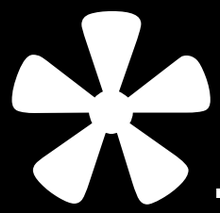
What HTTP/2.0 will*
do ~~for~~ TO you

Mark Nottingham
http://www.mnot.net/
or, @mnot



forward-looking statement

TECHNICAL

A projected ~~financial~~ statement based on management expectations. A forward-looking statement involves risks with regard to the accuracy of assumptions underlying the projections.

Discussions of these statements typically include words such as *estimate*, *anticipate*, *project*, and *believe*.

1. Some recent history

IETF HTTPbis WG

WHAT? to clarify RFC2616 and improve interop

WHO? Roy Fielding, Julian Reschke

WHO (2)? Apache, Mozilla, Chrome, IIS, Varnish, Squid, F5, Curl, ATS, IE, HAProxy...

WHEN: Almost finished!

WHEN (2): ... after **FIVE YEARS :(**

IETF HTTPbis WG (2)

STRICTLY chartered to avoid making a new version of the protocol.

Because EVERYBODY knows that's not going to happen.

MEANWHILE

November 2009: Mike Belshe and Roberto Peon announce SPDY

March 2011: Mike talks about SPDY to the HTTPbis WG at IETF80

~April 2011: Chrome, Google start using SPDY

March 2012: HTTPbis solicits proposals for new protocol work

March 2012: Firefox 11 ships with SPDY (off by default)

May 2012: Netcraft finds 339 servers that support SPDY

June 2012: Nginx announces SPDY implementation

July 2012: Akamai announces SPDY implementation

Tuesday: HTTPbis re-chartered to work on HTTP/2.0, based on SPDY

2. What is it?

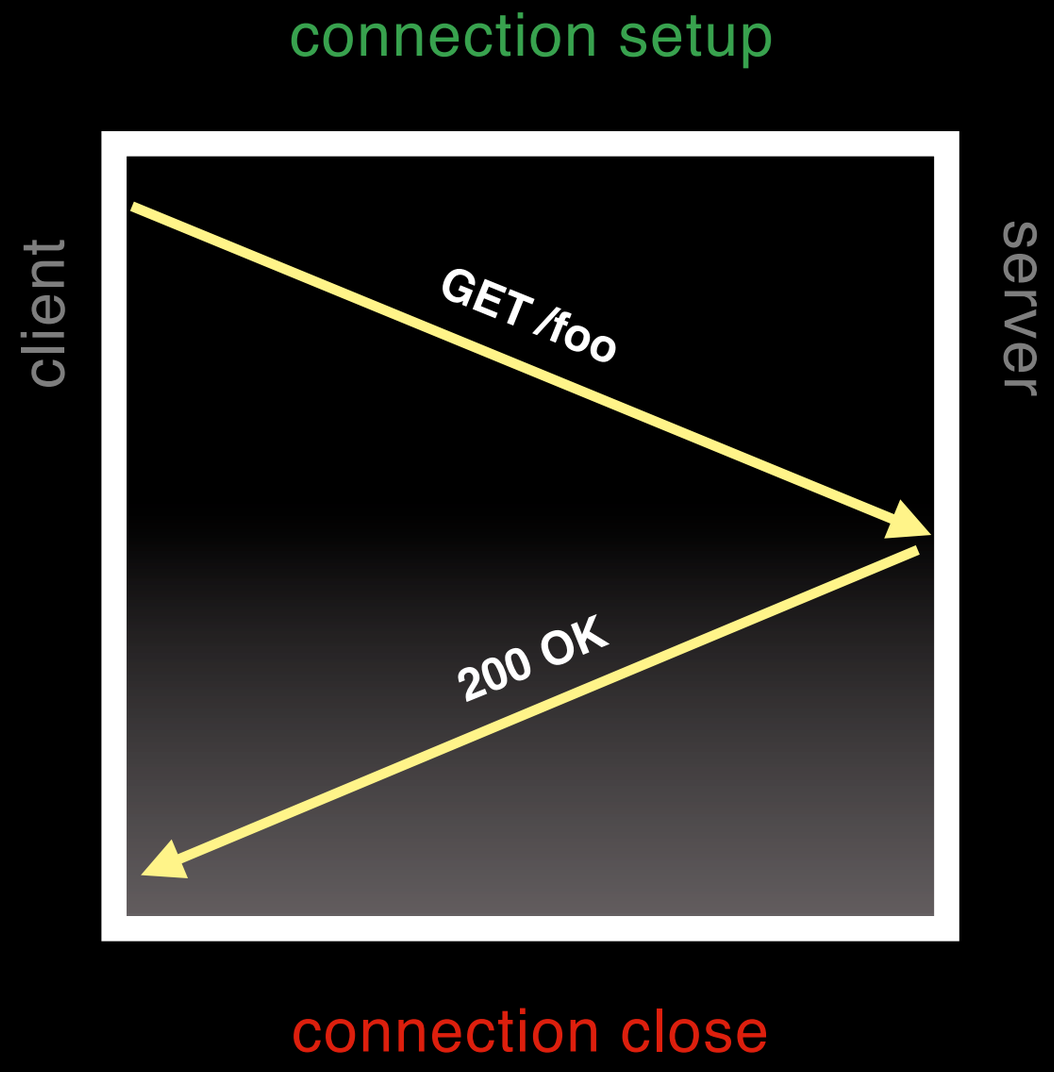
NO change to HTTP semantics; it's about how it gets onto the wire

