TCP, HTTP and SPDY

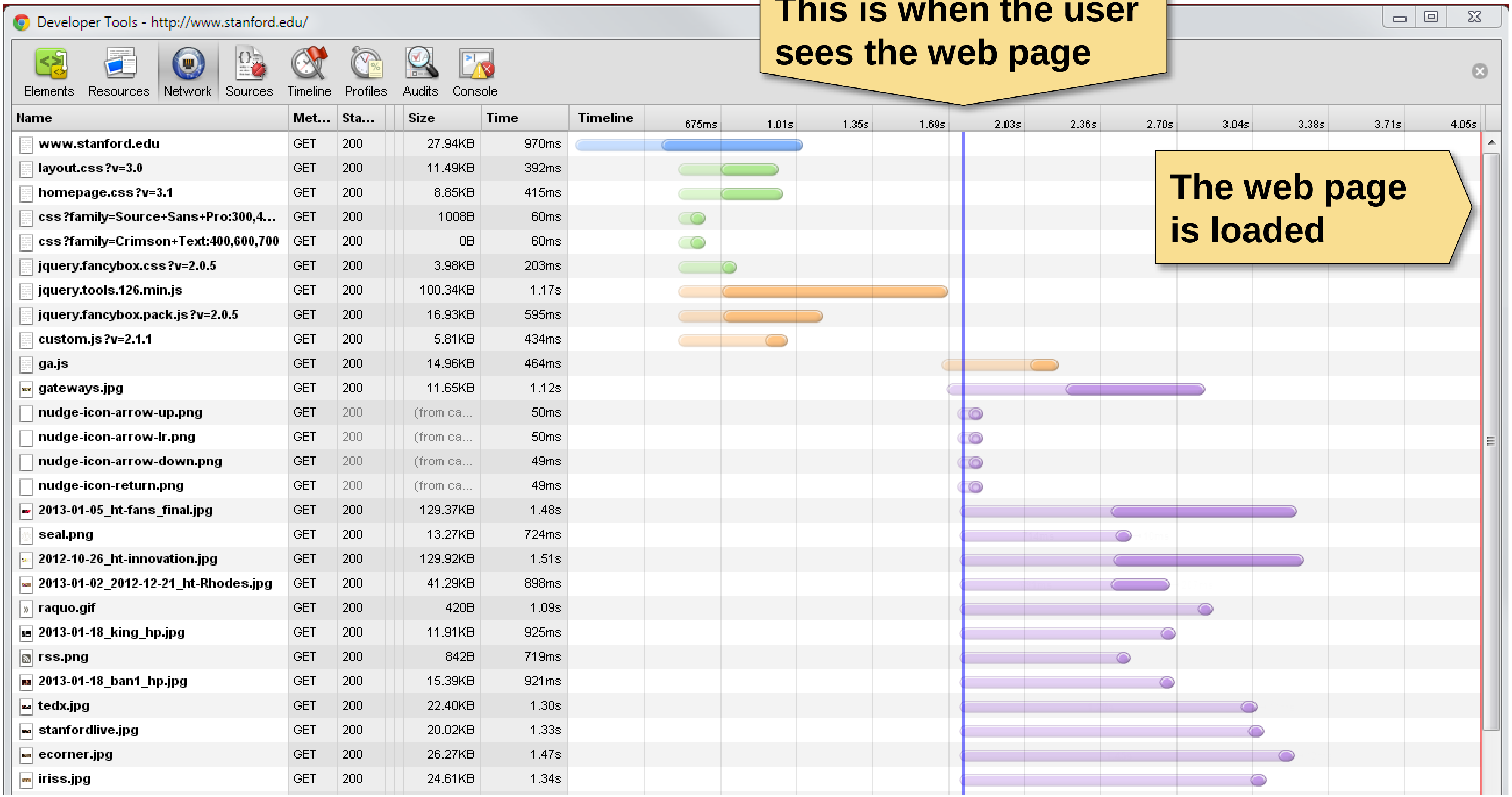
Ivan Pepelnjak (ip@ipSpace.net)

