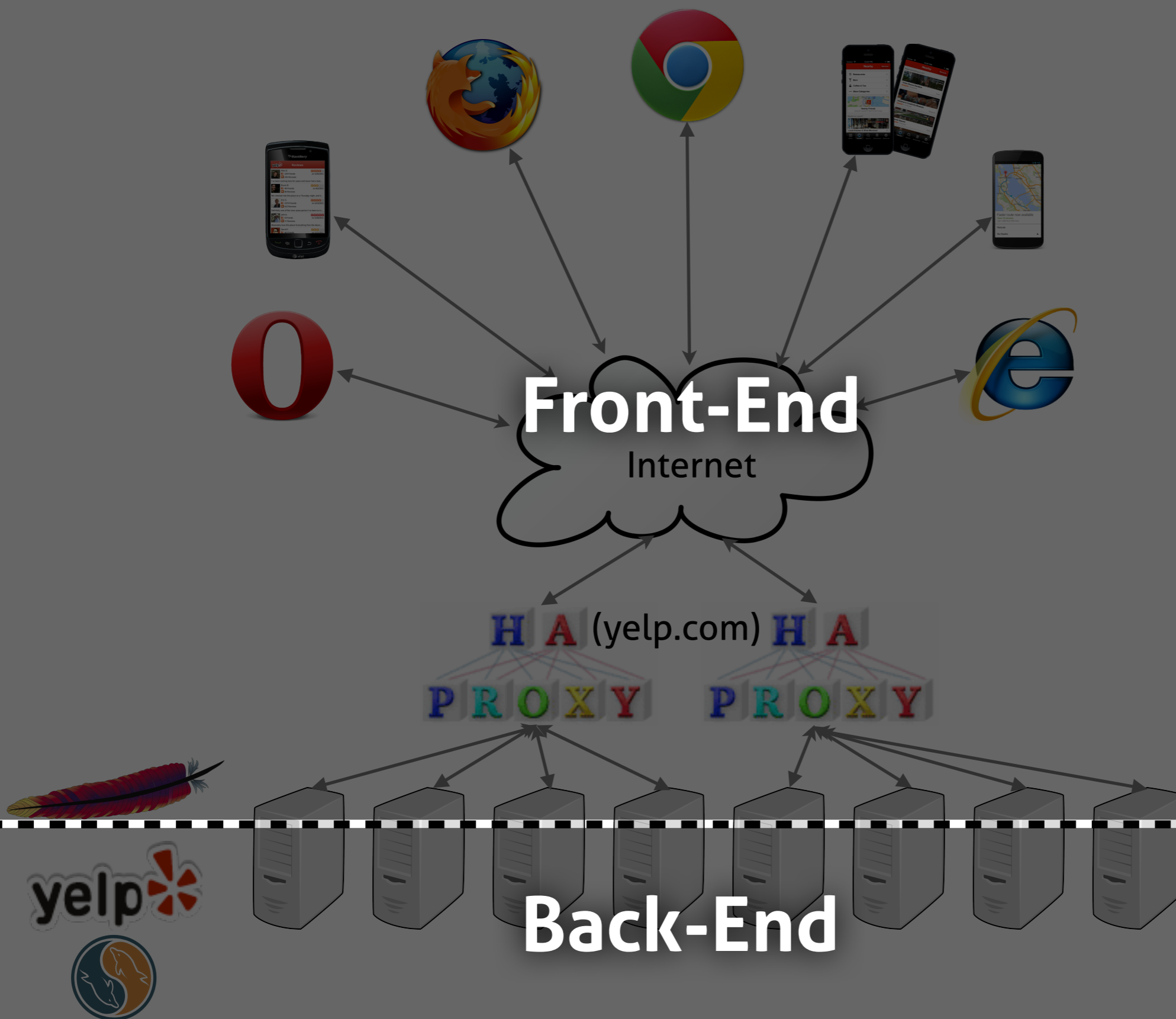


Friday, October 11, 13

The back-end of an app or site almost always deals with:

- * application development
- * supporting infrastructure
- * development tooling
- * operations / system administration

What Is The Front-End?



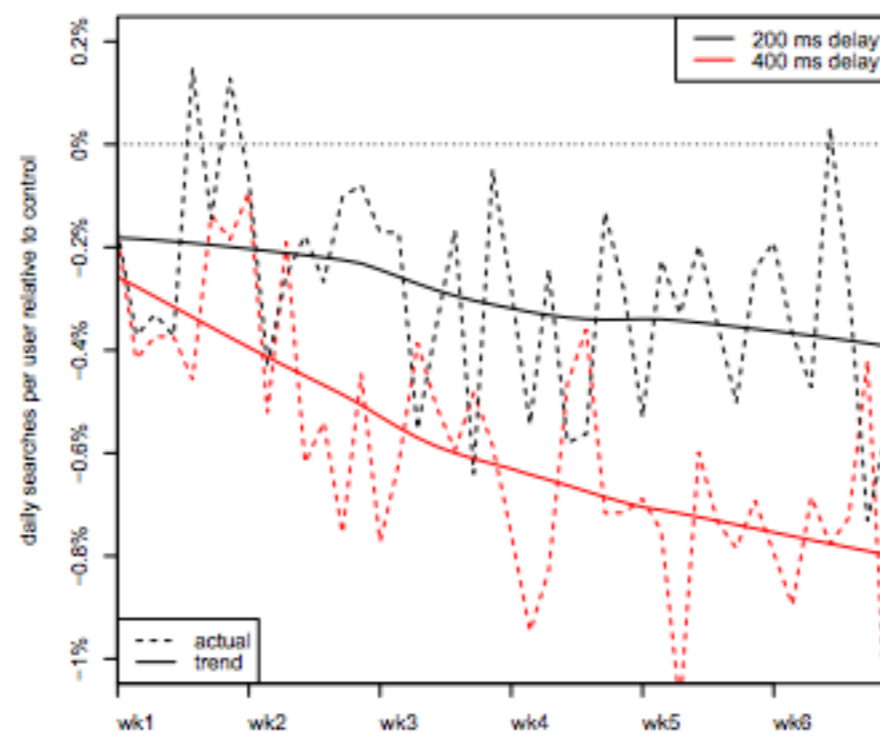
Friday, October 11, 13

The front-end usually refers to everything between the application and the client, including:

- * CSS, HTML, design and presentation of pages
- * Javascript / client-side application development
- * Content delivery/CDNs
- * Static asset serving
- * Sometimes HTTP server tuning

- Faster, better-performing sites retain users
- More users = more \$\$\$
- SEO results can be impacted [1]
- Faster sites reduce operating costs

Figure 2: Impact of Post-header Delays Over Time



Friday, October 11, 13

[1] <http://googlewebmastercentral.blogspot.com/2010/04/using-site-speed-in-web-search-ranking.html>

Figure 2 from http://services.google.com/fh/files/blogs/google_delayexp.pdf

- Other interesting numbers [2]
 - Bing: 2-second slowdown -> **-4.3%** revenue/user
 - Yahoo: 400ms slowdown -> **5-9% drop** in full-page traffic
 - Mozilla: 2.2 second speedup -> **15.4%** increase in downloads

Does the Front-End Matter?



- Back-end performance often majority of engineering focus
- Front-end performance often majority of time consumed

Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency	Timeline	1.00 s	1.50 s	2.00 s	2.50 s
www.yelp.com	GET	200 OK	text/html	Other	18.5 KB 80.9 KB	243 ms 217 ms					

Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency	Timeline	200 ms	300 ms	400 ms	500 ms	600 ms	700 ms	800 ms	900 ms
www.google.com	GET	200 OK	text/html	Other	0 B 113 KB	92 ms 85 ms									

Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency	Timeline	4.00 s	6.00 s	8.00 s
www.facebook.com	GET	200 OK	text/html	Other	0 B 634 KB	2.15 s 194 ms				

Friday, October 11, 13

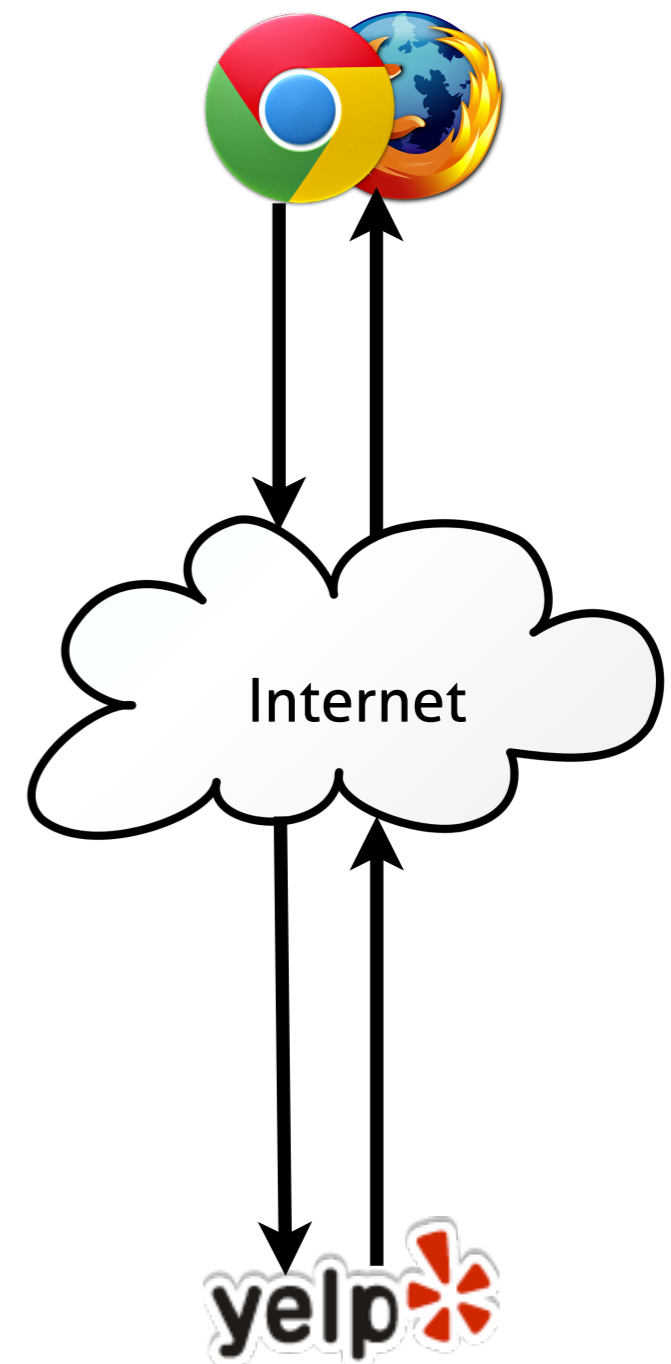
blue line = DOMContentLoaded – Document is loaded and parsed, but stylesheets, images, and other content is not loaded

red line = `load` event – page is loaded and ready

Principles

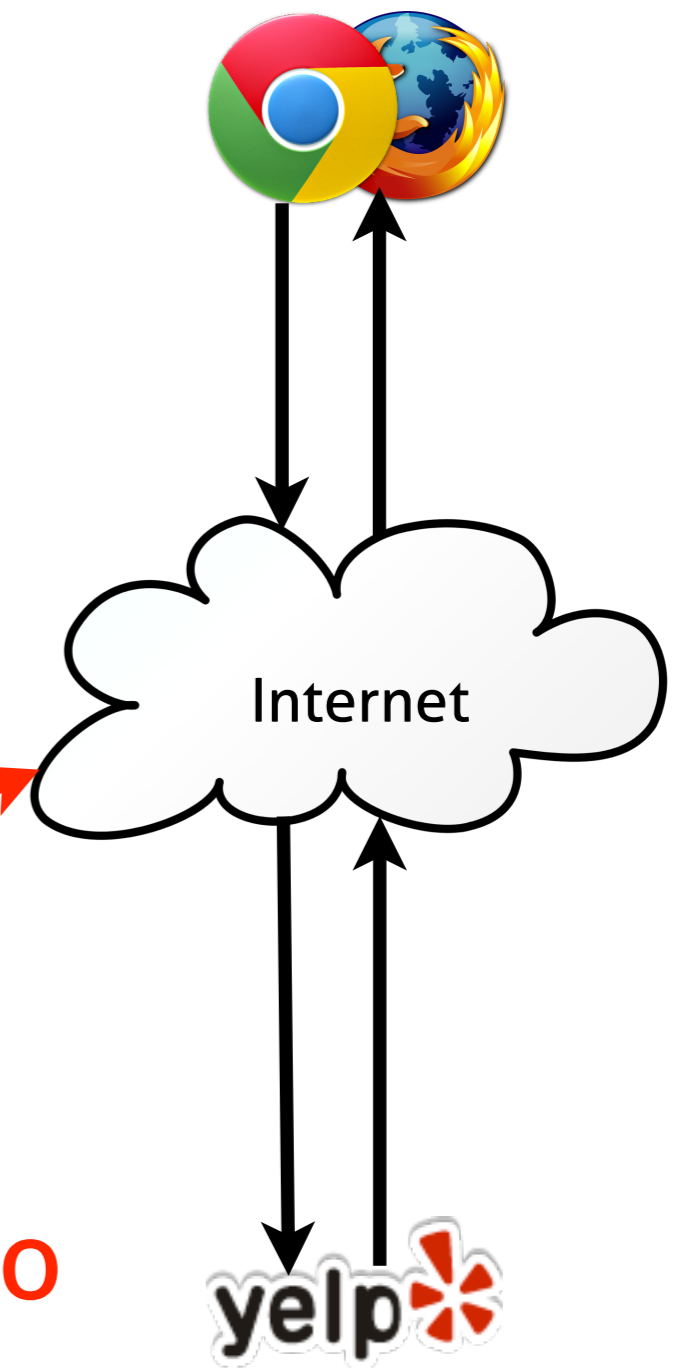
- Reduce content size
- Avoid the network
- Make the network faster
- Play nicely with clients
- Stay responsive

- Reduce content size
- Avoid the network
- Make the network faster
- Play nicely with clients
- Stay responsive



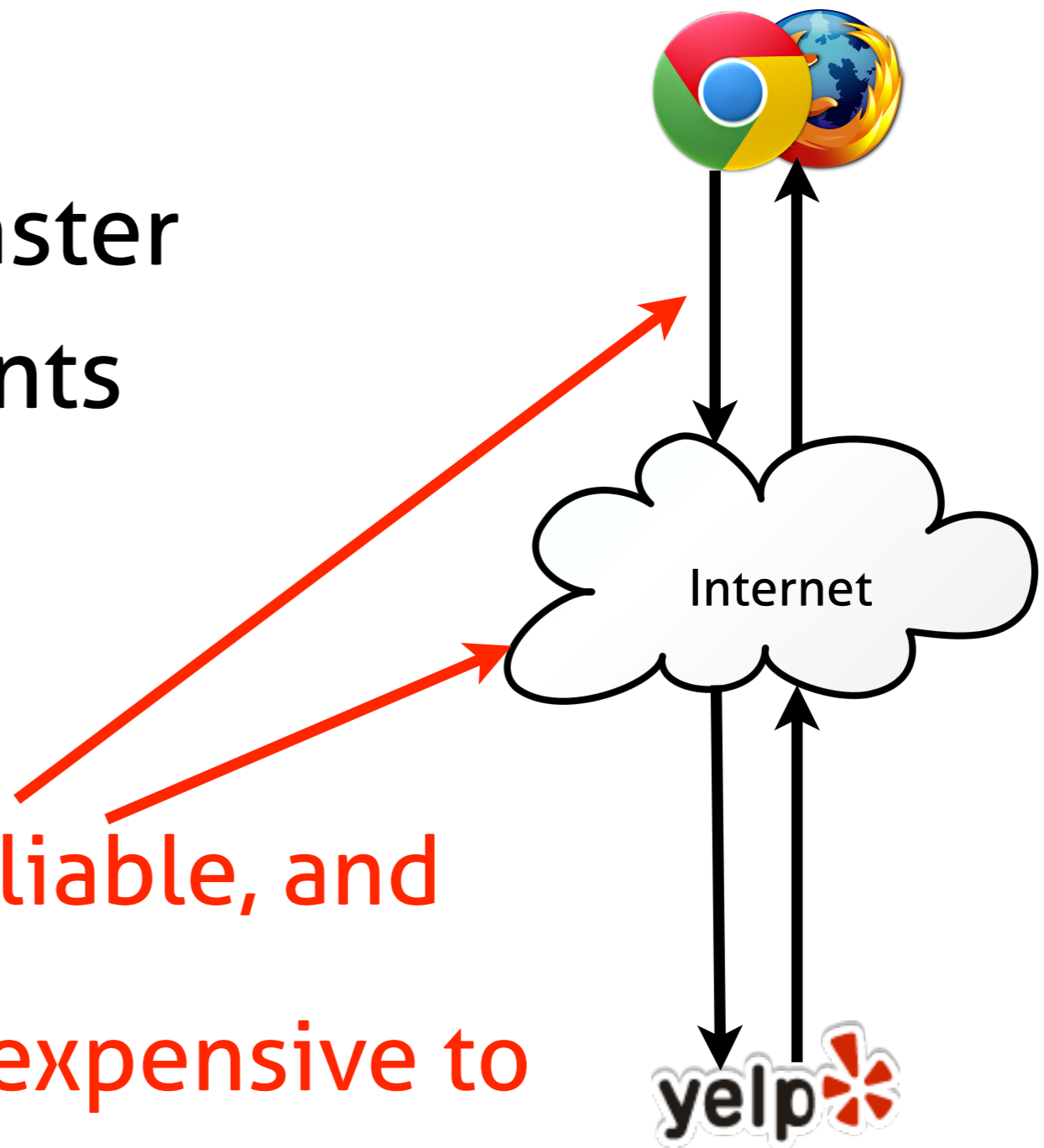
- Reduce content size
- Avoid the network
- Make the network faster
- Play nicely with clients
- Stay responsive

Slow, unreliable, and relatively expensive to traverse

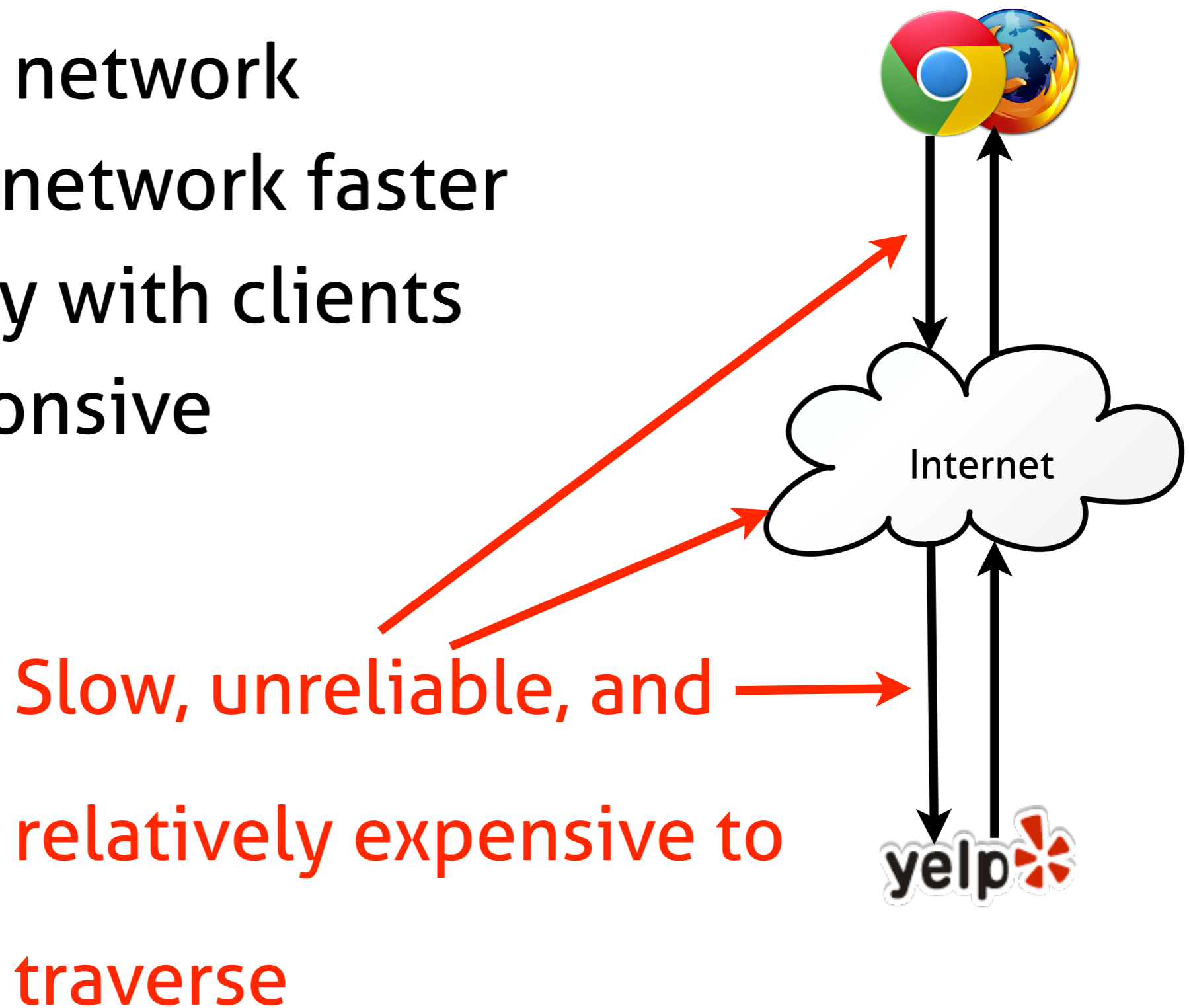


- Reduce content size
- Avoid the network
- Make the network faster
- Play nicely with clients
- Stay responsive

Slow, unreliable, and
relatively expensive to
traverse



- Reduce content size
- Avoid the network
- Make the network faster
- Play nicely with clients
- Stay responsive



Compression

- **Core idea: move less data around in the first place**

- Re-encode data to be smaller
 - encoding should be reversible and lossless
- Fewer bits = less data to move
 - less data = faster download!
- Most real-world data contains encoding inefficiencies
 - Text is especially guilty of this
 - Text is also the majority of what we push around the web

- **First step: minification**
 - Relies on **compilers** to understand the semantics of your code and optimize it
 - Takes your code and produces equivalent, smaller code
 - Many tools do this, some more efficiently than others
 - Google's Closure compiler
 - YUI compressor
 - UglifyJS
 - etc...