Nothing new to see here

Not magic

1. multiplexing
2. header compression
3. server push (?)
4. TLS (?)

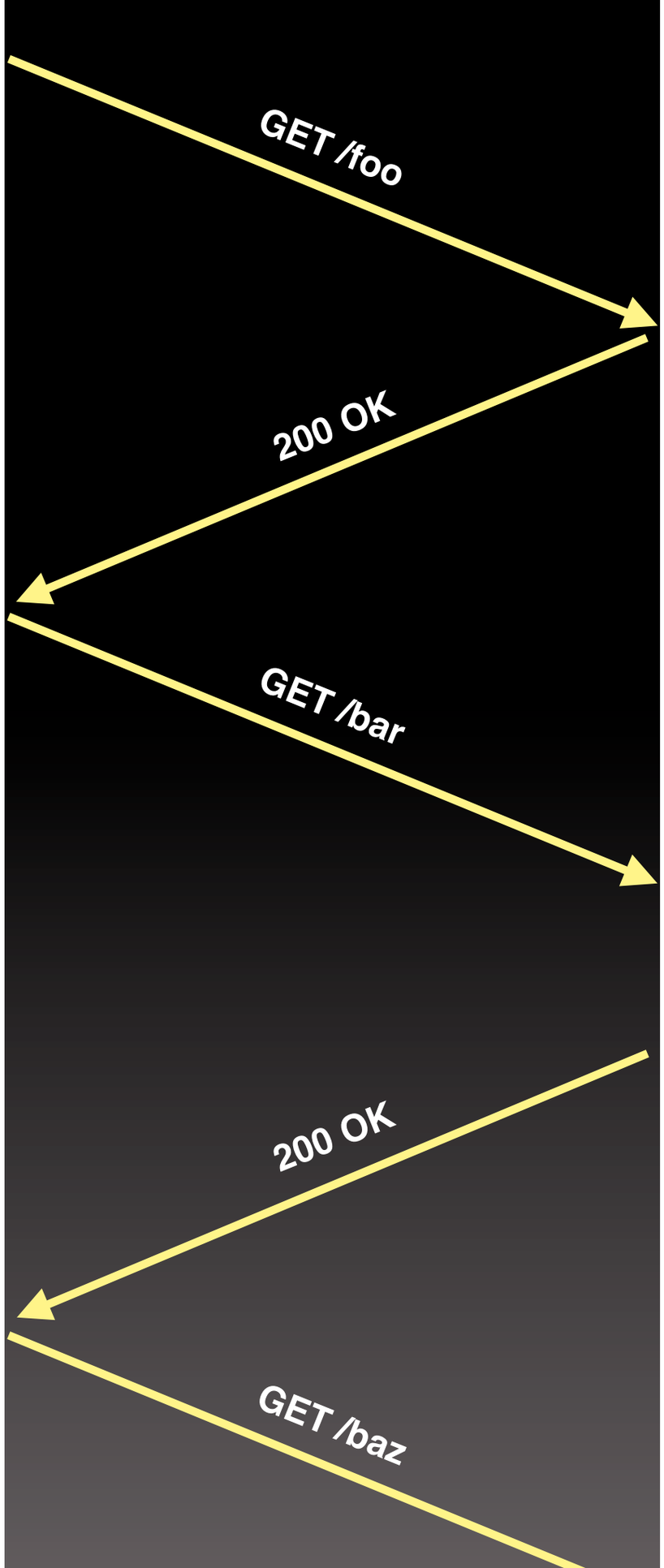
2.1 multiplexing



vanilla HTTP/1.0

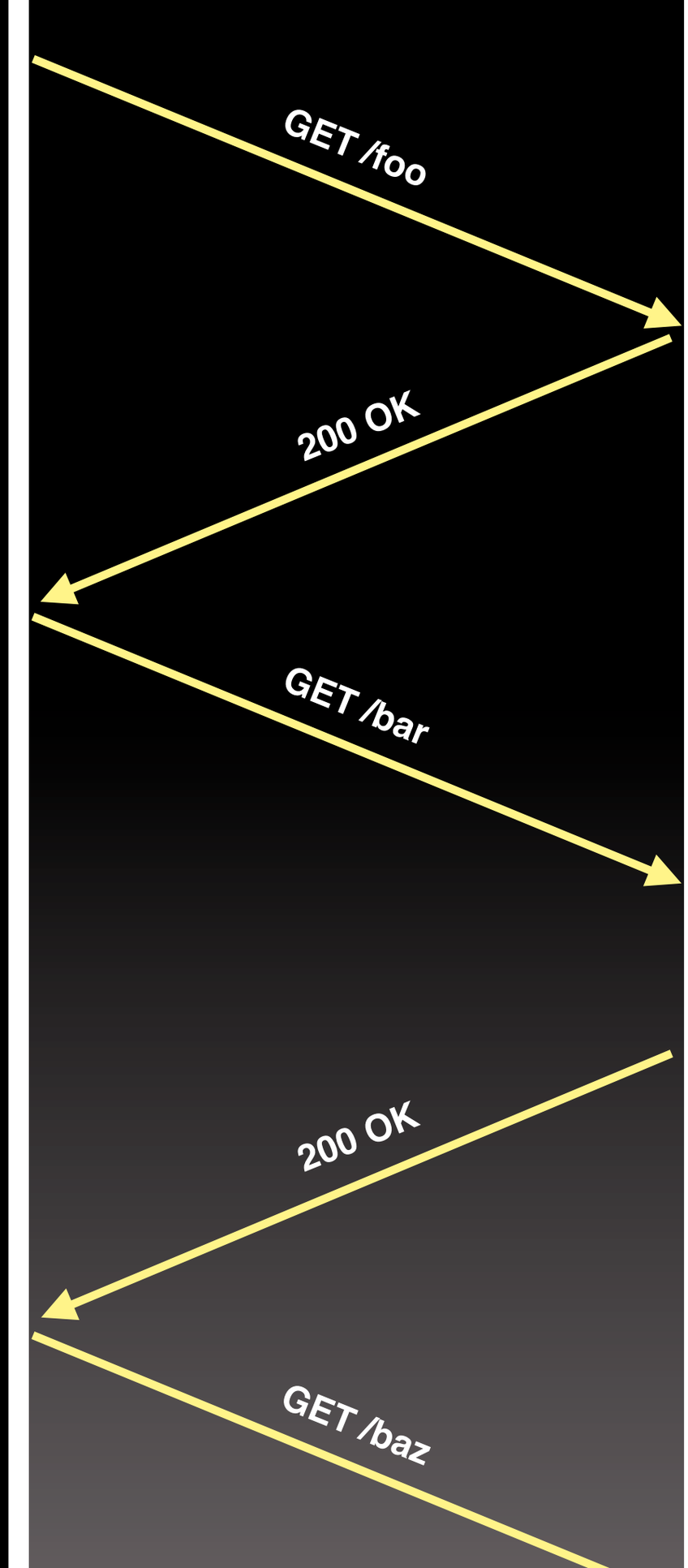