ipSpace.net AG

The Problem

This is when the user sees the web page

The web page is loaded



Developer Tools - http://www.stanford.edu/

Elements Resources Network Sources Timeline Profiles Audits Console

Name	Met...	Sta...	Size	Time
www.stanford.edu	GET	200	27.94KB	970ms
layout.css?v=3.0	GET	200	11.49KB	392ms
homepage.css?v=3.1	GET	200	8.85KB	415ms
css?family=Source+Sans+Pro:300,4...	GET	200	1008B	60ms
css?family=Crimson+Text:400,600,700	GET	200	0B	60ms
jquery.fancybox.css?v=2.0.5	GET	200	3.98KB	203ms
jquery.tools.126.min.js	GET	200	100.34KB	1.17s
jquery.fancybox.pack.js?v=2.0.5	GET	200	16.93KB	595ms
custom.js?v=2.1.1	GET	200	5.81KB	434ms
ga.js	GET	200	14.96KB	464ms
gateways.jpg	GET	200	11.65KB	1.12s
nudge-icon-arrow-up.png	GET	200	(from ca...	50ms
nudge-icon-arrow-lr.png	GET	200	(from ca...	50ms
nudge-icon-arrow-down.png	GET	200	(from ca...	49ms
nudge-icon-return.png	GET	200	(from ca...	49ms
2013-01-05_ht-fans_final.jpg	GET	200	129.37KB	1.48s
seal.png	GET	200	13.27KB	724ms
2012-10-26_ht-innovation.jpg	GET	200	129.92KB	1.51s
2013-01-02_2012-12-21_ht-Rhodes.jpg	GET	200	41.29KB	898ms
raquo.gif	GET	200	420B	1.09s
2013-01-18_king_hp.jpg	GET	200	11.91KB	925ms
rss.png	GET	200	842B	719ms
2013-01-18_ban1_hp.jpg	GET	200	15.39KB	921ms
tedx.jpg	GET	200	22.40KB	1.30s
stanfordlive.jpg	GET	200	20.02KB	1.33s
ecorner.jpg	GET	200	26.27KB	1.47s
iriss.jpg	GET	200	24.61KB	1.34s

Timeline: 675ms 1.01s 1.35s 1.69s 2.03s 2.36s 2.70s 3.04s 3.38s 3.71s 4.05s

What’s going on?

Why Is This a Problem?

Why are impatient and forgetful:

< 0.1sec	Instantaneous response (Nielsen, 1993)
1 sec	User’s flow of thoughts is interrupted
2 sec	Interference with short-term memory
10 sec	User is no longer focused on dialog

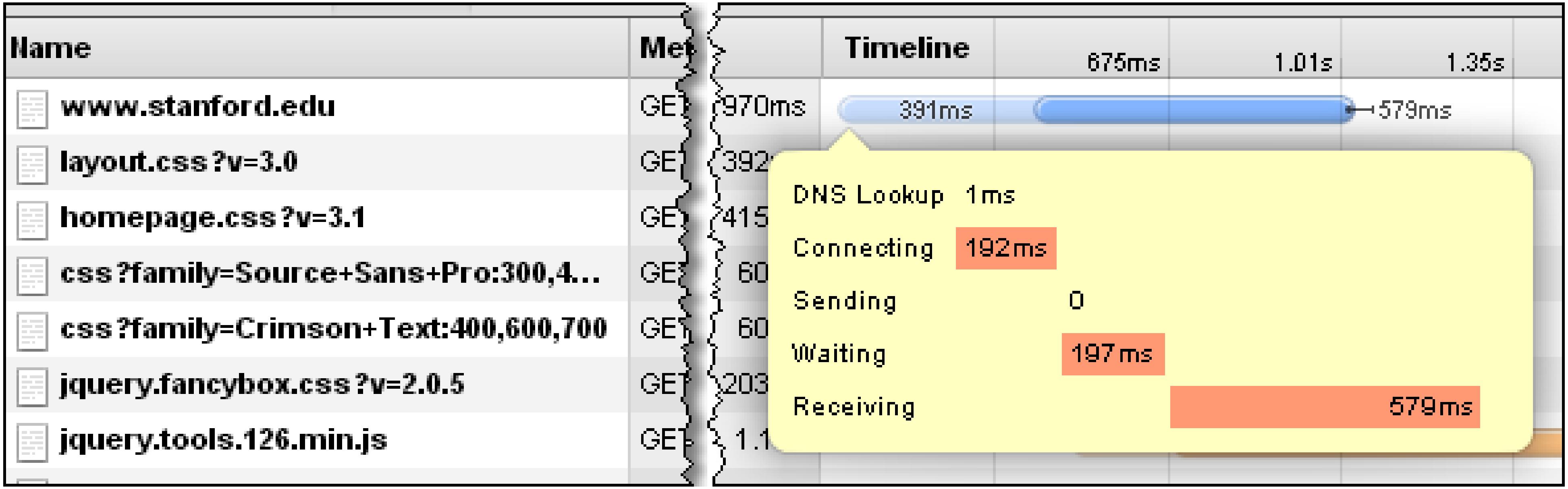
Some other numbers:

- Users abandon non-working web page in 3-4 seconds
- Half a second delay caused 20% drop in traffic (Google, 2006)
- Ultimate goal: 100 msec load time

Sources:

http://csi.ufs.ac.za/resres/files/Nah.pdf
http://www.strangeloopnetworks.com/web-performance-optimization-hub/topics/psychology-and-human-factors/
http://www.webperformancetoday.com/category/human-factors/
http://www.websiteoptimization.com/speed/tweak/psychology-web-performance/

The Problem – Details



- Most web pages have tens (or more) elements
- Every element is loaded with an HTTP request
- HTTP runs over TCP (HTTPS over TLS and TCP)

To understand web page loading behavior we have to understand TCP

Disclaimer

There's very little you can fix in TCP. Most optimization must be done in markup, back-end scripts and server configuration:

- Optimal markup with progressive enhancements;
- Image sprites and use of new CSS features instead of images
- Responsive images (load lo-res, replace with hi-res)
- DATA URI for small images
- Minimize cookies
- Prefetching
- Avoid redirects and DNS lookups
- Compression
- Browser-side local storage
- Caching, caching, caching

Sample article: http://queue.acm.org/detail.cfm?id=2434256

TCP And HTTP 101

TCP Mission Statement

TCP = Reliable stream delivery service

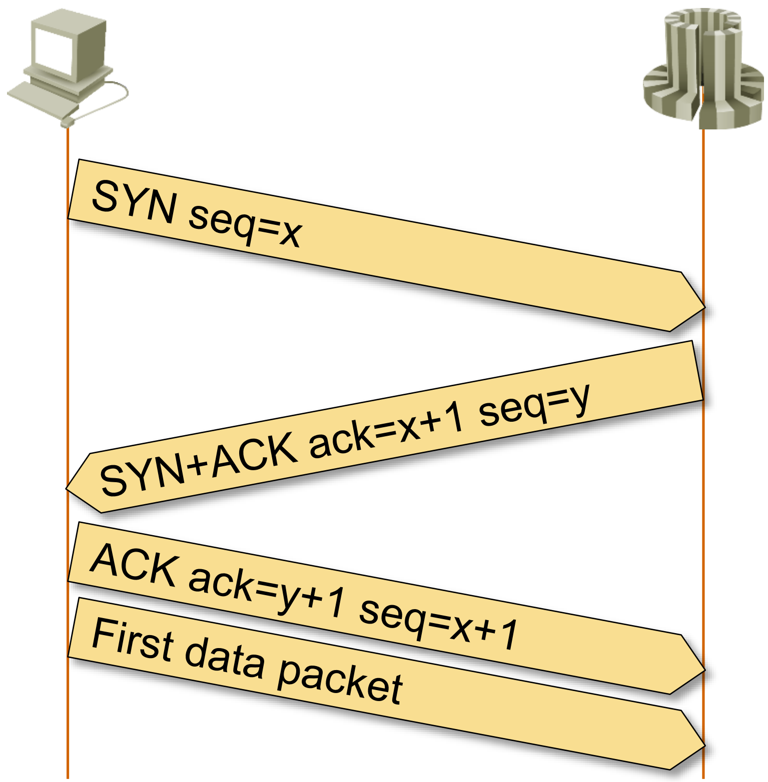
Handles:

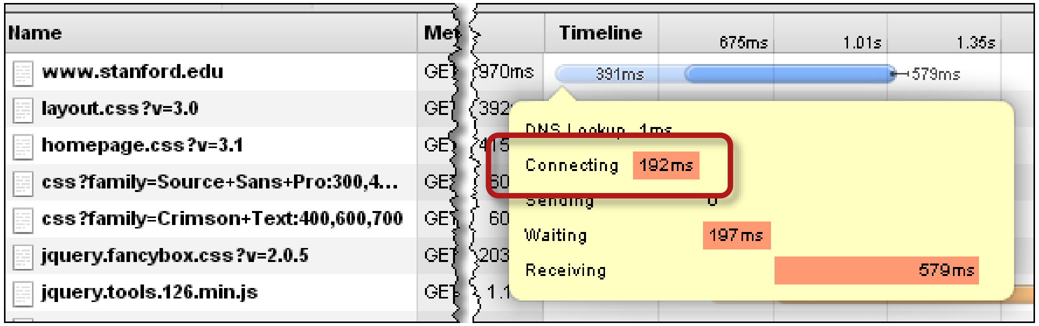
- Packet loss
- Packet duplication and reordering

Does not care about:

- Timely delivery
- Multiple sessions
- Structured data or record boundaries

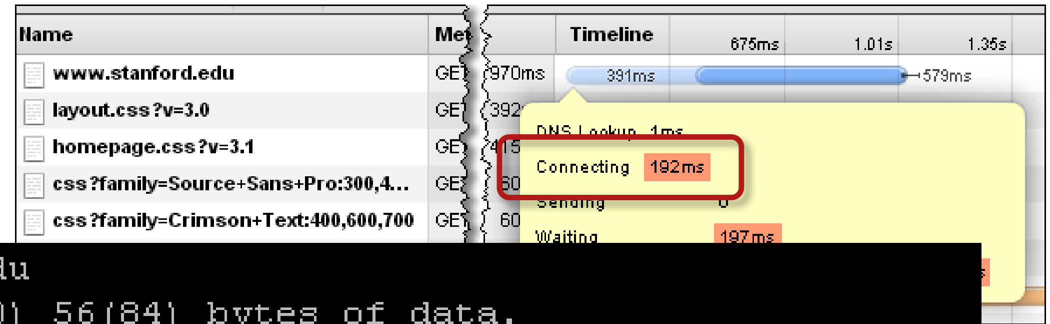
TCP Session Establishment: 3-Way Handshake





- TCP session established with a 3-way handshake
- RTT delay before first user data is sent

Is It Really That Far to Stanford?




```
[pipi@fedi ~]$ ping -c 3 www.stanford.edu
PING www-v6.stanford.edu (171.67.215.200) 56(84) bytes of data.
64 bytes from www-v6.Stanford.EDU (171.67.215.200): icmp_req=1 ttl=238 time=189 ms
64 bytes from www-v6.Stanford.EDU (171.67.215.200): icmp_req=2 ttl=238 time=189 ms
64 bytes from www-v6.Stanford.EDU (171.67.215.200): icmp_req=3 ttl=238 time=189 ms
```

```
[pipi@fedi ~]$ traceroute -q 1 www.stanford.edu
traceroute to www.stanford.edu (171.67.215.200), 30 hops max, 60 byte packets
 1  gw (192.168.200.193)  3.851 ms
 2  BSN-access.dsl.siol.net (213.250.19.90)  6.739 ms
 3  95.176.253.5 (95.176.253.5)  6.711 ms
 4  95.176.253.11 (95.176.253.11)  9.536 ms
 5  30gigabitethernet4-3.core1.fra1.he.net (80.81.192.172)  22.886 ms
 6  10gigabitethernet1-4.core1.par2.he.net (184.105.213.162)  32.934 ms
 7  10gigabitethernet7-1.core1.ash1.he.net (184.105.213.93)  110.724 ms
 8  10gigabitethernet7-4.core1.pao1.he.net (184.105.213.177)  181.123 ms
 9  stanford-university.10gigabitethernet1-4.core1.pao1.he.net (216.218.2
10  boundarya-rtr.Stanford.EDU (68.65.168.33)  190.420 ms
11  *
12  *
13  www-v6.Stanford.EDU (171.67.215.200)  192.288 ms
```