- Minification: before

```
1 function unusedFunction(note) {  
2     alert(note['text']);  
3 }  
4  
5 function displayNoteTitle(note) {  
6     alert(note['title']);  
7 }  
8  
9 var flowerNote = {};  
10 flowerNote['title'] = "Flowers";  
11 displayNoteTitle(flowerNote);
```

- After:


```
1 var a={};a.title="Flowers";alert(a.title);
```

- Minification is great, but we can do better
- minified assets are still human-readable
 - but we can encode the minified data in a *non* human-readable format
- Almost all modern web servers support **gzip**
- gzip savings for text are huge -- 70% is standard!
 - not so great for other formats (images, sounds), but can still yield some savings


- What *is* gzip?
 - Open-source, cross-platform compression tool
 - Uses DEFLATE algorithm
 - Basically replaces common occurrences of data with references to a single copy
- Why gzip?
 - Fast to decompress, awesome compression ratio for text
 - Slow to compress, but usually that's done beforehand anyway
 - Algorithm not *really* covered by patents

- Web servers won't give gzipped files unless asked to
 - Fortunately most browsers will ask for it
- gzip, other encodings specified using the **Accept-Encoding** header
 - However, it's up to the server to decide whether or not to send the gzipped assets
 - Non-plaintext responses are always indicated by the **Content-Encoding** header

- Example: Yelp's CSS package

Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency
 www-pkg-en_US.css s3-media4.ak.yelpcdn.com/assets/2/w	GET	200 OK	text/css	www.yelp.com/ :106 Parser	36.8 KB 212 KB	790 ms 281 ms

- Example: Yelp's CSS package


Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency
 www-pkg-en_US.css s3-media4.ak.yelpcdn.com/assets/2/w	GET	200 OK	text/css	www.yelp.com/ :106 Parser	36.8 KB 212 KB	790 ms 281 ms

Uncompressed asset size



- Example: Yelp's CSS package


Compressed asset size

Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency
 www-pkg-en_US.css s3-media4.ak.yelpcdn.com/assets/2/w	GET	200 OK	text/css	www.yelp.com/ :106 Parser	36.8 KB 212 KB	790 ms 281 ms

Uncompressed asset size

- Example: Yelp's CSS package

Compressed asset size

Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency
 www-pkg-en_US.css s3-media4.ak.yelpcdn.com/assets/2/w	GET	200 OK	text/css	www.yelp.com/:106 Parser	36.8 KB 212 KB	790 ms 281 ms

Uncompressed asset size

- That's ~82.6% compression ratio!
 - ...or 35ms instead of 207ms download on average DSL line

- But what about compression time?

```
fhats at yelp-fhats in ~
$ du -h www-pkg-en_US.css
216K    www-pkg-en_US.css
fhats at yelp-fhats in ~
$ time gzip www-pkg-en_US.css

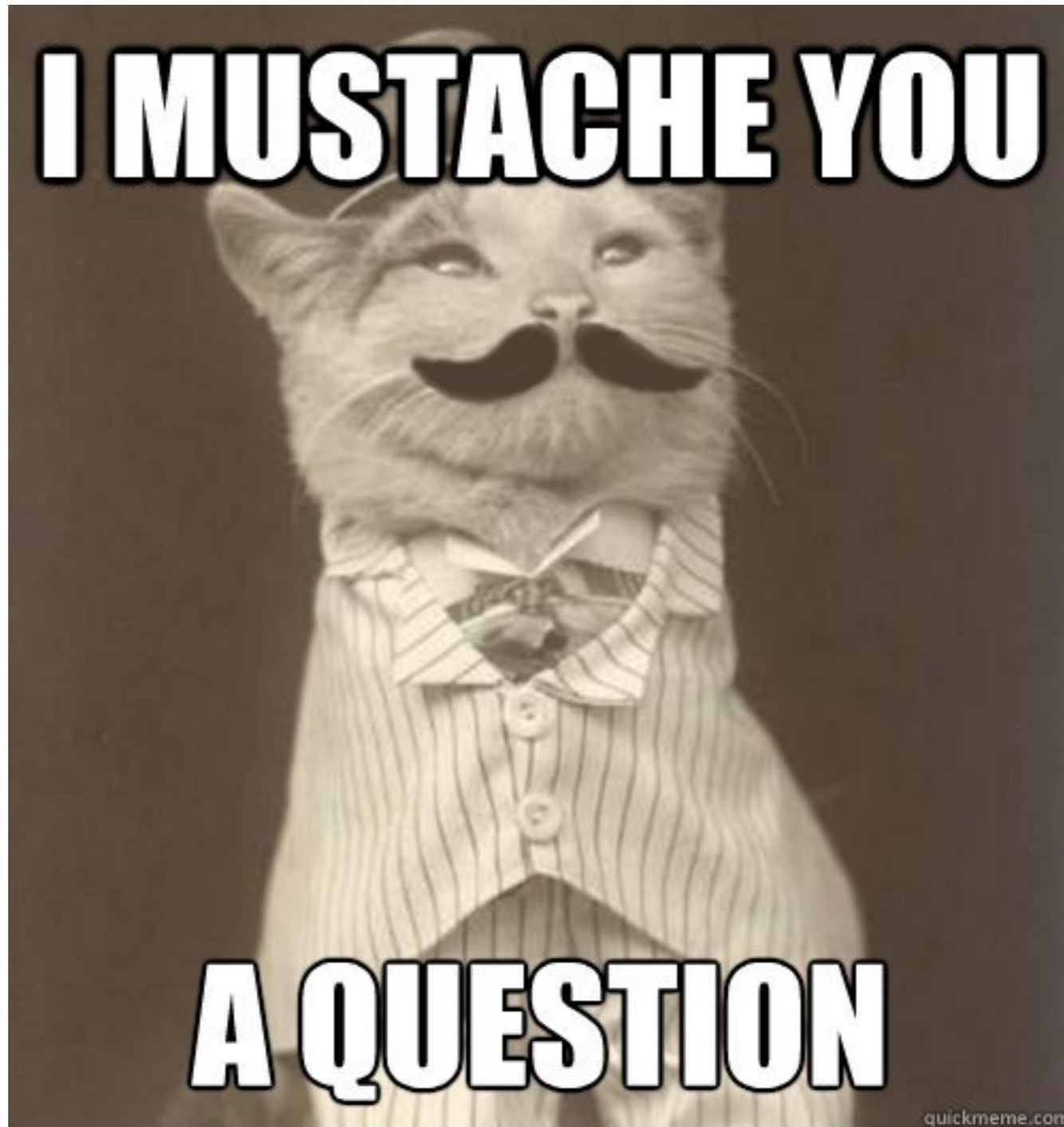
real    0m0.012s
user    0m0.010s
sys     0m0.002s
fhats at yelp-fhats in ~
$ █
```

- Compression usually done once, cached
 - but compression time usually negligible anyway



Friday, October 11, 13

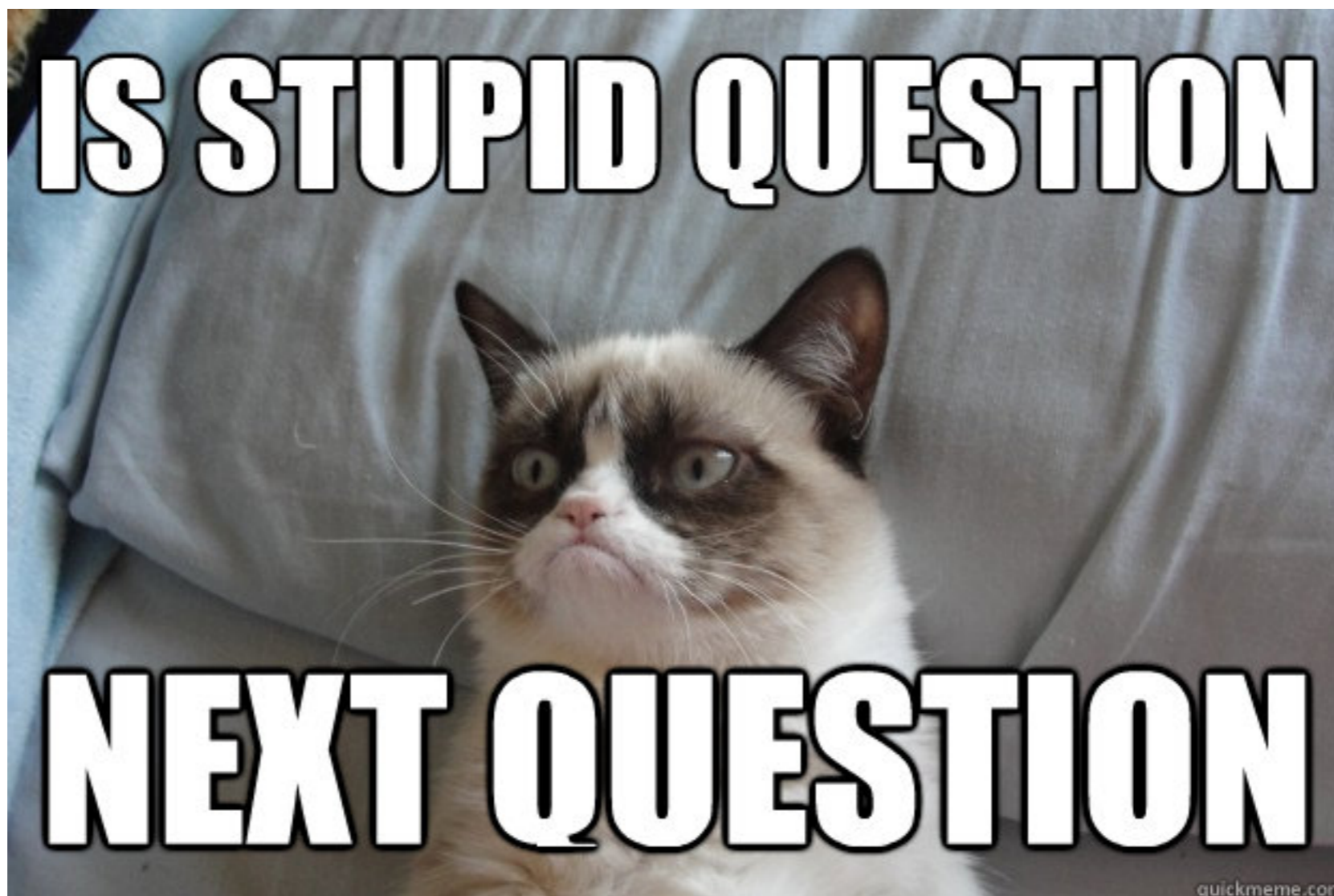
CDNs



- Problem: I need to send a letter
 - Q: There are five mailboxes near me. Which one do I pick?

- **Problem:** I need to send a letter
 - **Q:** There are five mailboxes near me. Which one do I pick?
 - **A:** The closest one!

- Problem: I need to send a letter
 - **Q:** There are five mailboxes near me. Which one do I pick?
 - **A:** The closest one!



- Problem: I need to send a ~~letter~~ **web request**
 - **Q:** There are five mailboxes near me. Which one do I pick?
 - **A:** The closest one!

- Problem: I need to send a ~~letter~~ **web request**
 - **Q:** There are five ~~mailboxes~~ **datacenters** near me. Which one do I pick?
 - **A:** The closest one!

- Problem: Light is still pretty slow across long distances

- Problem: Light is still pretty slow across long distances
- My house to elsewhere in SF:

```
fhats at ocelot in ~  
$ ping www.sfo1.yelp.com  
PING www.sfo1.yelp.com (199.255.189.60) 56(84) bytes of data.  
64 bytes from www.sfo1.yelp.com (199.255.189.60): icmp_req=1 ttl=52 time=21.7 ms  
64 bytes from www.sfo1.yelp.com (199.255.189.60): icmp_req=2 ttl=52 time=22.1 ms  
64 bytes from www.sfo1.yelp.com (199.255.189.60): icmp_req=3 ttl=52 time=23.8 ms
```

- Problem: Light is still pretty slow across long distances
- My house to elsewhere in SF:

```
fhats at ocelot in ~  
$ ping www.sfo1.yelp.com  
PING www.sfo1.yelp.com (199.255.189.60) 56(84) bytes of data.  
64 bytes from www.sfo1.yelp.com (199.255.189.60): icmp_req=1 ttl=52 time=21.7 ms  
64 bytes from www.sfo1.yelp.com (199.255.189.60): icmp_req=2 ttl=52 time=22.1 ms  
64 bytes from www.sfo1.yelp.com (199.255.189.60): icmp_req=3 ttl=52 time=23.8 ms
```

- My house to the east coast:

```
fhats at ocelot in ~  
$ ping www.iad1.yelp.com  
PING www.iad1.yelp.com (199.255.190.60) 56(84) bytes of data.  
64 bytes from www.iad1.yelp.com (199.255.190.60): icmp_req=1 ttl=50 time=88.4 ms  
64 bytes from www.iad1.yelp.com (199.255.190.60): icmp_req=2 ttl=50 time=88.0 ms  
64 bytes from www.iad1.yelp.com (199.255.190.60): icmp_req=3 ttl=50 time=88.1 ms
```

- Solutions:
 - Speed up light
 - Get more patient users
 - Put users closer to servers
 - Put servers closer to users

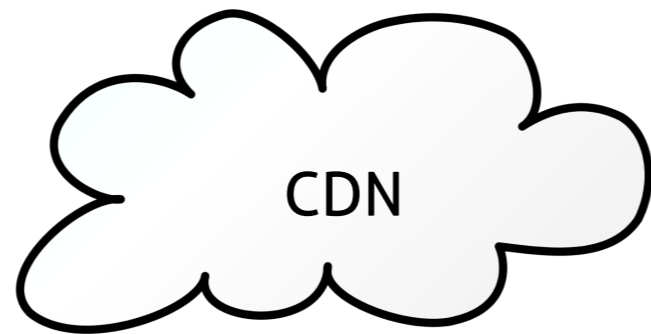
- Solutions:
 - ~~Speed up light~~
 - Get more patient users
 - Put users closer to servers
 - Put servers closer to users

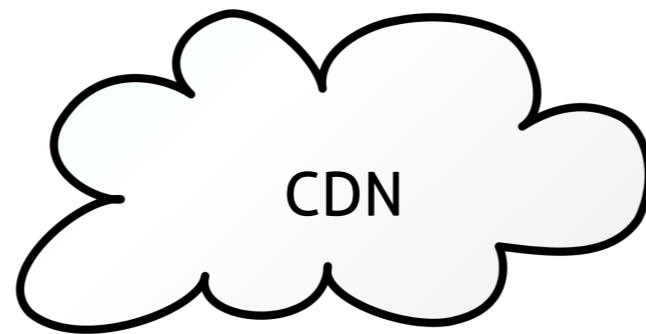
- Solutions:
 - ~~Speed up light~~
 - ~~Get more patient users~~
 - Put users closer to servers
 - Put servers closer to users

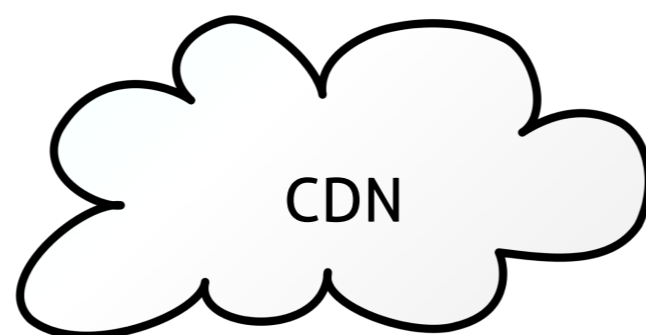
- Solutions:
 - ~~Speed up light~~
 - ~~Get more patient users~~
 - ~~Put users closer to servers~~
 - Put servers closer to users

- **CDN - Content Delivery Network**
 - enormous networks of servers
 - Akamai >100,000 servers worldwide
 - globally distributed
 - acts like a big cache of your content
 - examples:
 - Akamai
 - Cloudfront (Amazon Web Services)
 - Fast.ly
 - CDNetworks
 - etc ...