One request per TCP connection

OMG ITS SO SLOW



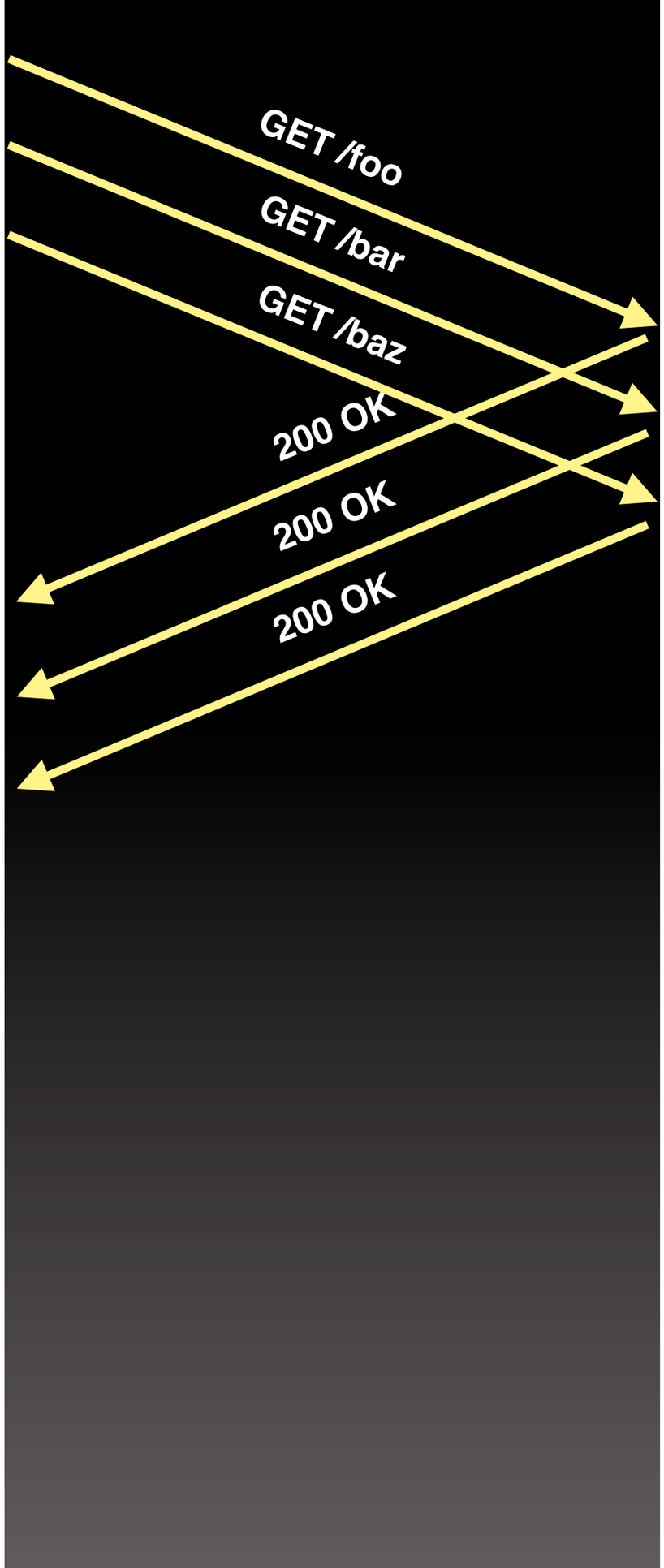
HTTP/1.0
(with Keep-Alive)