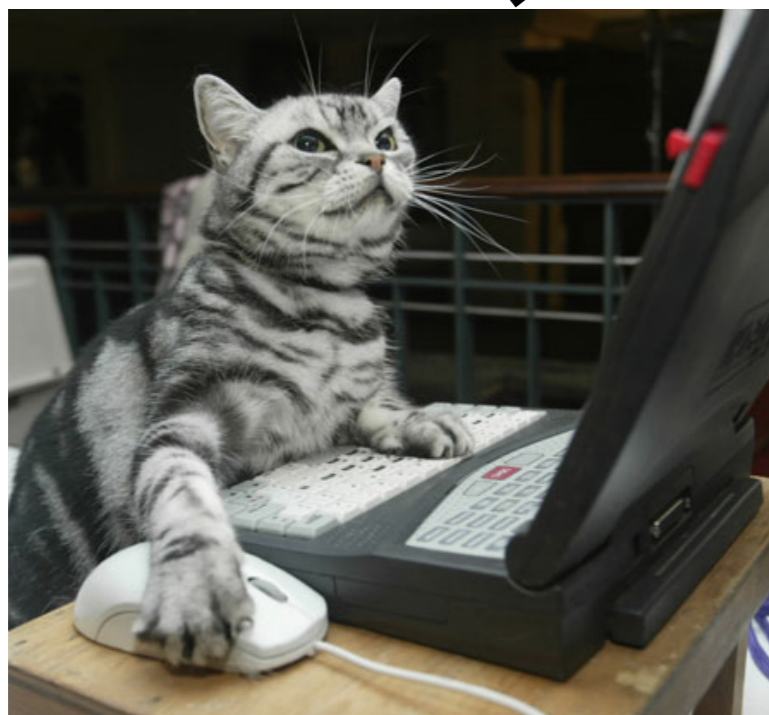
- How do CDNs work?
 - Basically: they act like a big proxy
 - The first time a CDN gets a request, it fetches the request from an **origin**
 - After the first request, the CDN saves a copy of the origin's response
 - Every time that URL is requested again, the CDN returns its saved copy of the asset

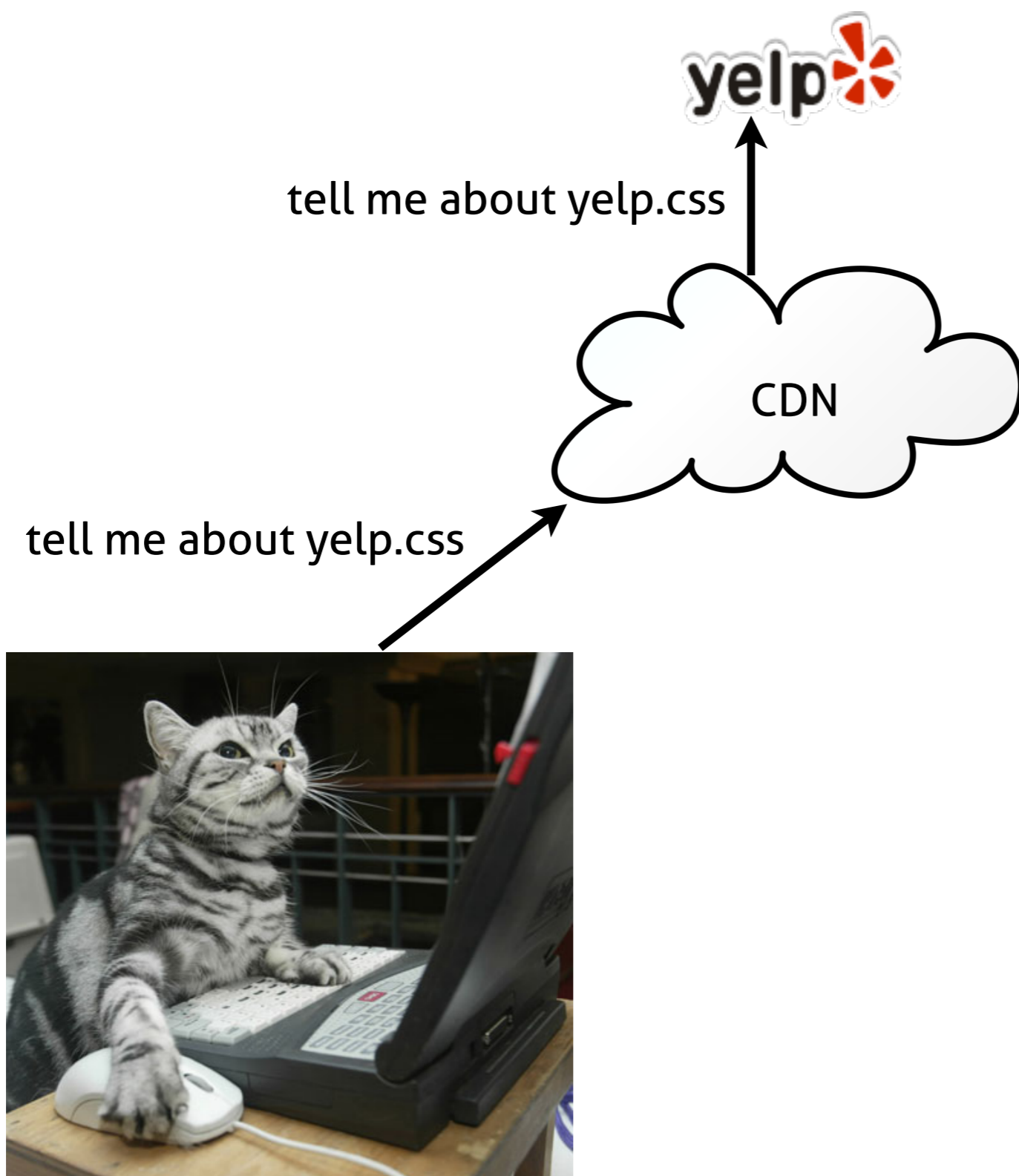


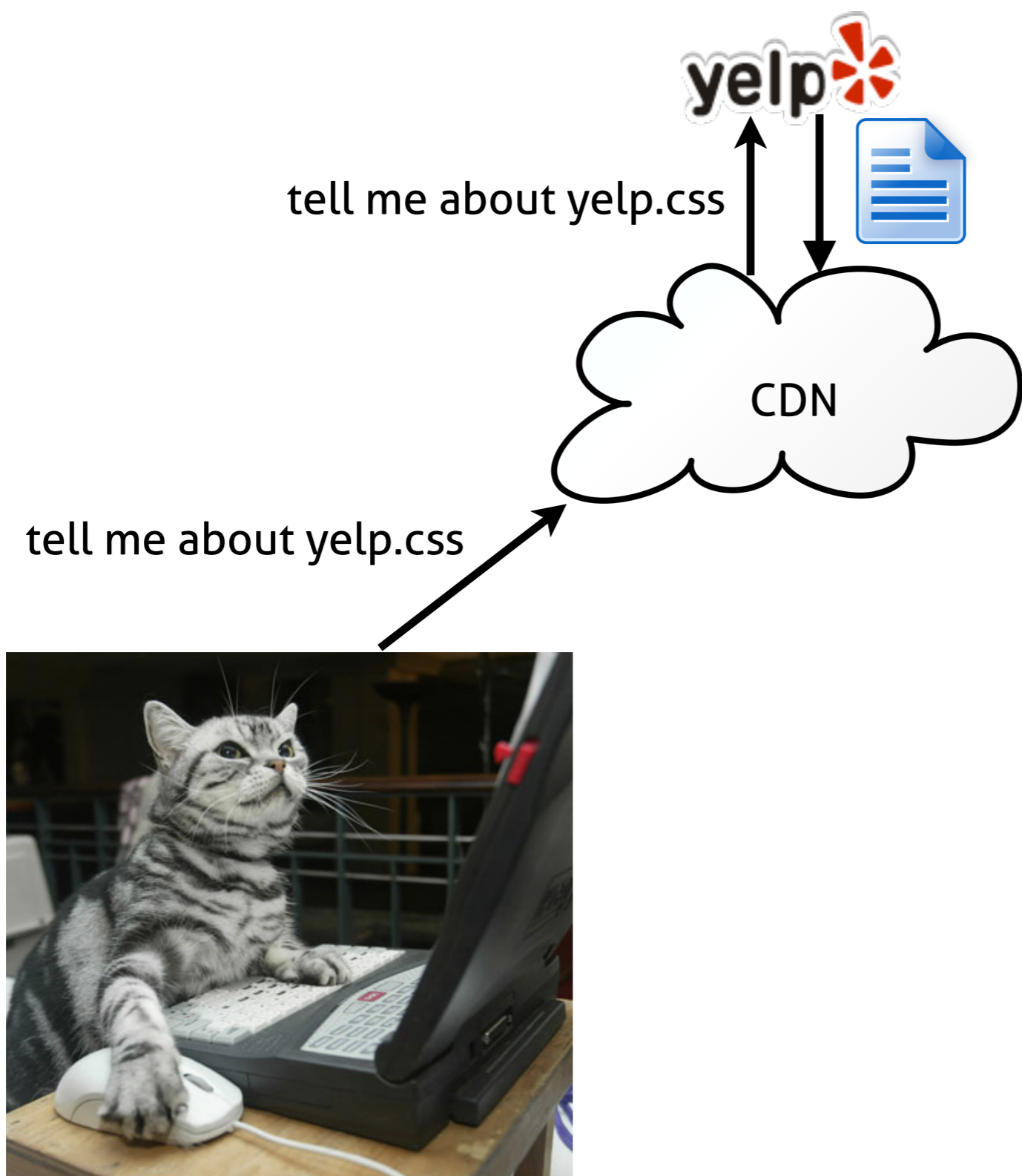


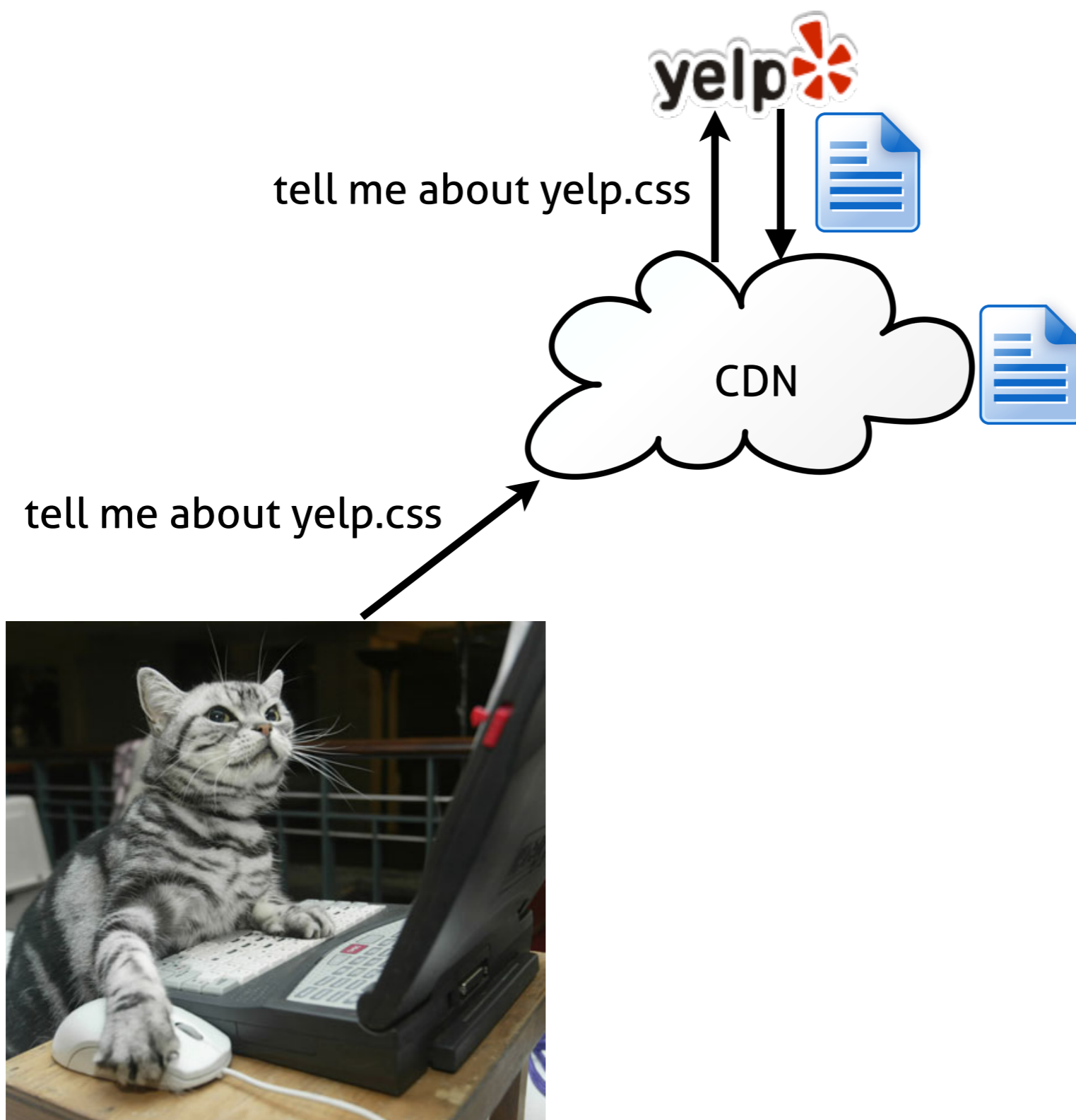


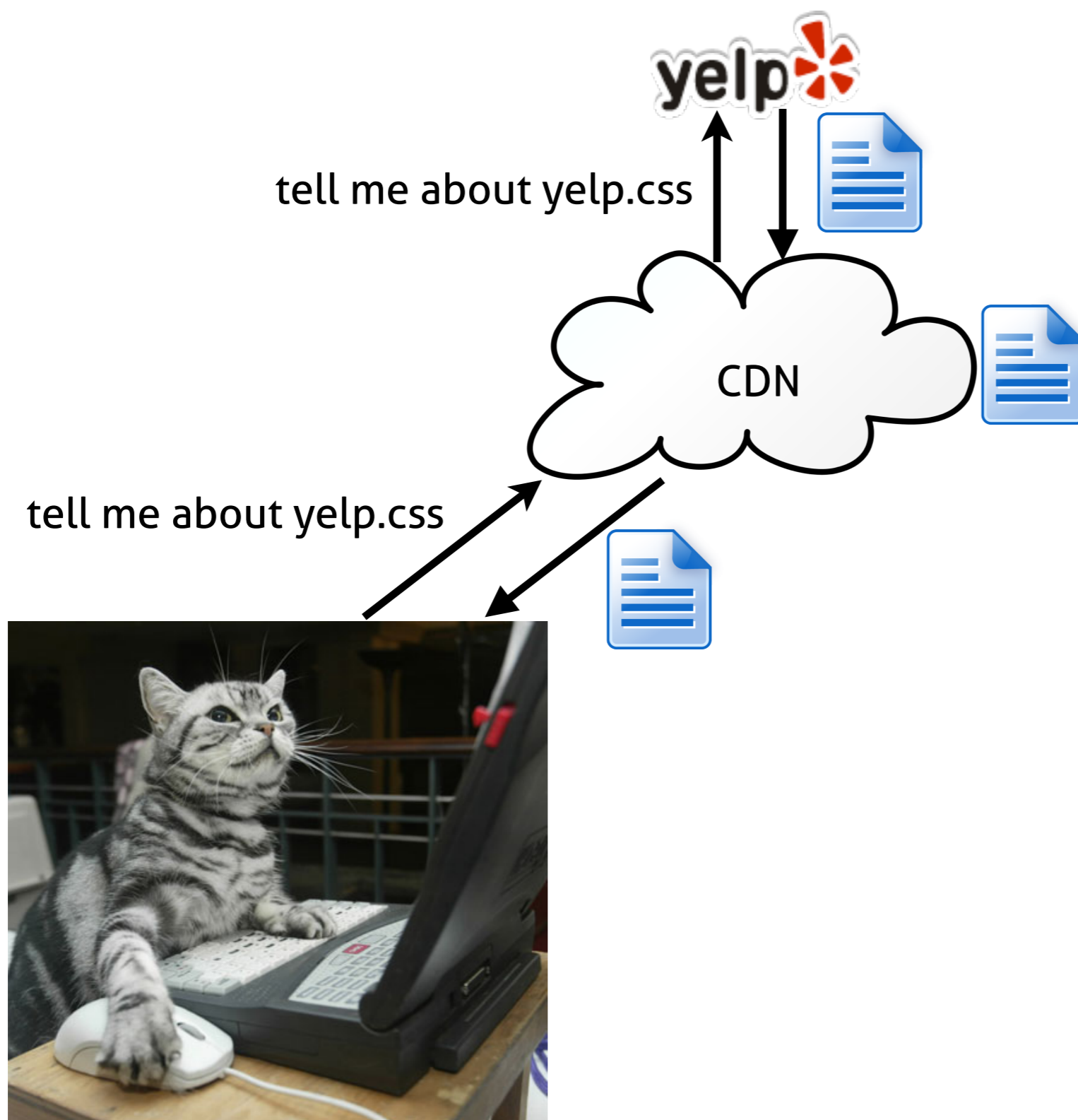
tell me about yelp.css

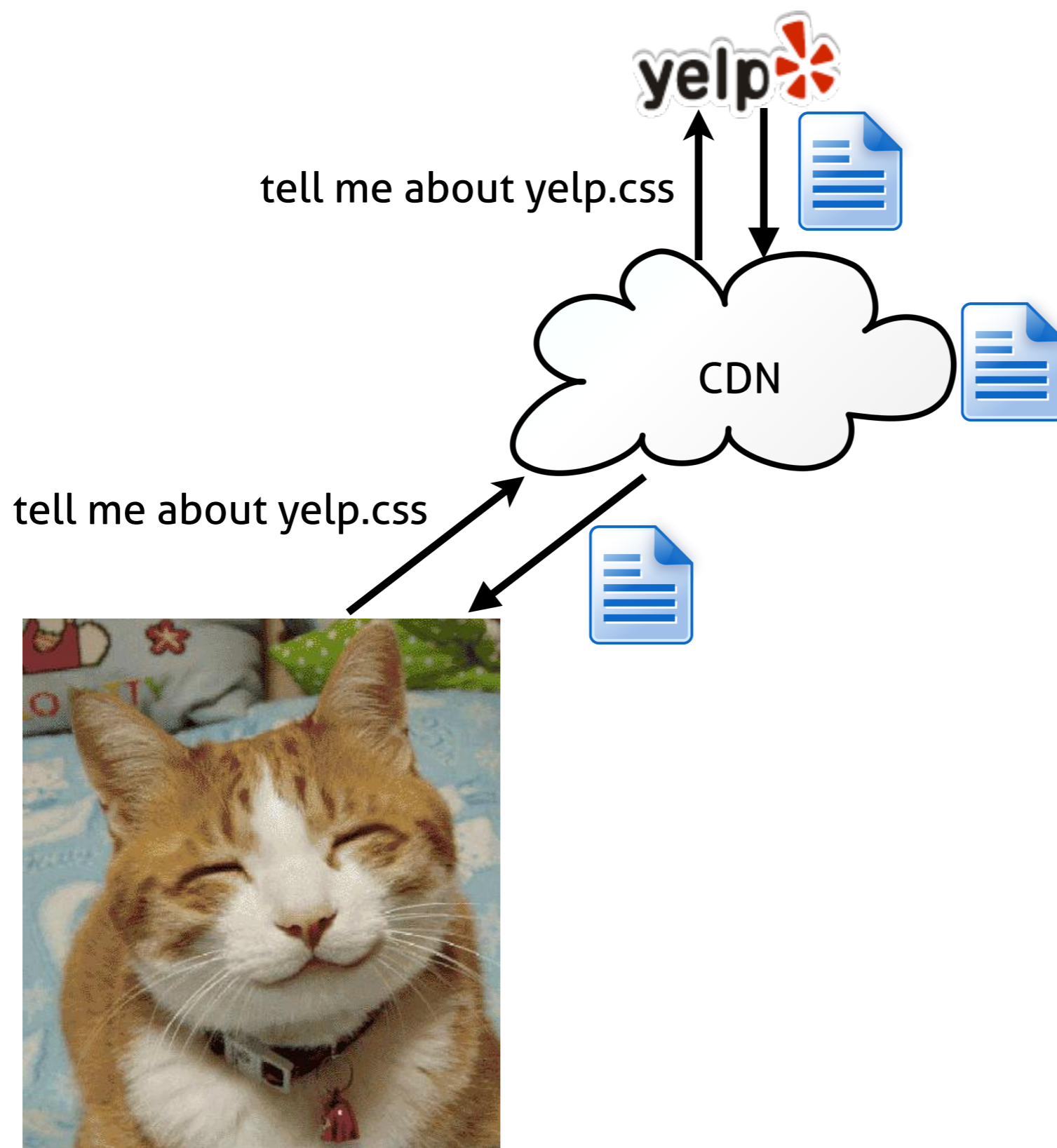


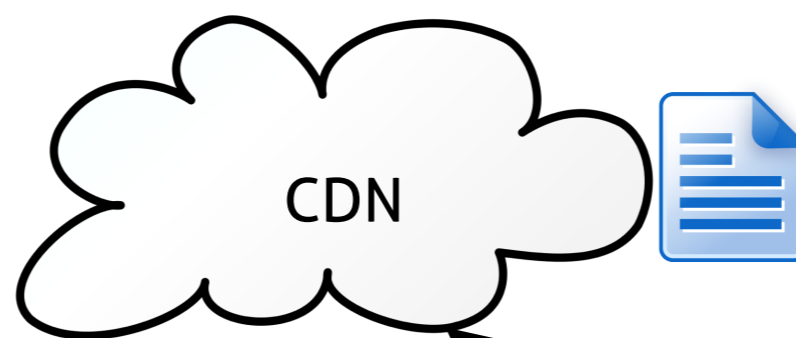




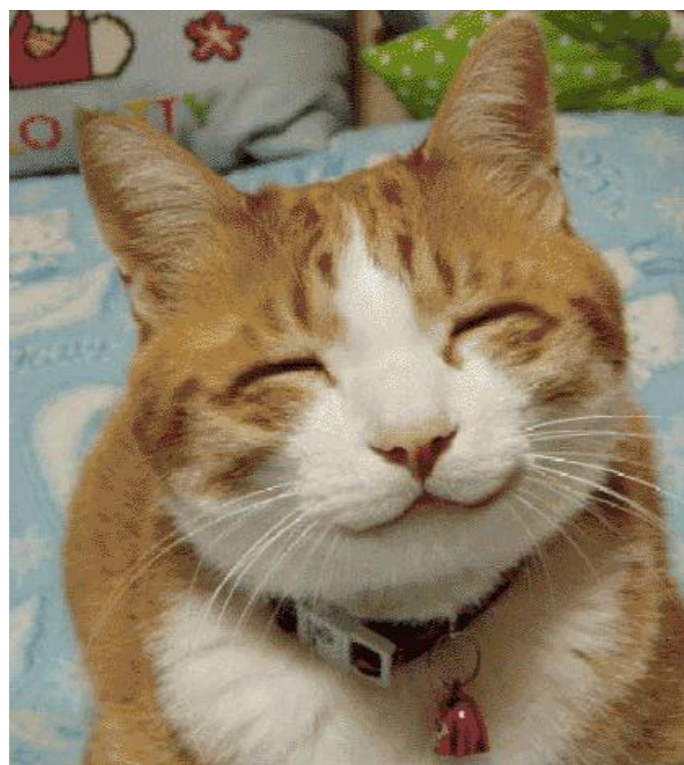


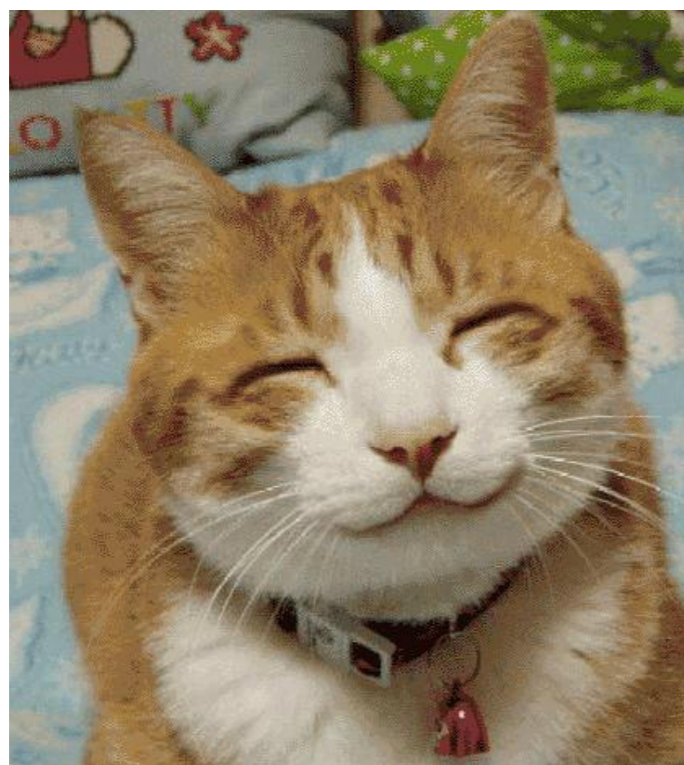
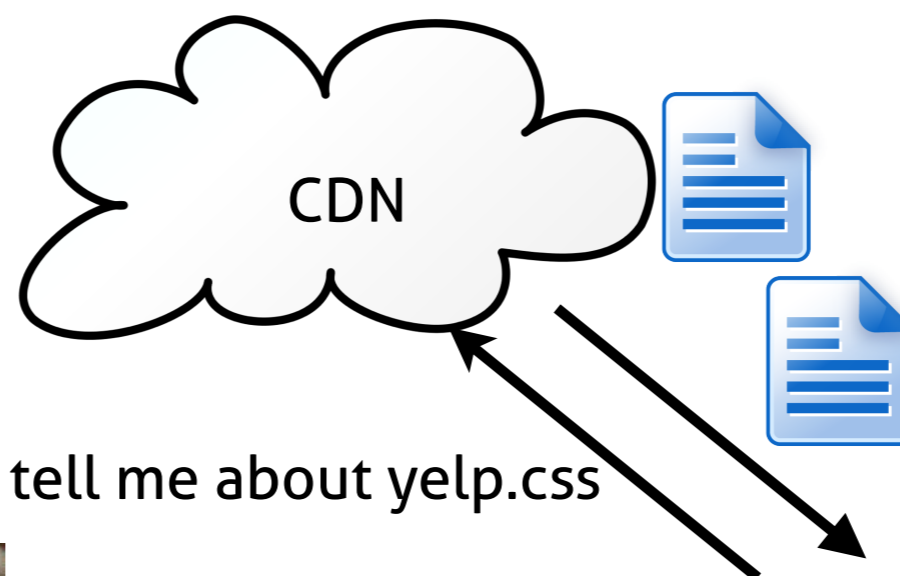


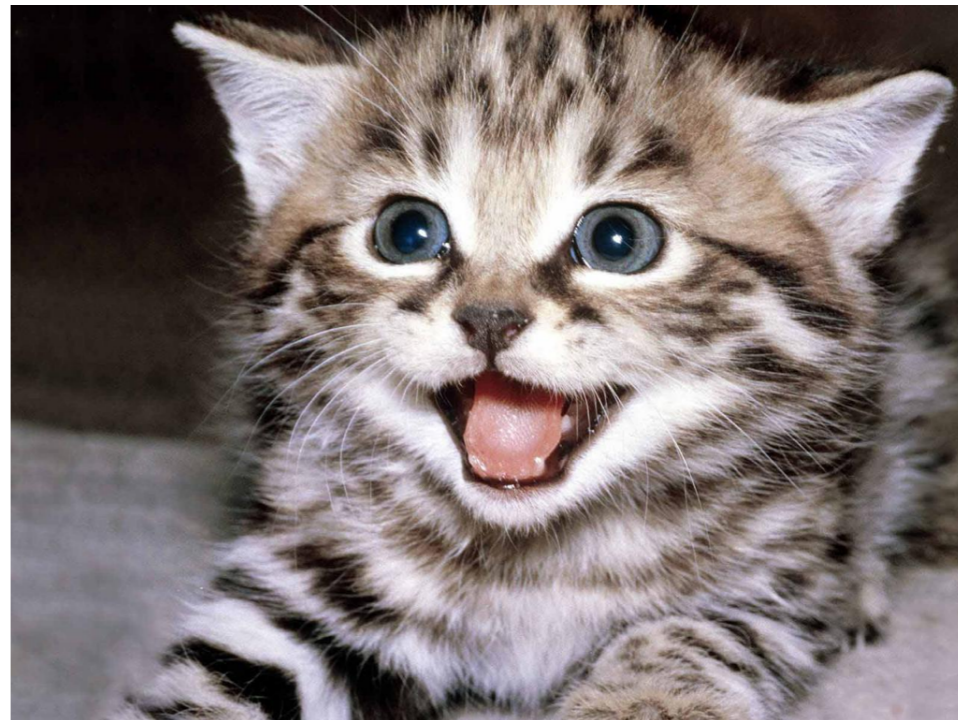
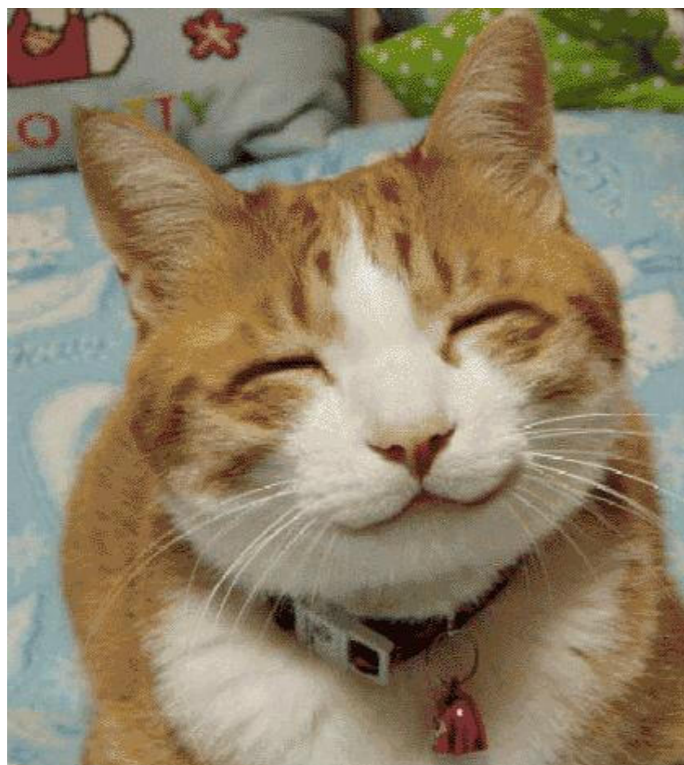
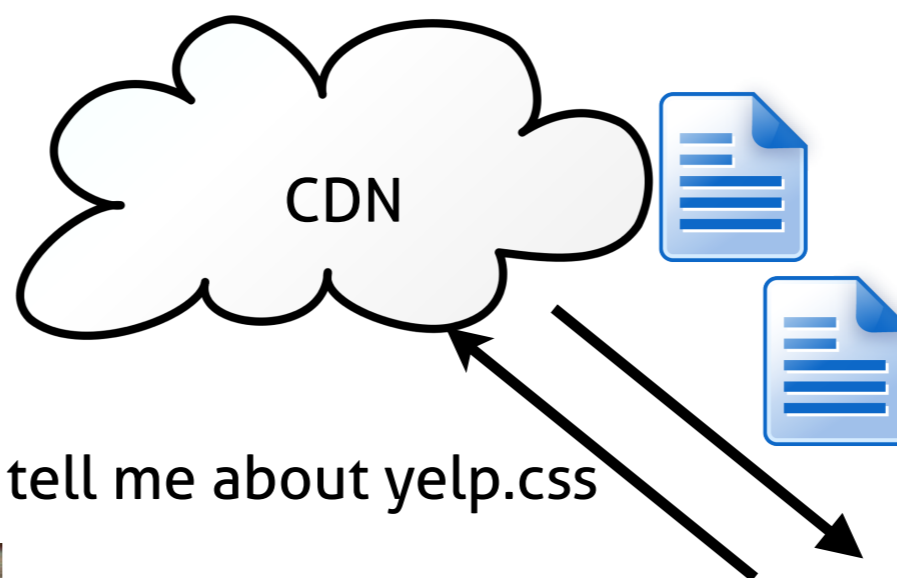




tell me about `yelp.css`







- **Q:** How can different servers all around the world be at the same internet address?
- **A:** Anycast + DNS!
 - DNS servers inject themselves into routing tables all around the world
 - DNS servers serve records only for servers closest to them

- Is it really that much faster?
 - pix or it didn't happen

```
fhats at ocelot in ~
$ time curl -s http://s3-media3.ak.yelpcdn.com/photo/JlhIY94GIcLZxsUmIWYCiA/60s.jpg >/dev/null

real    0m0.189s
user    0m0.004s
sys     0m0.000s
fhats at ocelot in ~
$ time curl -s (cdn origin) .com/photo/JlhIY94GIcLZxsUmIWYCiA/60s.jpg >/dev/null

real    0m0.357s
user    0m0.000s
sys     0m0.008s
```

- **Tip: Keep a backup CDN around**
 - CDNs are really good at offloading traffic
 - CDNs are also really good at being available
 - ...but if they *do* fail, you've got a thundering herd of people all asking you for way more traffic than you were planning on
 - Having a backup CDN can keep you from being completely toppled if your CDN has trouble
 - It can also help you keep your primary CDN honest

Subdomain Sharding


RFC 2616 (HTTP 1.1):

“A single-user client **SHOULD NOT** maintain more than **2 connections** with any server or proxy.”

- ...but my yelp.com homepage has 102 external assets!
 - if each asset takes 50 ms to download...
 - ...and I can only download 2 assets at a time...
 - ...then it will take me 2.5 seconds **just** to download external assets for yelp.com

- We can do better!

- **Aside: Parallel connections by browser:**

-  Firefox: 6
-  Chrome: 6
-  Opera: 8
-  Internet Explorer: 2-6
- (per-host)

- **Goal:** trick browsers into thinking you are more websites than you actually are
- **Solution:** put your static (CDN-backed) content behind different subdomains

- Without sharding:
 - media.yelp.com -> 184.28.79.144
- With sharding:
 - media1.yelp.com -> 184.28.79.144
 - media2.yelp.com -> 184.28.79.144
 - media3.yelp.com -> 184.28.79.144
 - media4.yelp.com -> 184.28.79.144

```
<title>San Francisco Restaurants, Dentists, Bars, Beauty Salons, Doctors</title>
<link rel="stylesheet" type="text/css" media="all" href="http://s3-media4.ak.yelpcdn.com/assets/2/www/css/d65342206dc8/www-
esheet" type="text/css" media="all" href="http://s3-media2.ak.yelpcdn.com/assets/2/www/css/c6d380bf68c4/homepage-en_US.css">
<link rel="alternate" type="application/rss+xml" title="Yelp - Recent Reviews Near San Francisco, CA"
om/syndicate/area/rss.xml?loc=San+Francisco%2C+CA">
```


- my yelp.com homepage with 4 shards
 - ...still with 102 assets...
 - ...each averaging about 50 ms to download...
 - ...can now download 8 assets at a time...
 - ...now only takes .6 seconds!
 - according to the RFC. real browsers will use more connections.

- **Pitfall: DNS lookups**
 - Because each shard is a different hostname, each shard requires DNS lookups
 - **DNS lookups are expensive**, and block the browser from doing anything else until finished

```
fhats at ocelot in ~  
$ time host yelp.com >/dev/null  
  
real    0m0.023s  
user    0m0.000s  
sys     0m0.004s
```

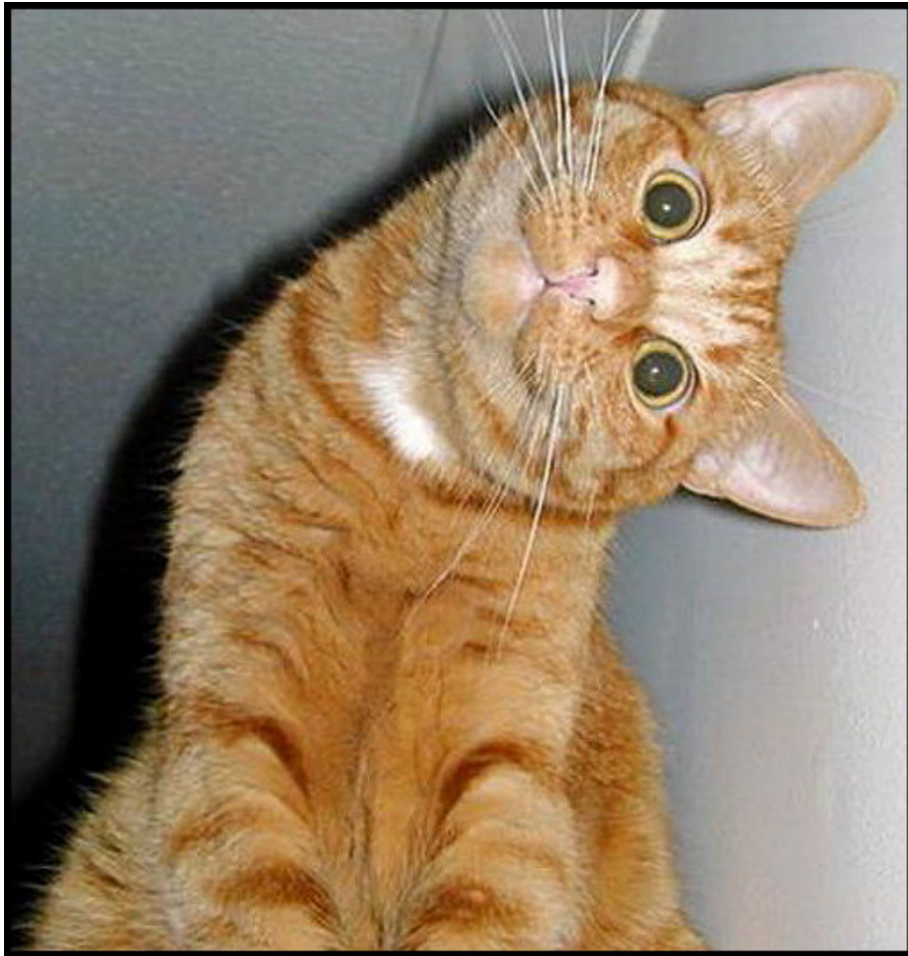
- **Solution:** only use 2-4 subdomain shards
 - ...that's still ~12-24 concurrent connections on modern browsers

Why Are Round-Trips Bad?