able to reuse connections, avoid connection setup



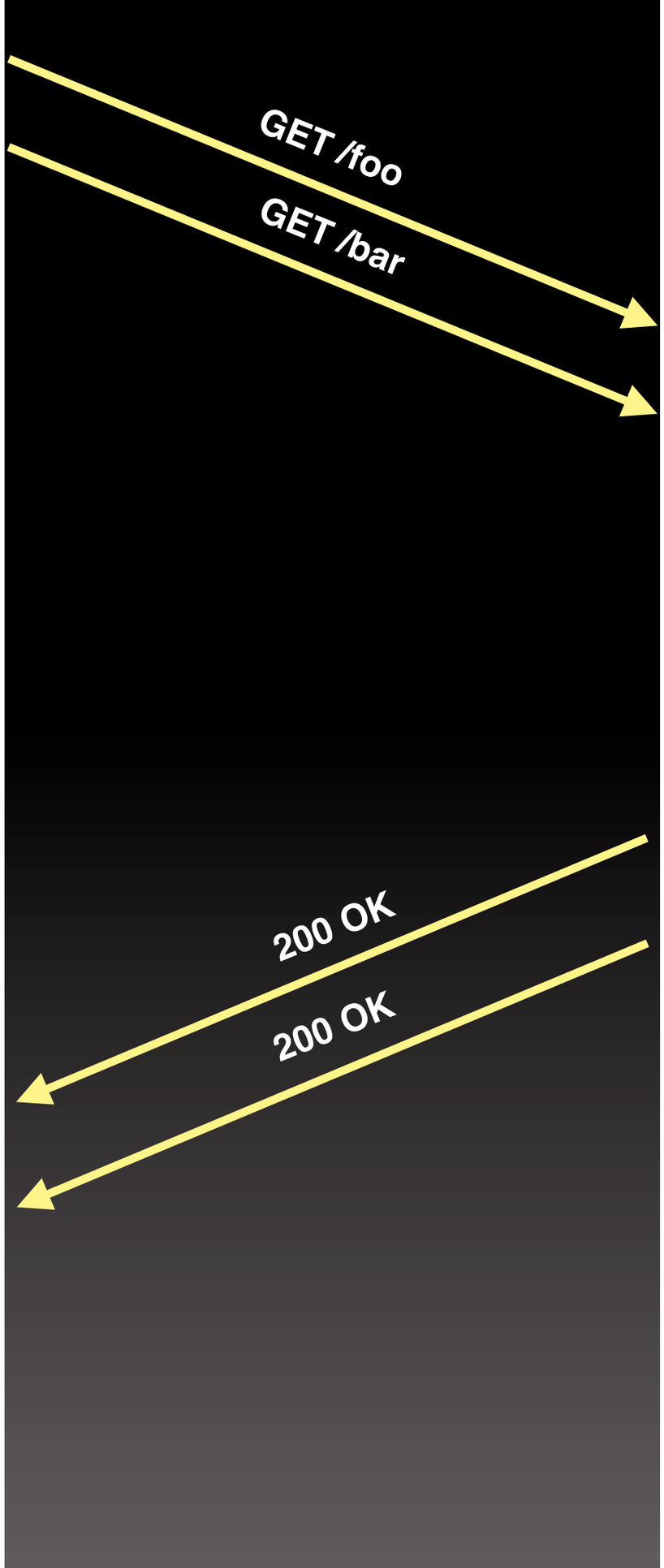
BUT
it still blocks

one outstanding
request at a time



HTTP/1.1
(with pipelining)

multiple, ordered
requests



head-of-line blocking

Still serialised!