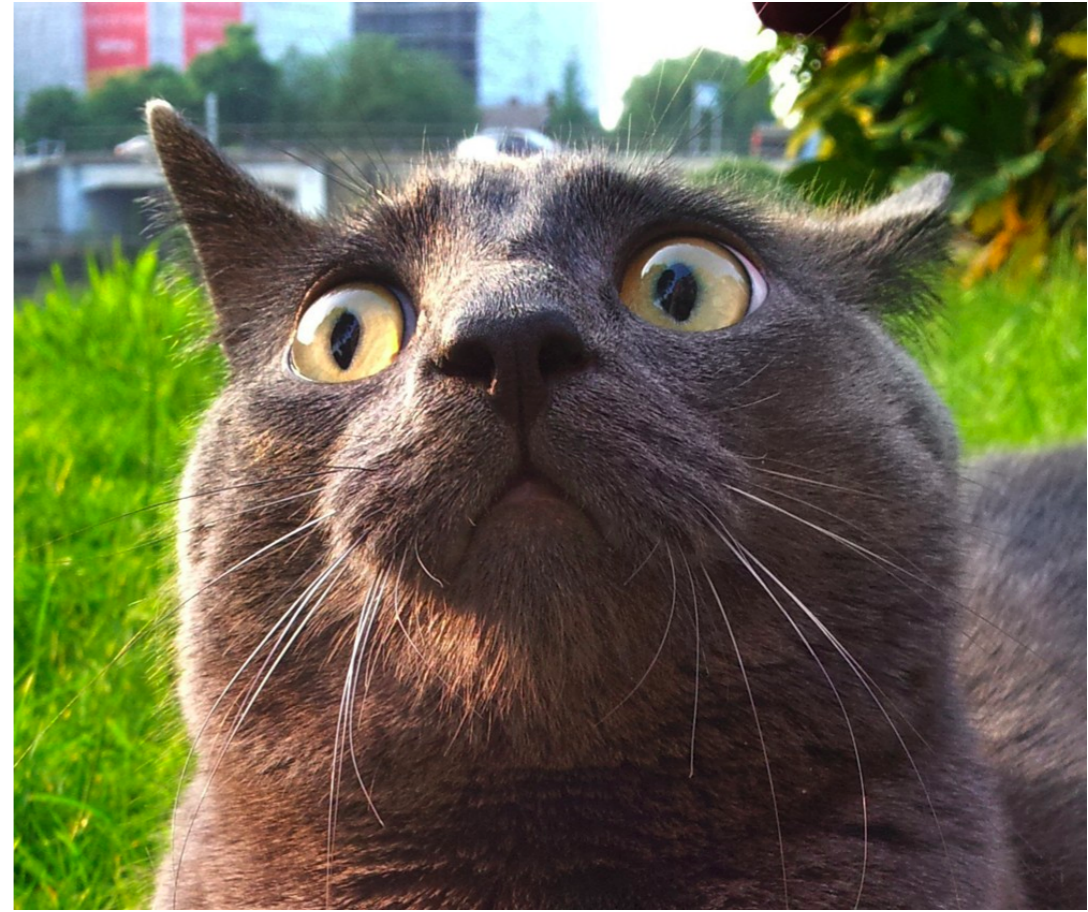
- A brief aside: why are round-trips bad?
- Initiating TCP connections is *slow*
 - especially when you have to do it a lot!
- Why is initiating a TCP connection slow?
 - Three-way handshake
 - TCP slow-start/congestion control

- TCP Three-Way Handshake

Client

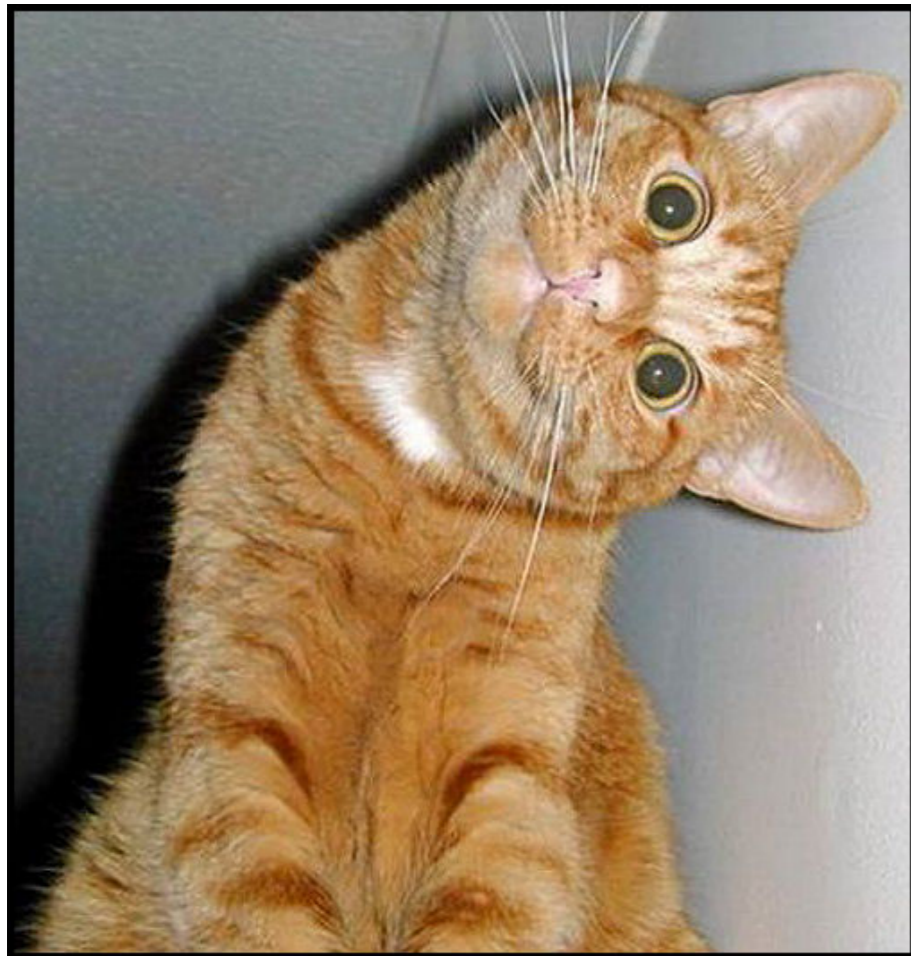


Server

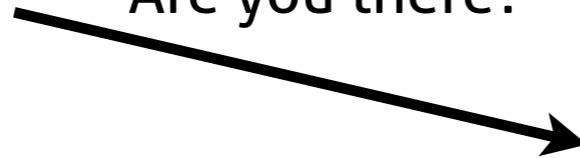


- TCP Three-Way Handshake

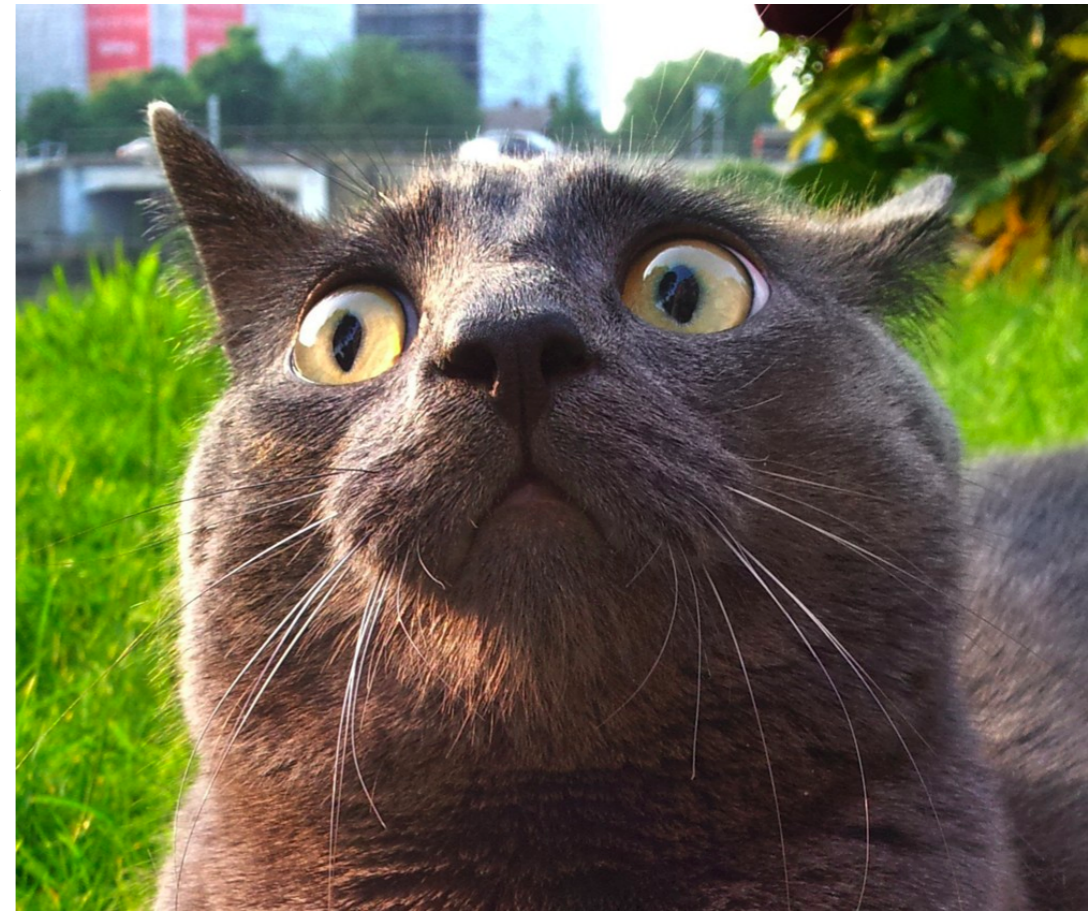
Client



"Are you there?"

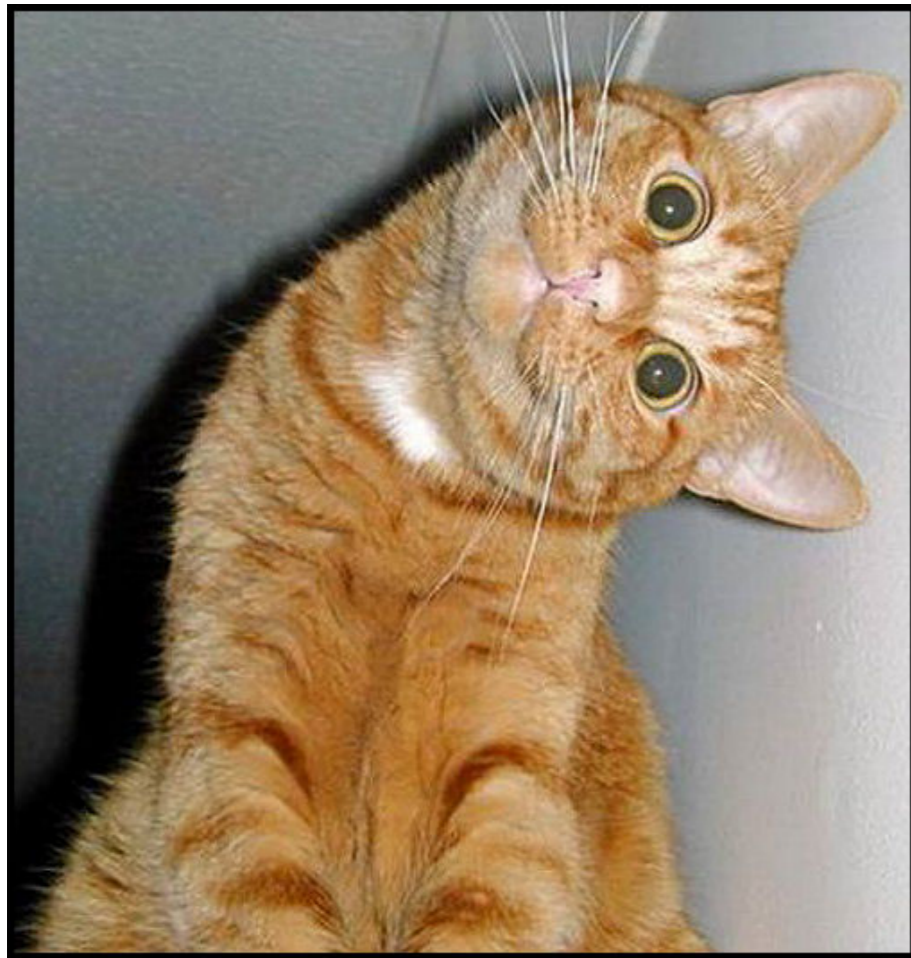


Server

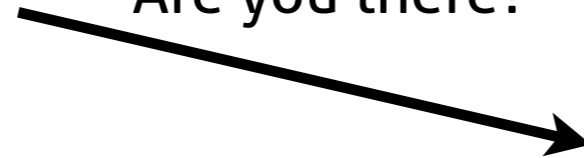


- TCP Three-Way Handshake

Client



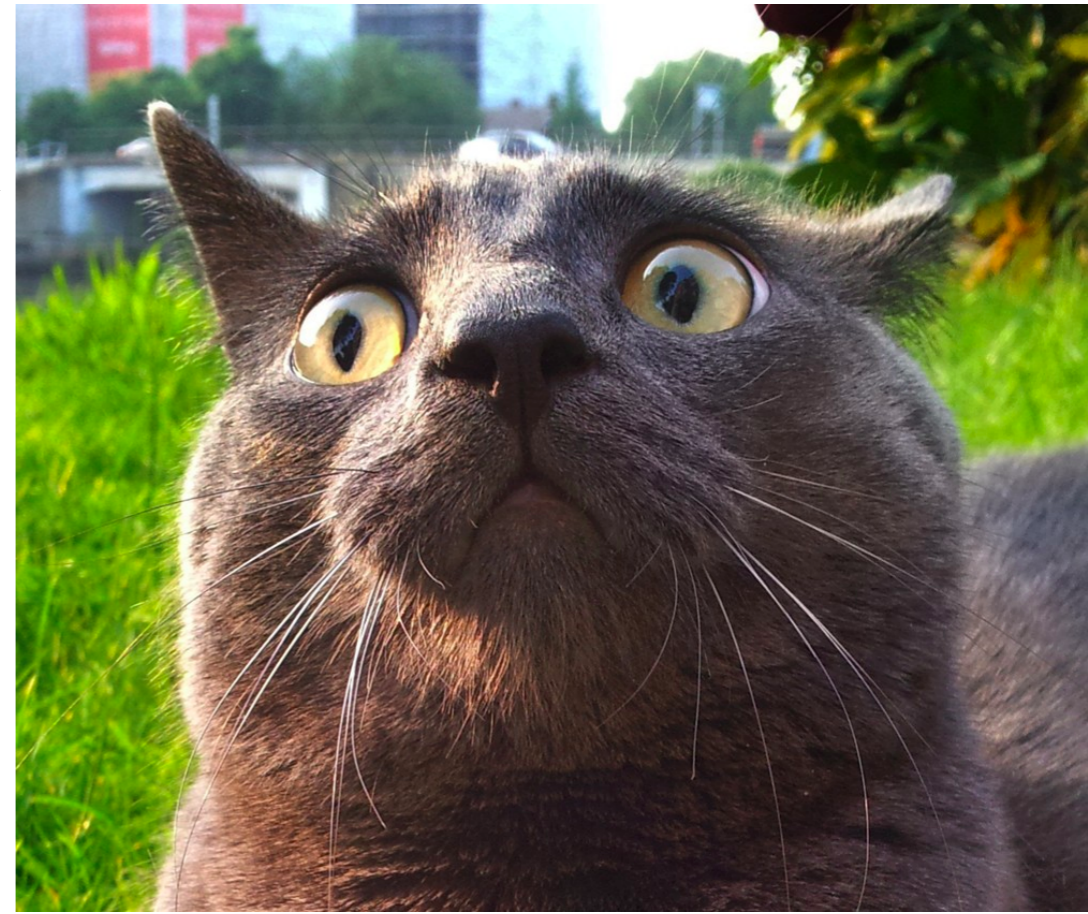
"Are you there?"



"Yeah, I'm here.
Are you there?"

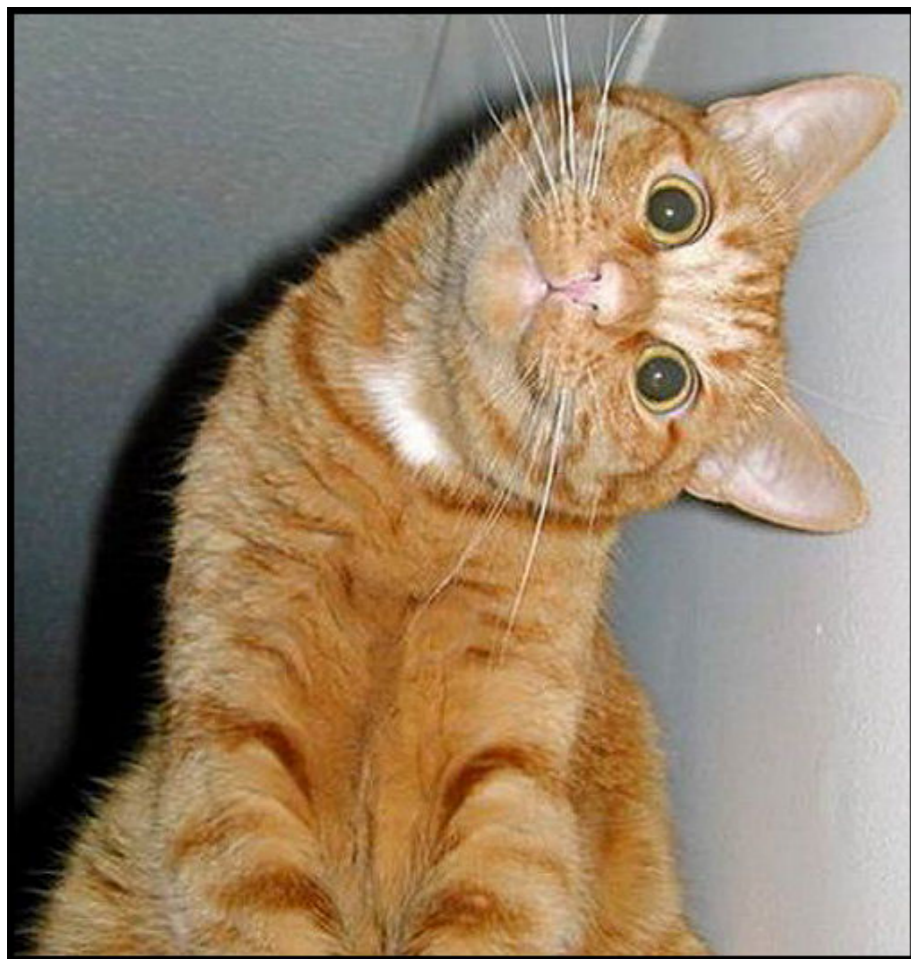


Server

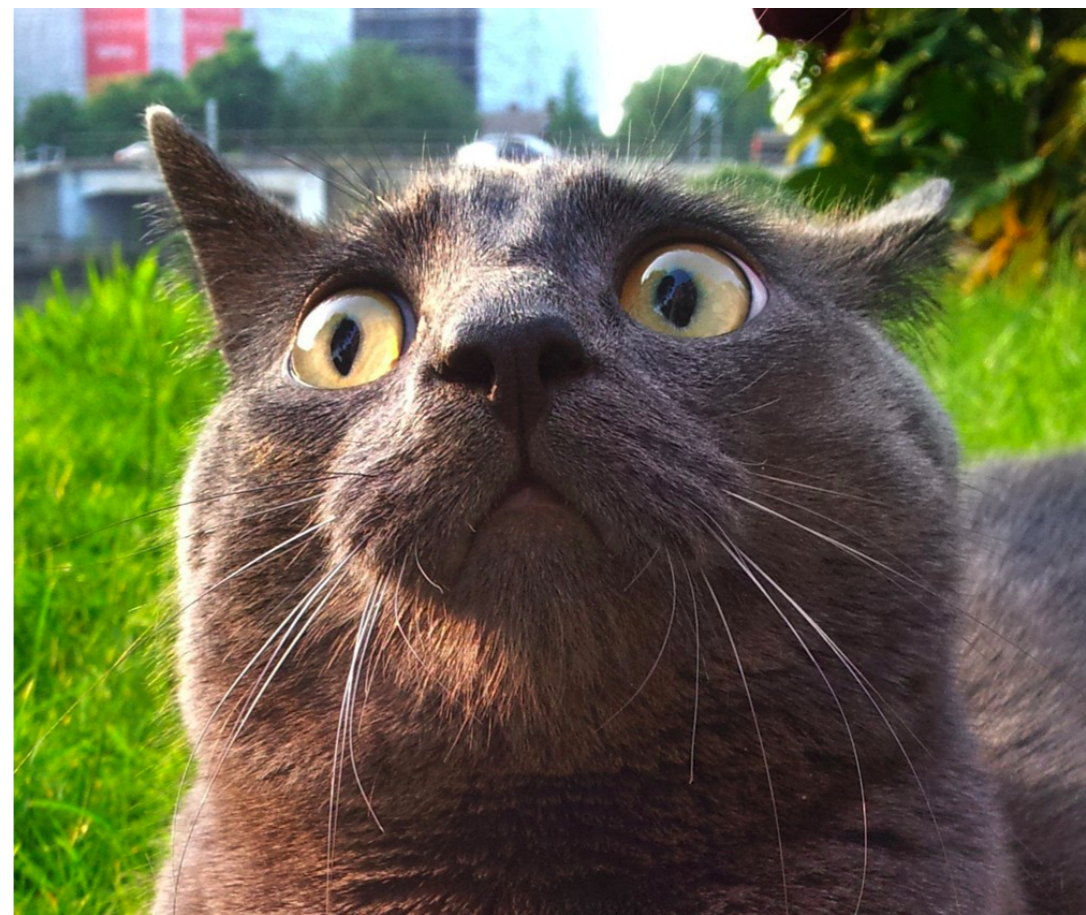


- TCP Three-Way Handshake

Client



Server



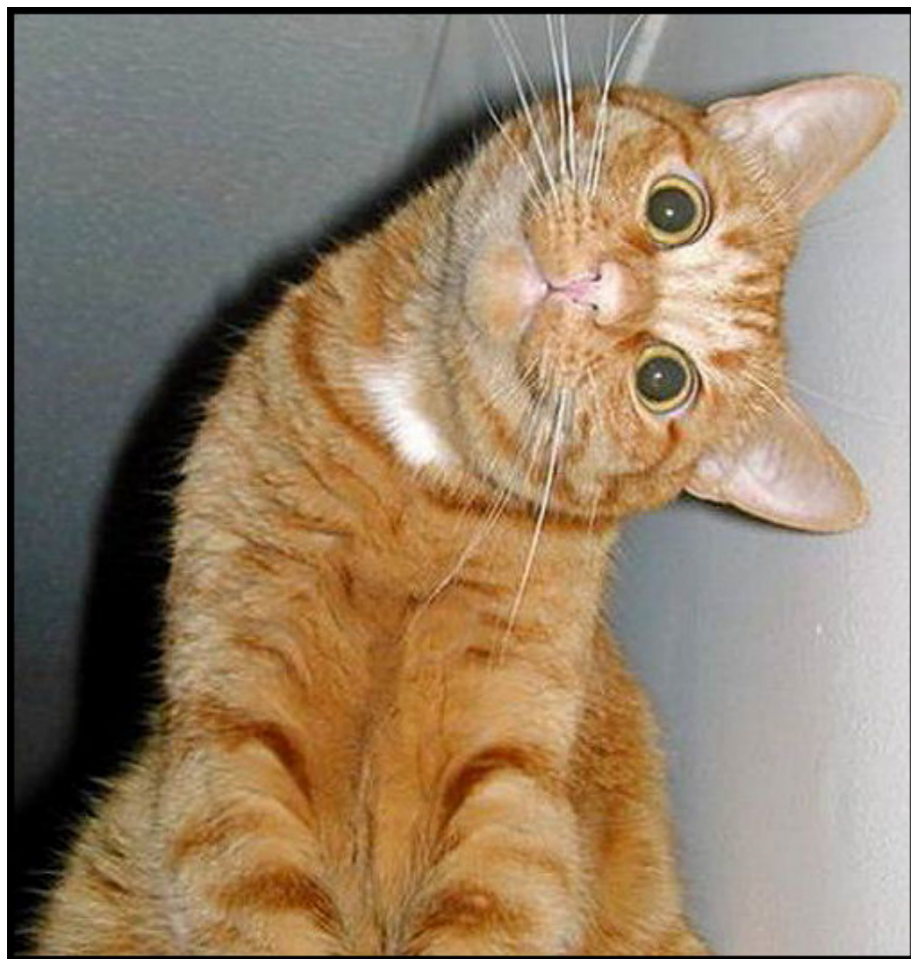
"Are you there?"

"Yeah, I'm here.
Are you there?"

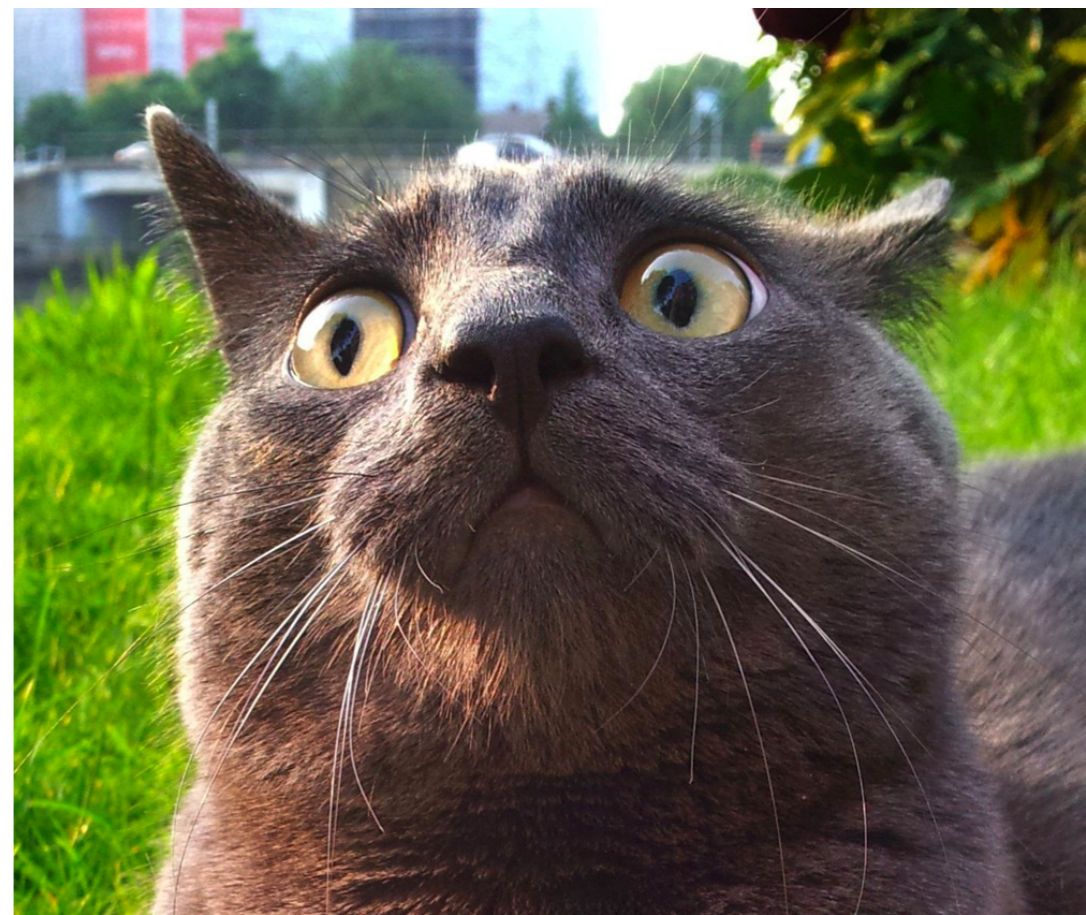
"Yep"

- TCP Three-Way Handshake

Client



Server



"Are you there?"

(10ms)

"Yeah, I'm here.

Are you there?"

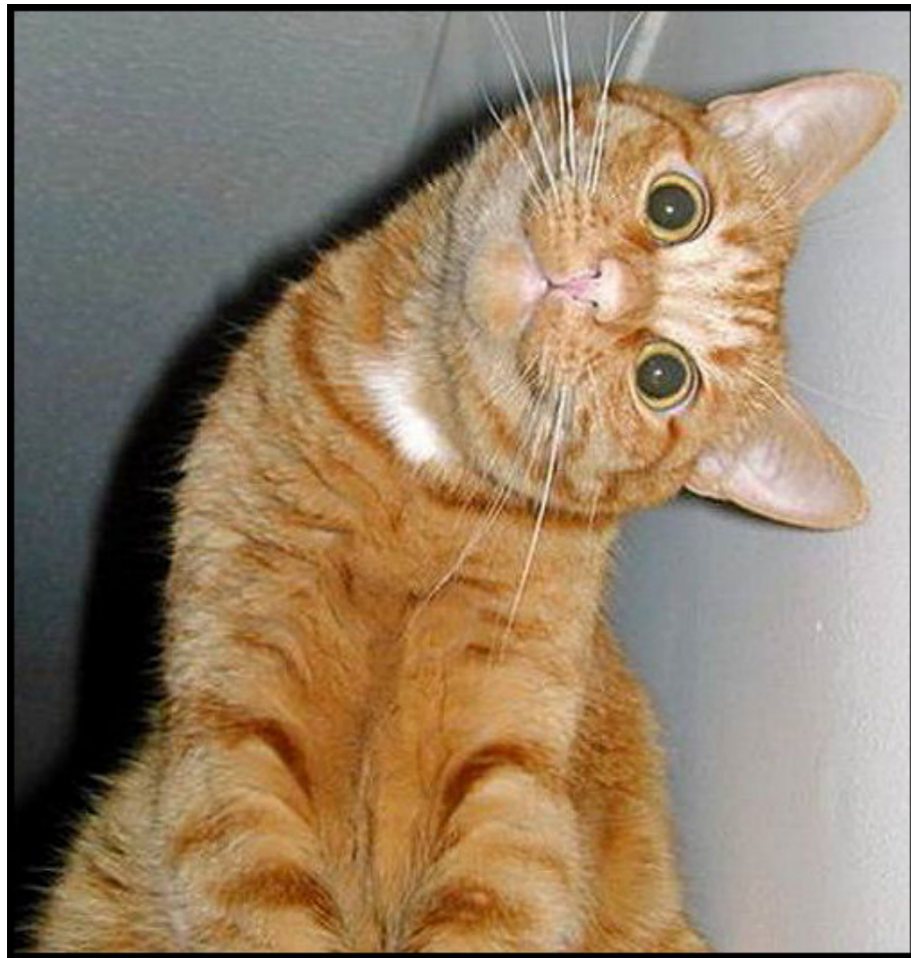
(10ms)

"Yep"

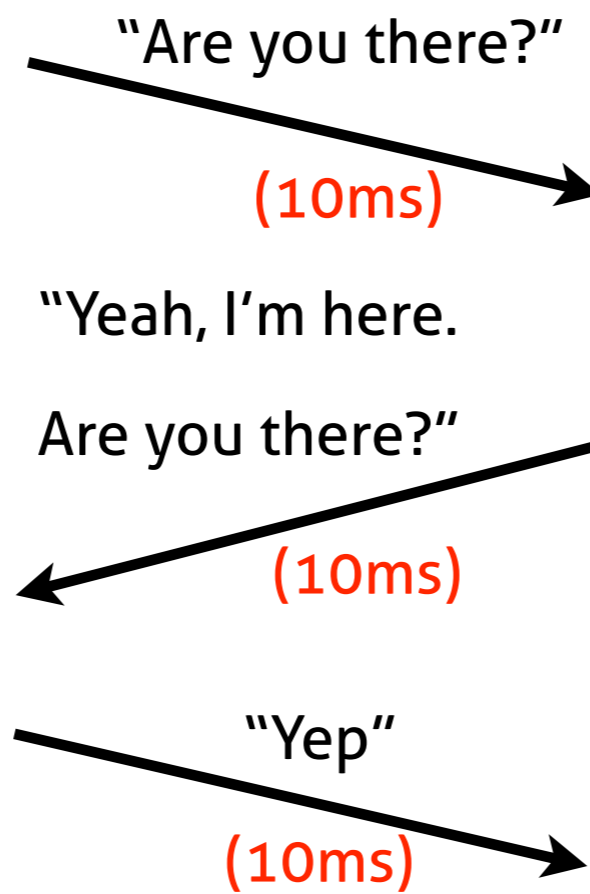
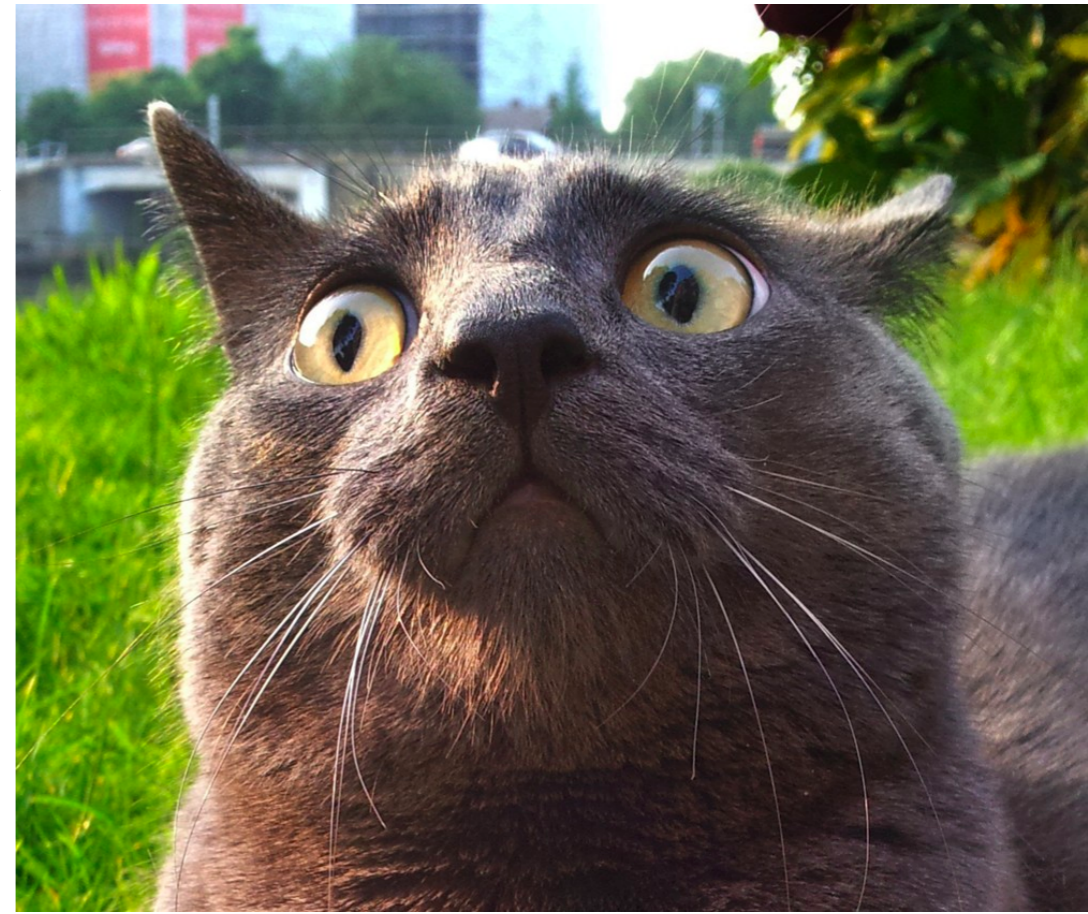
(10ms)

- TCP Three-Way Handshake

Client



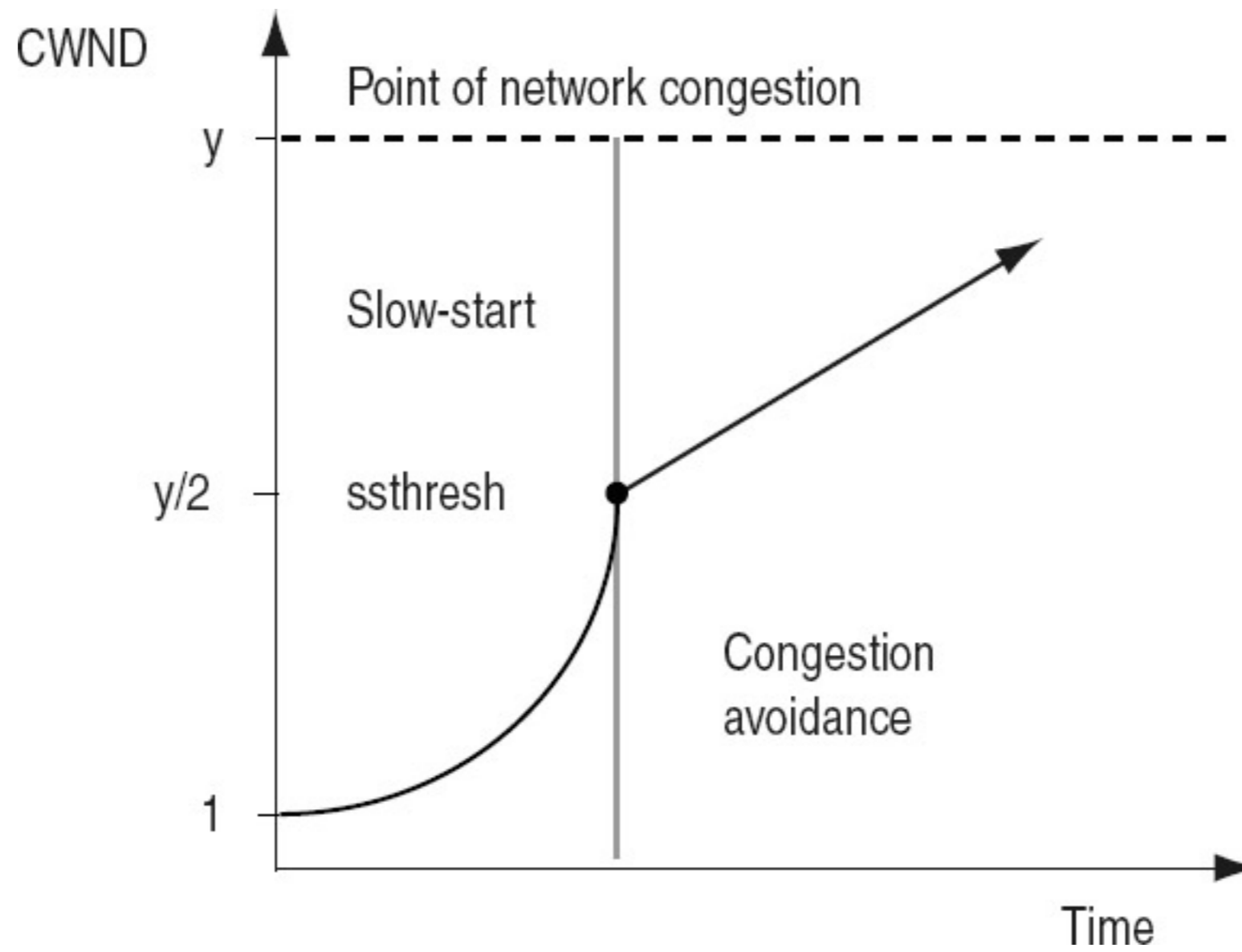
Server



30ms... before we can start sending data?!

- TCP Slow Start
 - TCP tries to limit link congestion
 - TCP **congestion window** limits how much data can be sent at one time before the receiver needs to acknowledge it
 - congestion window starts very small and grows exponentially
 - great for long-lived connections
 - terrible for short ones

Reducing Round-Trips



Asset Pipelining

- Reduce number of roundtrips by combining assets together
- Javascript, CSS, and images can all be combined together
- Round trips correlate with the number of **types** of assets, rather than number of **total** assets

- Combine Javascript and CSS all into single files

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js"></script>  
<script src="//fhatpipe-medial.fredhatfull.com/js/bootstrap/bootstrap.min.js"></script>  
<script src="//fhatpipe-media3.fredhatfull.com/js/iso8601.min.js"></script>
```

```
<script src="//fhatpipe-medial.fredhatfull.com/js/web.js"></script>
```

```
/body>
```

- Combine Javascript and CSS all into single files

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js"></script>  
<script src="//fhatpipe-medial.fredhatfull.com/js/bootstrap/bootstrap.min.js"></script>  
<script src="//fhatpipe-media3.fredhatfull.com/js/iso8601.min.js"></script>  
  
<script src="//fhatpipe-medial.fredhatfull.com/js/web.js"></script>  
/body>
```



- Combine Javascript and CSS all into single files

```
<script src="https://github.global.ssl.fastly.net/assets/frameworks-2380aeb62de9a4760c888de666aabb294cccaae1.js" type="text/javascript"></script>
```

- Sprite images together
 - Make all of your smaller images into one big image
 - ...you know, like game programmers do!