Large downloads / long “think” time can block other requests

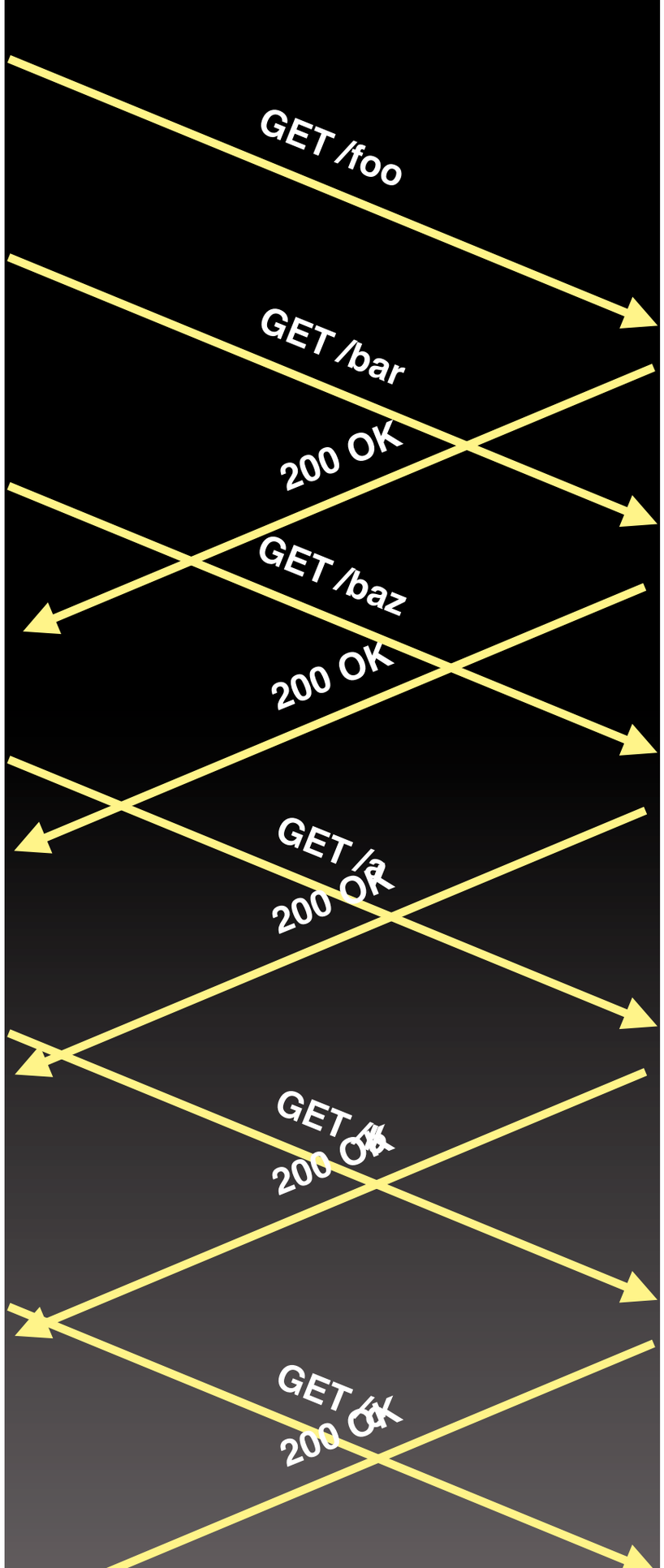
The State of the Art

- **Use persistent connections (“keep-alives”)**
- **Pipelining is getting a little deployment**
- **Use multiple connections for parallelism**
  - RFC2616 said “2”; HTTPbis says “reasonable”
  - Browsers use 4-8; bad people use more
- **Build lots of heuristics into browsers for connection reuse**
- **Hope that it all works out**

What's so bad about that?

- **TCP is built for long-lived flows**
  - More connections = shorter flows
    - Congestion control doesn't have time to ramp up
    - Makes Buffer Bloat worse
- **Fairness (user to user, app to app)**
- **How many connections is the best?**

cascading waterfalls



SPDY multiplexing

- one connection
- many requests
- prioritisation
- out of order
- interleaved

NO* QUEUING

2.2 header compression

```
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
(KHTML, like Gecko) Version/6.0.1 Safari/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_LCLVpU%26_kid%3D1%26_ver%3D1%26_mac
%3DlVnlM3hMdb3Cs3hqMVuk_dQEixsqQzUlNYCs9H_Kj8c.;
user_prefs=1&2596706699&q0tPzMlJLaoEAA==
Connection: keep-alive
```