- Sprite images together
 - You can select the image you want in a sprite by using CSS's **background-image** and **background-position** properties, as well as **width** and **height**:

```
1 .homepage-user-social {
2     display: inline-block;
3     top: 0;
4     width: 16px;
5     height: 16px;
6     background-image: ...;
7     background-repeat: no-repeat;
8 }
9
10 .homepage-badge-user-social {
11     background-position: -3.0px -311.0px;
12 }
```

Friday, October 11, 13

From my colleague JR's presentation on similar things: http://jrheard.com/frontend_long/#slide19

You should go buy that guy a beer. He's super cool.

Caching

- Idea: avoid round-trips entirely when data has not changed
- **Solution:** tag each resource with the date it was modified or identifier for version
 - HTTP Response Header: **Last-Modified** or **ETag**

```
Request Method: GET
Status Code: 200 OK
▼ Request Headers view source
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
  Accept-Encoding: gzip,deflate,sdch
  Accept-Language: en-US,en;q=0.8
  Connection: keep-alive
  Host: jrheard.com
  User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/30.0.1599.69 Safari/537.36
▼ Response Headers view source
  Accept-Ranges: bytes
  Connection: Keep-Alive
  Content-Length: 1531135
  Content-Type: text/html
  Date: Fri, 11 Oct 2013 08:42:26 GMT
  Keep-Alive: timeout=5, max=75
  Last-Modified: Thu, 18 Oct 2012 23:51:24 GMT
  Server: Apache
```

- Browser asks for data using a **conditional GET**
 - HTTP Header: **If-Modified-Since, If-None-Match**
 - Asset only returned if data has changed

```
Request Method: GET
Status Code: 304 Not Modified
Request Headers view source
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
Cache-Control: max-age=0
Connection: keep-alive
Host: jrneard.com
If-Modified-Since: Thu, 18 Oct 2012 23:51:24 GMT
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/30.0.1599.69 Safari/537.36
Response Headers view source
Connection: Keep-Alive
Date: Fri, 11 Oct 2013 08:44:17 GMT
Keep-Alive: timeout=5, max=75
Server: Apache
```

- **Pros:**
 - Data only returned if it has changed since the last time the client asked for it
- **Cons:**
 - Still requires an entire round-trip just to get a response from the server that nothing has changed
- **We can do better**

- Core: asset at a URL never changes
 - when assets need to change, tell clients to ask for a new URL corresponding to the changed asset
- Tell browsers to cache for a very long time
 - HTTP Headers: **Expires, Cache-Control**

Request URL: http://s3-media4.ak.yelpcdn.com/assets/2/www/css/d65342206dc8/www-pkg-en_US.css

Request Method: GET

Status Code: ● 200 OK

▼ Request Headers [view source](#)

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8

Accept-Encoding: gzip, deflate, sdch

Accept-Language: en-US,en;q=0.8

Cache-Control: max-age=0

Connection: keep-alive

Host: s3-media4.ak.yelpcdn.com

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/30.0.1599.69 Safari/537.36

▼ Response Headers [view source](#)

Accept-Ranges: bytes

Cache-Control: max-age=315360000

Connection: keep-alive

Content-Encoding: gzip

Content-Length: 37366

Content-Type: text/css

Date: Fri, 11 Oct 2013 08:57:50 GMT

Server: AmazonS3

Vary: Accept-Encoding

x-amz-version-id: 2M5BMvaMU2wb5laW0aLEiXCiFiU0js6M

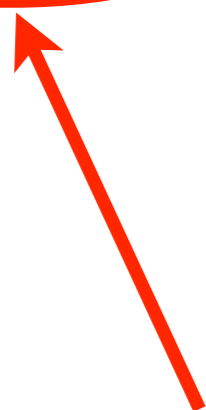
- **Pros:**
 - Assets only fetched on the first page load
 - Additional page loads load static assets directly from local cache!
- **Cons:**
 - Have to change URLs when assets change
 - Have to rely on intermediaries not corrupting your assets
 - CDNs do this sometimes.
 - Build in switches to let you forcefully un-cache things

- OK, but how do you make it work?

http://s3-media4.ak.yelpcdn.com/assets/2/www/css/d65342206dc8/www-pkg-en_US.css

http://s3-media4.ak.yelpcdn.com/assets/2/www/css/d65342206dc8/www-pkg-en_US.css

Subdomain shard



http://s3-media4.ak.yelpcdn.com/assets/2/www/css/d65342206dc8/www-pkg-en_US.css

CDN-backed domain



http://s3-media4.ak.yelpcdn.com/assets/2/www/css/d65342206dc8/www-pkg-en_US.css

Asset URL



`/assets/2/www/css/d65342206dc8/www-pkg-en_US.css`

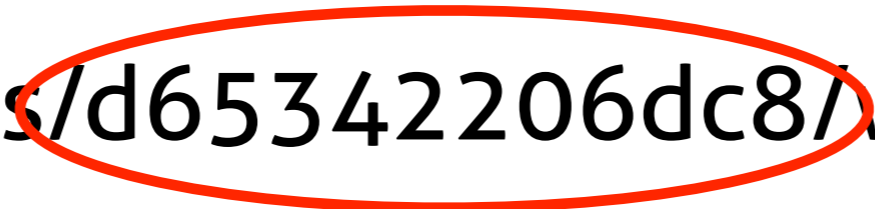
/assets/2/www/css/d65342206dc8/www-pkg-en_US.css

cache dumping mechanism



/assets/2/www/css/d65342206dc8/www-pkg-en_US.css

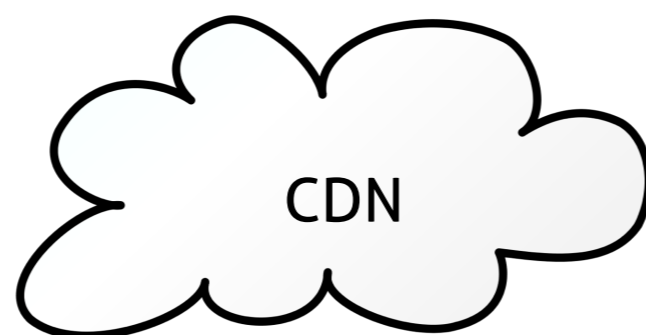
Asset hash



- **hashing functions**
 - mathematical functions which are 'one-way'
 - each input produces "unique" output
 - input cannot be recovered from output
 - often used to digest files into unique fingerprints
 - examples: **md5, SHA1**

```
fhats at yelp-fhats in ~  
$ echo 'Go Bears!' | md5  
3c54d7ab7822a8fc65a1f941bd8bae73  
fhats at yelp-fhats in ~  
$ echo 'Go Beers!' | md5  
8902c09e17d2f3ebca8d51b91920a575
```

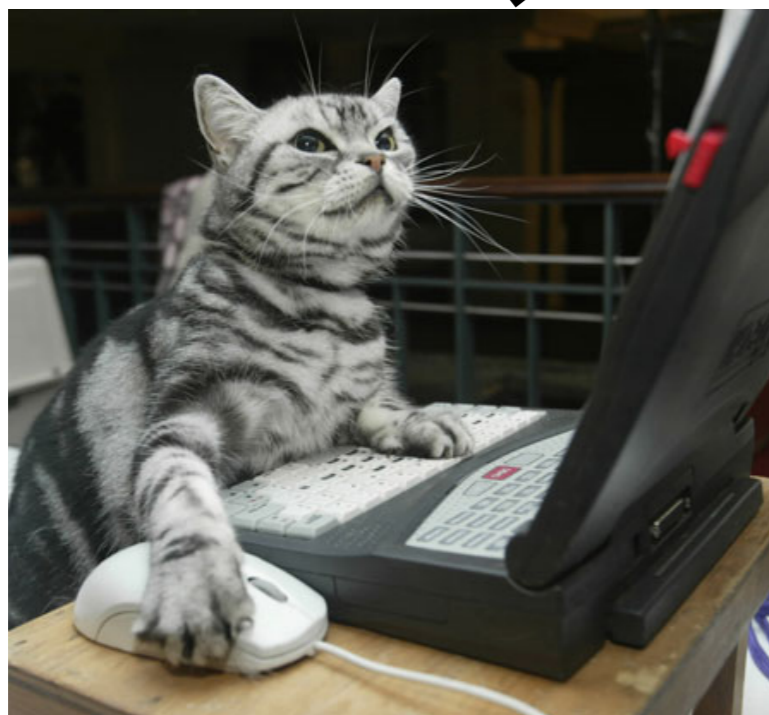
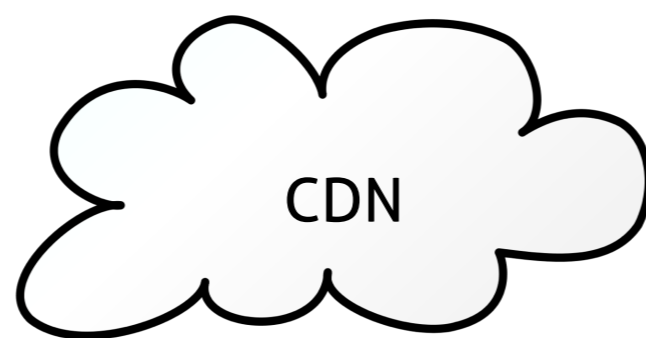
 latest: `/assets/css/3c54d7ab/yelp.css`



 latest: `/assets/css/3c54d7ab/yelp.css`

tell me about

`/assets/css/3c54d7ab/yelp.css`

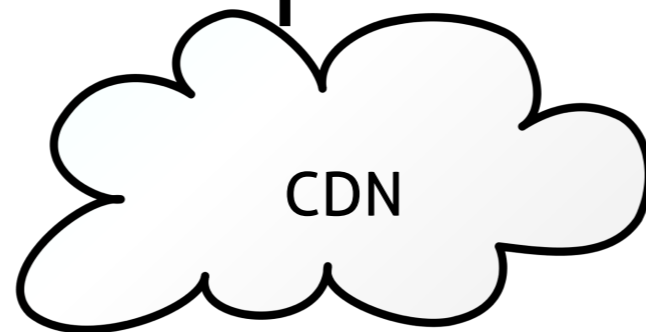


tell me about

`/assets/css/3c54d7ab/yelp.css`



latest: `/assets/css/3c54d7ab/yelp.css`



tell me about

`/assets/css/3c54d7ab/yelp.css`

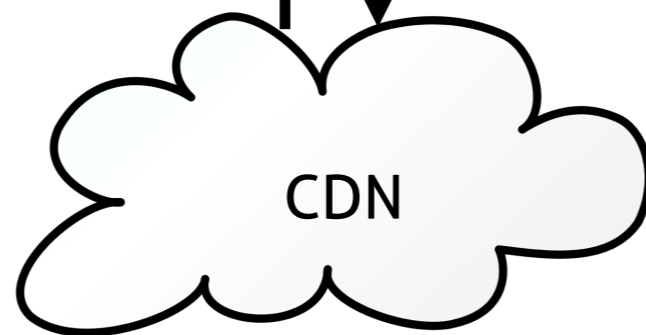


tell me about

`/assets/css/3c54d7ab/yelp.css`



latest: `/assets/css/3c54d7ab/yelp.css`



tell me about

`/assets/css/3c54d7ab/yelp.css`

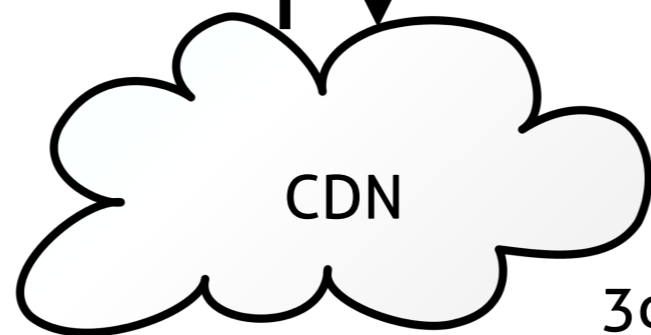


tell me about

`/assets/css/3c54d7ab/yelp.css`



latest: `/assets/css/3c54d7ab/yelp.css`



3c54d7ab

tell me about

`/assets/css/3c54d7ab/yelp.css`

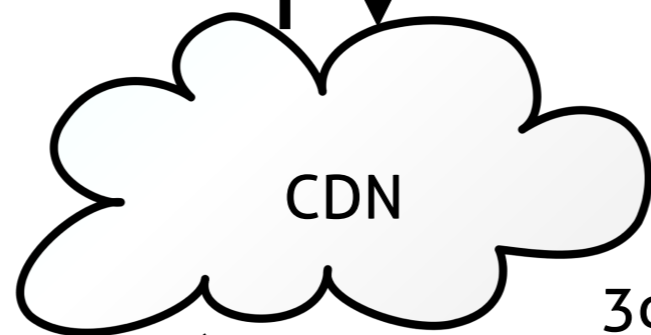


tell me about

`/assets/css/3c54d7ab/yelp.css`



latest: `/assets/css/3c54d7ab/yelp.css`



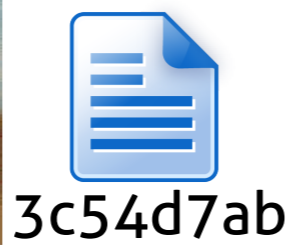
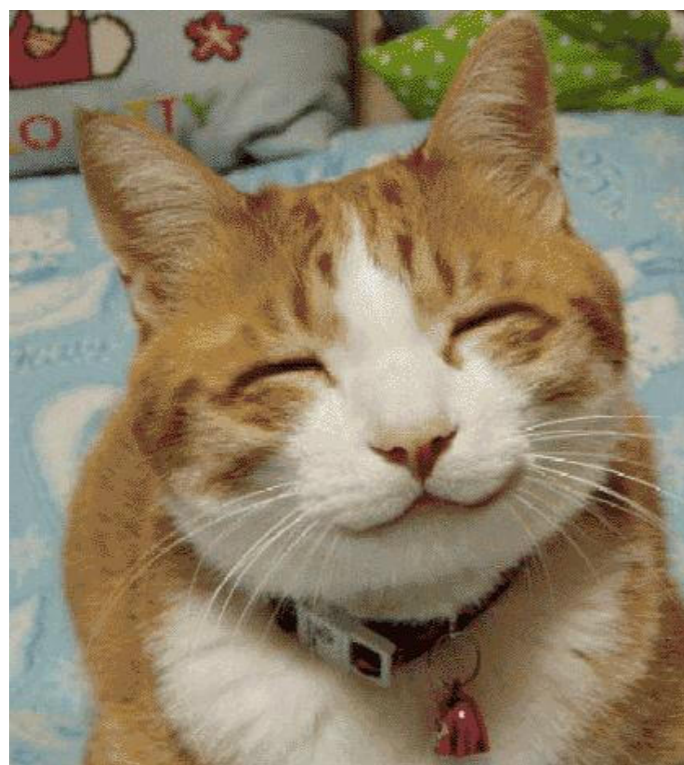
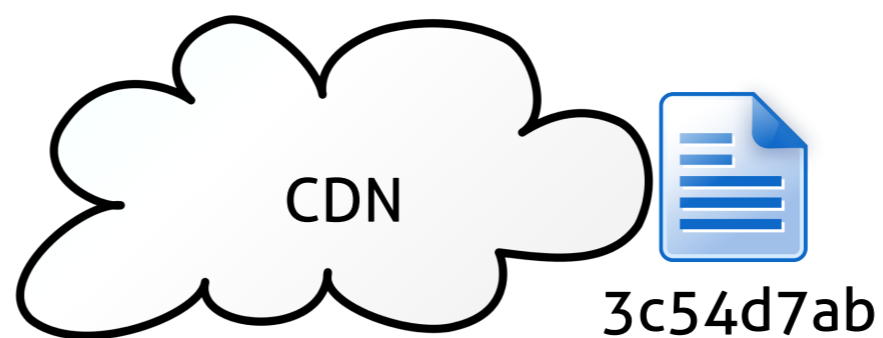
3c54d7ab

tell me about

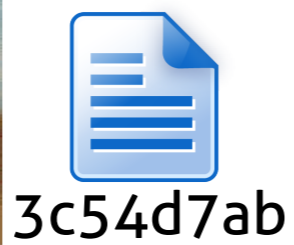
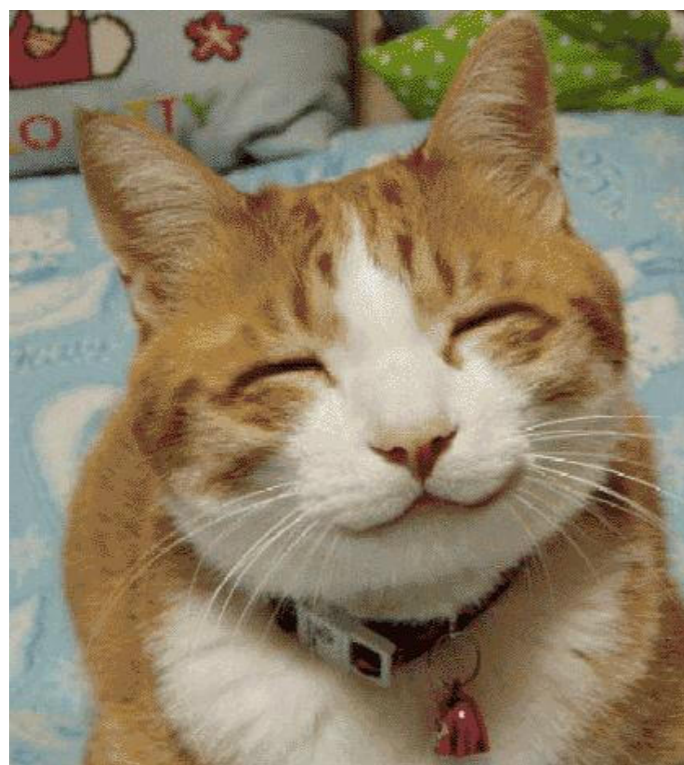
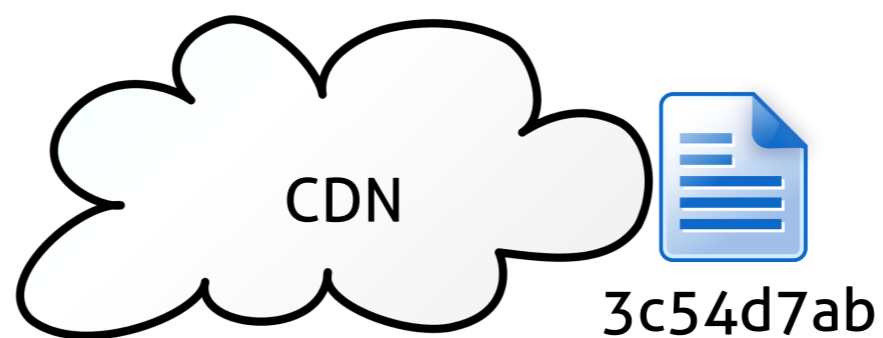
`/assets/css/3c54d7ab/yelp.css`



 latest: /assets/css/3c54d7ab/yelp.css

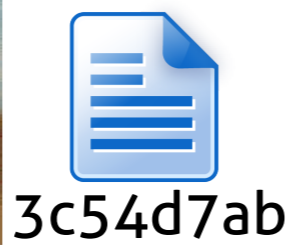
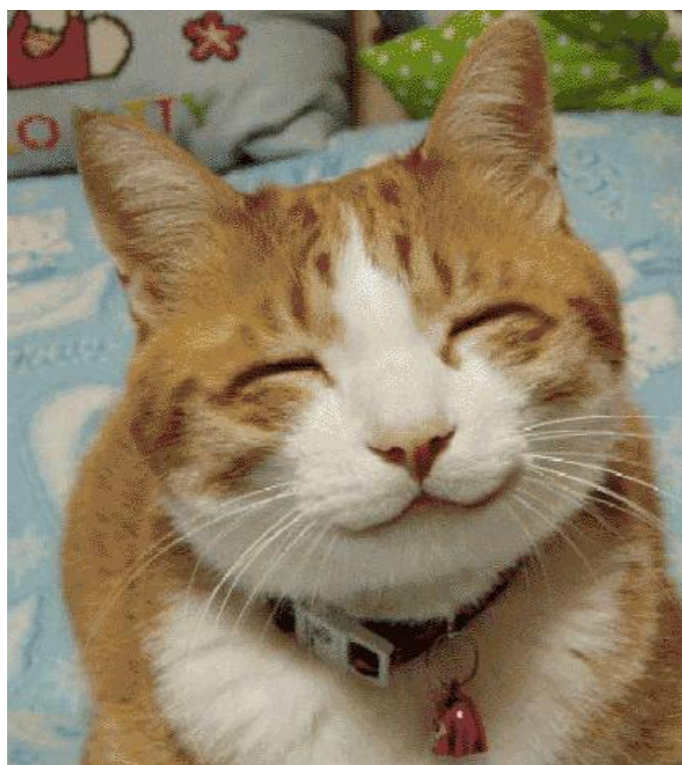
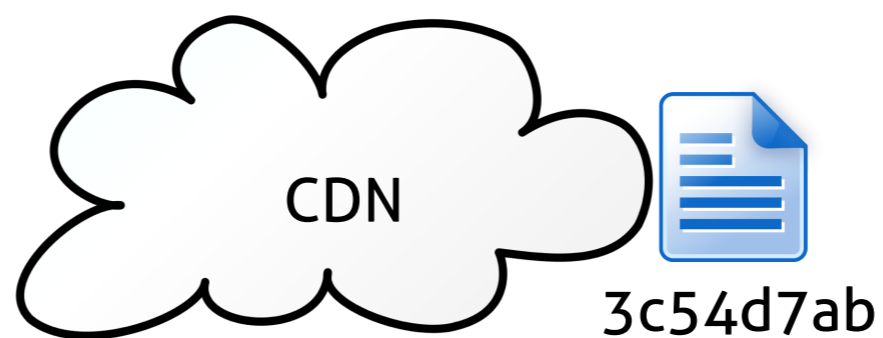


 latest: /assets/css/3c54d7ab/yelp.css

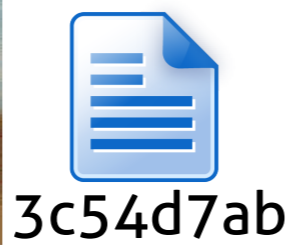
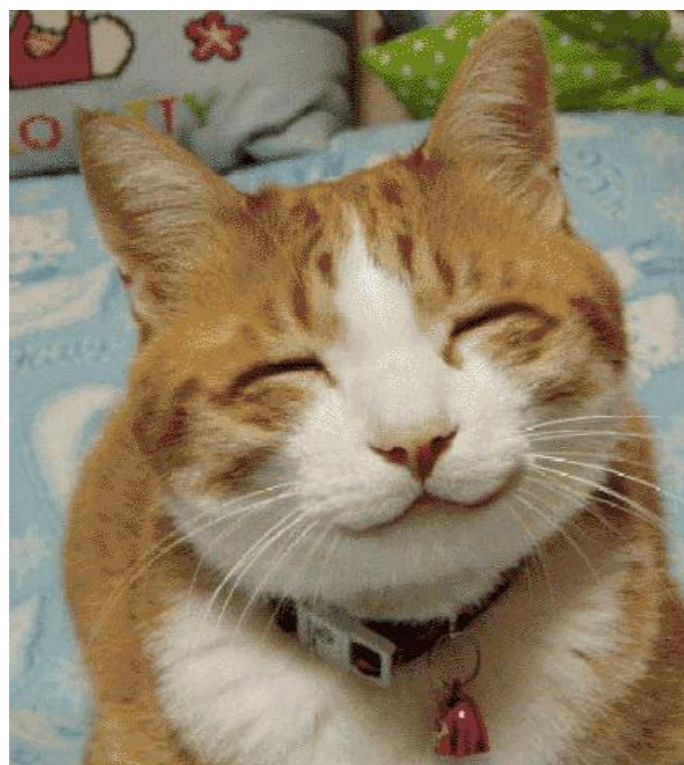
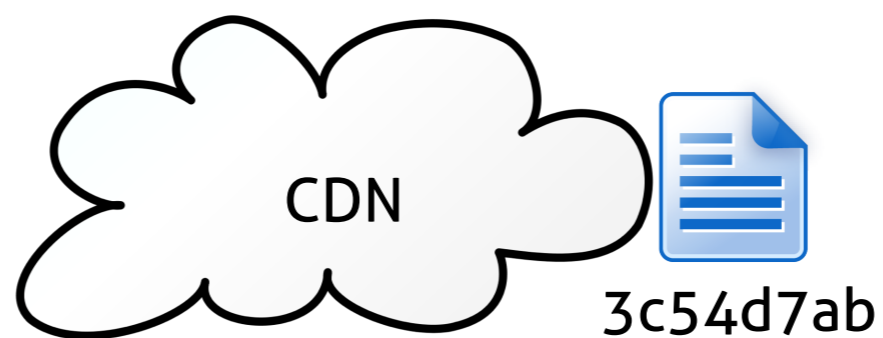


(time passes)

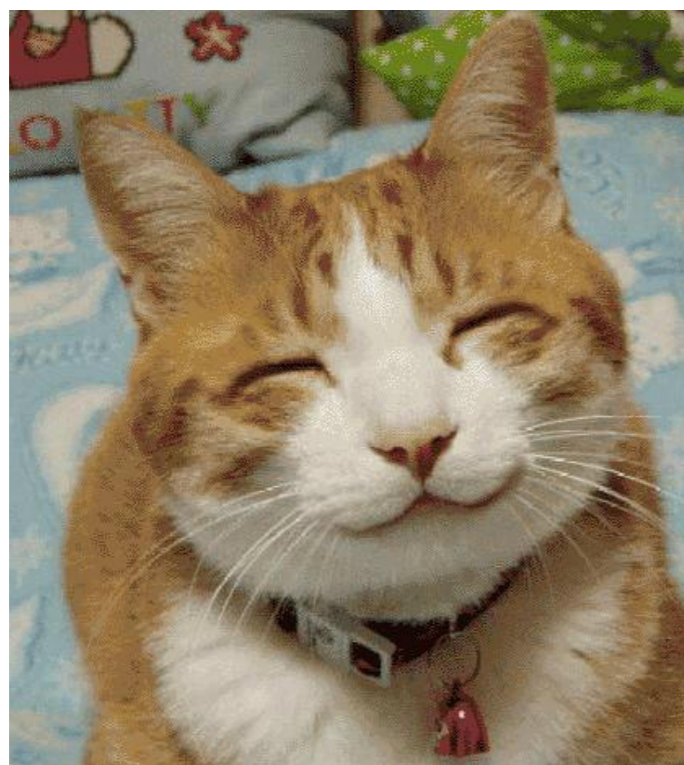
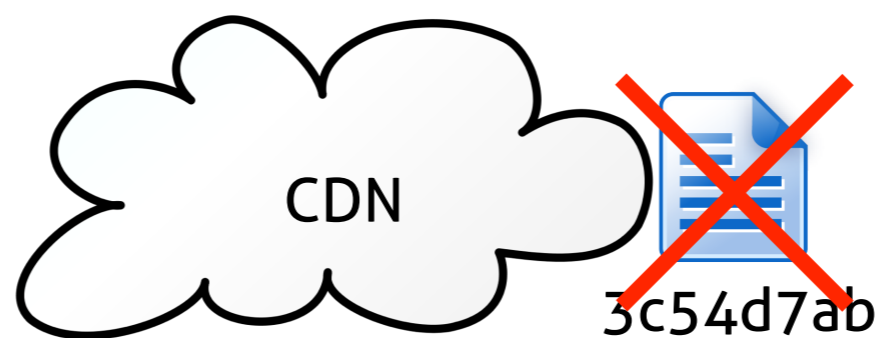
 latest: /assets/css/3c54d7ab/yelp.css



yelp  latest: /assets/css/**8902c09e**/yelp.css



yelp  latest: /assets/css/**8902c09e**/yelp.css



Playing Nicely with Browsers

- We want our site to load quickly
- We **also** want our site to *feel* responsive
 - To do this, we need to make sure our web page *performs well*
 - The performance of our web page and javascript is dependent on how much work we make the browser do
 - We can also give the browser hints to help it render more efficiently

- Standard web page:
 - HTML
 - CSS
 - Javascript
- CSS and Javascript usually loaded from external source (CDN)

- What's a DOM?
 - Document Object Model
 - represents the browser's internal understanding of the page elements
 - DOM typically refers to the elements on a page *before* they have stylesheets applied to them

- **Important: Don't block DOM parsing**
 - Downloading CSS does **not** block DOM parsing, so put your CSS references as early as possible
 - This usually means your `<link>` tag will be towards the front of your `<head>` section
 - Downloading Javascript **does** block DOM parsing
 - Put Javascript references as late in your HTML as possible
 - Typically this means that your `<script>` tags should go at the end of your `<body>` section

- Repaints
 - Repaint - browser re-calculates positions and styles of elements on a page
 - Also called reflows
 - Can be partial or full
 - Repaints are expensive and slow
 - Frequently a cause of sluggish-feeling UIs

- **Minimize repaints:**
 - Put stylesheets at the top of the page
 - Minimize unnecessary DOM changes
 - Summarize/batch CSS changes using javascript
 - Batch DOM changes by using document fragments

- Don't block on non-essential assets
 - Use AJAX to load non-assets at a later point in the page's lifecycle
 - That way pages 'feel' more responsive, and you can defer loading more content to the background

- **AJAX**
 - **Asynchronous Javascript And XML**
 - ...though XML is rarely used with AJAX any more...
 - Allows Javascript to download other assets later
 - Useful for deferring content that isn't needed immediately
 - Made easier by Javascript libraries like jQuery, Prototype, and AngularJS

- Image sizing
 - **Don't resize images in HTML!!**
 - This usually means that:
 - your images are improperly sized, meaning you're doing more network work than necessary
 - you're going to cause the browser a lot of pain as it re-sizes images on the fly

- Cookie-less Domains
 - Cookies are important
 - They help track user sessions
 - They store related data across requests
 - Cookies can also get really big
 - my GMail cookie is almost 3KB
 - Cookies get sent with **every request** to the domain they were issued from
 - an extra 3KB per request? that's already 2 packets worth of overhead!

- **Cookie-less Domains**
 - Your static assets don't need to know about cookies
 - Short, small requests to e.g. your CDN need to minimize overhead
 - Use a **separate domain** for your static assets in order to prevent cookies from being sent
 - e.g. `yelpcdn.com`

Monitoring Front-End Performance

- How do you monitor front-end performance?
 - Back-end is easy -- just log request processing start and stop, plus any other checkpoints
 - Front-end is not easy to track without being the user yourself

- Why bother monitoring front-end performance?
 - Back-end delivery times are only part of the story
 - Front-end problems could go unnoticed
 - Front-end performance matters a lot to user satisfaction

- **Solution: Navigation Timing API & AJAX**
 - **Navigation Timing API**
 - **Exposes various browser events:**
 - DNS lookup
 - connection start and end
 - redirection events
 - DOM loading/unloading/parsing events
 - browser request processing events
 - **Other APIs:**
 - Reporting about browser capabilities
 - Testing of external services
 - e.g. CDNs

- Use AJAX to send resulting data home
 - Batch up observations to reduce round-trips
 - Be sure to report frequently, as it might be a while before some events happen

Troubleshooting

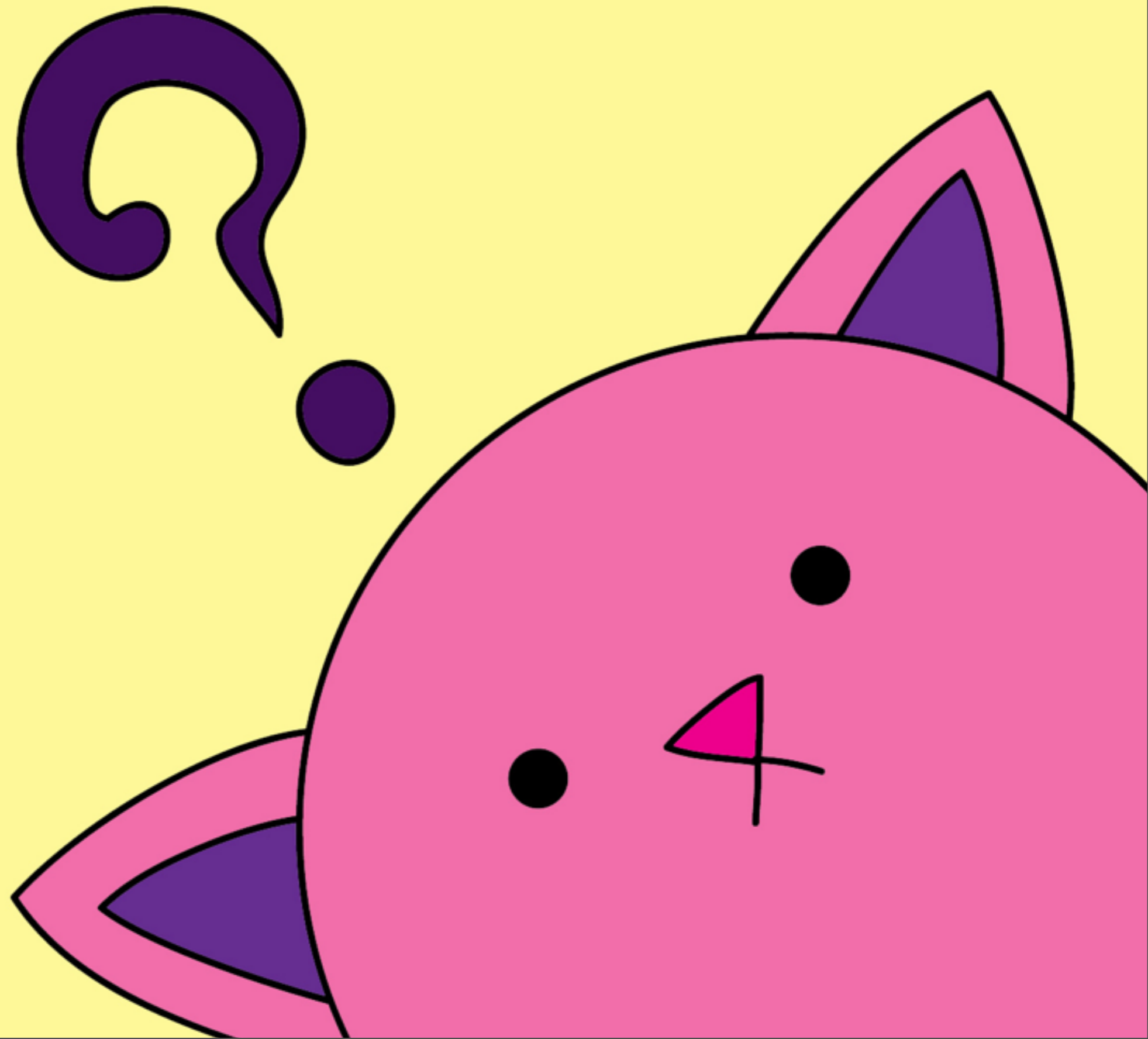
- Modern browsers have come a long way
- Chrome, Firefox, and Opera all have built-in developer tools
 - These tools are invaluable when figuring out what the browser is actually doing with your page

- Typical features:
 - Profiler
 - Heap inspector
 - Javascript debugger
 - Network inspector
 - DOM inspector

- Demo

- What's the difference between Yelp's homepage compressed and uncompressed?
 - Use **curl** - downloads items from internet
 - **man curl** to get help
 - **-v**: provide verbose output
 - **-H**: specify HTTP headers
 - **>**: redirect command output to file

Questions?



Friday, October 11, 13

fhats@yelp.com

@fredhatfull

- <http://ricerca.mat.uniroma3.it/users/lombardi/cloud.png>
- <http://clipartist.info/RSS/openclipart.org/2011/August/13-Saturday/server-999px.png>
- <http://robertianhawdon.me.uk/blog/wp-content/uploads/2013/08/haproxy-logo-150x78.png>
- http://incubator.apache.org/triplesoup/images/apache_feather.png
- <http://www.linuxbrigade.com/wp-content/uploads/2013/06/mysql.png>
- https://lh3.ggpht.com/7O3H3V0fEBumwJlqDLD03t1fmwl8fH9YoBsPwB2UO_aiBilM7OAOe2gkFB3wrojJqbM=w300
- <http://www.iclarified.com/images/news/33493/138309/138309-1280.png>
- http://upload.wikimedia.org/wikipedia/en/1/10/Internet_Explorer_7_Logo.png
- <https://lh6.ggpht.com/xR8CnvODNJorg76Y0JUWpKppG4TNZf10n8SM6EssuBmaf2L4wfkxpV4umLBziEkjg=w300>
- <http://www.technobuffalo.com/wp-content/uploads/2013/06/yelp-ios-app-update.png>
- <http://blackberryrocks.com/wp-content/uploads/2011/08/yelp-blackberry-app-version-2-0.png>
- http://i.i.cbsi.com/cnwk.1d/i/tim2/2013/07/10/enhanced%2B2_270x450.png
- <http://somefun.net/fun/wp-content/uploads/2013/01/Cat-rar.jpg>
- http://gentlemint.com/media/images/2012/02/14/bfb2303f.jpg.505x650_q85.jpg
- <http://bedadi.com/wp-content/uploads/2012/11/cat-on-computer.jpg>
- [http://www-03.ibm.com/software/lotus/symphony/gallery.nsf/GalleryClipArtAll/8CE0CE18AB15F52185257596003225D4/\\$File/Icon-Document03-Blue.png](http://www-03.ibm.com/software/lotus/symphony/gallery.nsf/GalleryClipArtAll/8CE0CE18AB15F52185257596003225D4/$File/Icon-Document03-Blue.png)
- <http://robervations.ca/wp-content/uploads/2011/02/happy-cat1.jpg>
- http://images2.wikia.nocookie.net/_cb20121101220042/camphalfbloodroleplay/images/thumb/d/d4/HappyCat.jpeg/1280px-HappyCat.jpeg
- <http://1.bp.blogspot.com/-kRnZimNNJsA/UBlEl68mn0I/AAAAAAAAARns/yCBKphe6nG4/s1600/funny-cat-pictures-009-001.jpg>
- http://3.bp.blogspot.com/-J6WK7HAE_78/T1lvmpl8FAI/AAAAAAAAACSI/xrcJT7el-Tk/s1600/cute+cat+funny.jpg
- <http://zst.utp.edu.pl/Technologie/TCP/3Fig.%202.%20Slow-start%20with.jpg>
- <http://christianconcertalerts.com/wp-content/uploads/2012/12/Pittsburgh-Moving-Movers-at-Work.jpeg>
- <https://www.macgamut.com/uploads/content/images/cat.jpg>