525 bytes

```
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
(KHTML, like Gecko) Version/6.0.1 Safari/536.26.14
Accept: */*
DNT: 1
Referer: http://www.etsy.com/
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Cookie: autosuggest_split=1;
etala=111461200.1476767743.1349274889.1349274889.1349274889.1.0;
etalb=111461200.1.10.1349274889; last_browse_page=%2F; uaid=uaid%3DVdhk5W6sexG-
_Y7ZBeQFa3cq7yMQ%26_now%3D1325204464%26_slt%3Ds_LCLVpU%26_kid%3D1%26_ver%3D1%26_mac
%3DlVnlM3hMdb3Cs3hqMVuk_dQEixsqQzUlNYCs9H_Kj8c.;
user_prefs=1&2596706699&q0tPzMlJLaoEAA==
Connection: keep-alive
```

226 new bytes; 690 total

```
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
(KHTML, like Gecko) Version/6.0.1 Safari/536.26.14
Accept: */*
DNT: 1
Referer: http://www.etsy.com/
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Cookie: autosuggest_split=1;
etala=111461200.1476767743.1349274889.1349274889.1349274889.1.0;
etalb=111461200.1.10.1349274889; last_browse_page=%2F; uaid=uaid%3DVdhk5W6sexG-
_Y7ZBeQFa3cq7yMQ%26_now%3D1325204464%26_slt%3Ds_LCLVpU%26_kid%3D1%26_ver%3D1%26_mac
%3DlVnlM3hMdb3Cs3hqMVuk_dQEixsqQzUlNYCs9H_Kj8c.;
user_prefs=1&2596706699&q0tPzMlJLaoEAA==
Connection: keep-alive
```

14 new bytes; 683 total

```
GET /assets/dist/js/bootstrap/username-suggester.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
(KHTML, like Gecko) Version/6.0.1 Safari/536.26.14
Accept: */*
DNT: 1
Referer: http://www.etsy.com/
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Cookie: autosuggest_split=1;
etala=111461200.1476767743.1349274889.1349274889.1349274889.1.0;
etalb=111461200.1.10.1349274889; last_browse_page=%2F; uaid=uaid%3DVdhk5W6sexG-
_Y7ZBeQFa3cq7yMQ%26_now%3D1325204464%26_slt%3Ds_LCLVpU%26_kid%3D1%26_ver%3D1%26_mac
%3DlVnlM3hMdb3Cs3hqMVuk_dQEixsqQzUlNYCs9H_Kj8c.;
user_prefs=1&2596706699&q0tPzMlJLaoEAA==
Connection: keep-alive
```

28 new bytes; 698 total

- **Four requests**
- **2,596 bytes total**
- **Minimum three packets in most places**
  - One for the HTML, two+ for assets
- **1,797 redundant bytes**

HTTP headers on a connection are highly similar

Request URI

User-Agent

Cookies

Referer

Big req * many reqs / small IW = SLOW

- **Patrick's test:**
  - 83 asset requests
  - IW = 3
  - ~1400 bytes of headers
- **Uncompressed: 7-8RT**
- **Compressed (zlib): 1RT**

2.3 server push (?)

2.4 TLS?

3. What does it all **mean**?

HTTP/2.0 is going to change Web Engineering...

... but not change **HTTP APIs**. Much.

Leaky Abstractions

~~Reduce Requests~~

- Image spriting - not necessary
- CSS / JS can be in multiple files
- HTTP APIs can be finer-grained without sacrificing performance
- Third party content is an even bigger problem

~~Domain Sharding~~

- **Multiplexing SHOULD make multiple connections unnecessary**
- **Key is to get the TCP connection "warm" as quickly as possible, and keep it there**

~~Header Optimisation~~

- **Now, adding new headers is easy**
  - Very small performance / latency / bandwidth impact
- **What should be in headers can be in headers**
- **Content Negotiation might become interesting again**

Predictability

Better Error Handling

Debugging will need Tools

Lots of tweaking

- Prioritisation
- When to Push
- Flow Control

Transition Decisions

- **De-sharding**
- **De-combining scripts**
- **De-spriting**
- **etc.**

Guess what, Steve?

4. What's still wrong

4.1 Security is hard

4.2 TCP is awkward

- **In-order delivery = head-of-line blocking**
- **Initial congestion window is small**
- **Packet loss isn’t handled well**

HTTP as the “Everything Protocol”?

Some Links

http://bit.ly/httpbis-home

http://www.chromium.org/spdy

http://www.mnot.net/