- 30 ms within Europe
- 80 ms across Atlantic
- 70 ms across US
- 10 ms on West Coast

The Difference Between Theory and Practice

```
[pipi@fedi ~]$ ping -c 3 www.stanford.edu
PING www-v6.stanford.edu (171.67.215.200) 56(84) bytes of data.
64 bytes from www-v6.Stanford.EDU (171.67.215.200): icmp_req=1 ttl=238 time=189 ms
64 bytes from www-v6.Stanford.EDU (171.67.215.200): icmp_req=2 ttl=238 time=189 ms
64 bytes from www-v6.Stanford.EDU (171.67.215.200): icmp_req=3 ttl=238 time=189 ms
```

Input interpretation:

Ljubljana, Osrednjeslovenska to Stanford, California, United States

Distance: Show non-metric units

9798 km (kilometers)

Direct travel times: More

aircraft (550 mph)	11 hours
sound	8 hours
light in fiber	45.8 ms (milliseconds)
light in vacuum	32.7 ms (milliseconds)

(assuming constant-speed great-circle path)

The Impact of Transmission Technology

```
$ ping www.nil.com

Pinging www.nil.com [192.168.253.10] with 32 bytes of data:
Reply from 192.168.253.10: bytes=32 time=8ms TTL=253
Reply from 192.168.253.10: bytes=32 time=8ms TTL=253
Reply from 192.168.253.10: bytes=32 time=8ms TTL=253
Reply from 192.168.253.10: bytes=32 time=9ms TTL=253
```

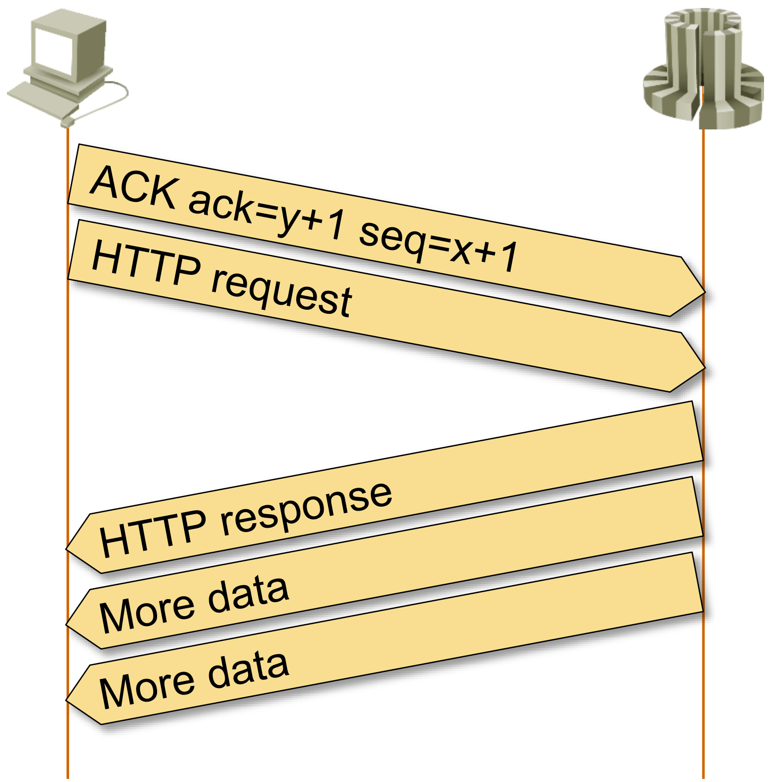
Fiber Internet access + VPN tunnel

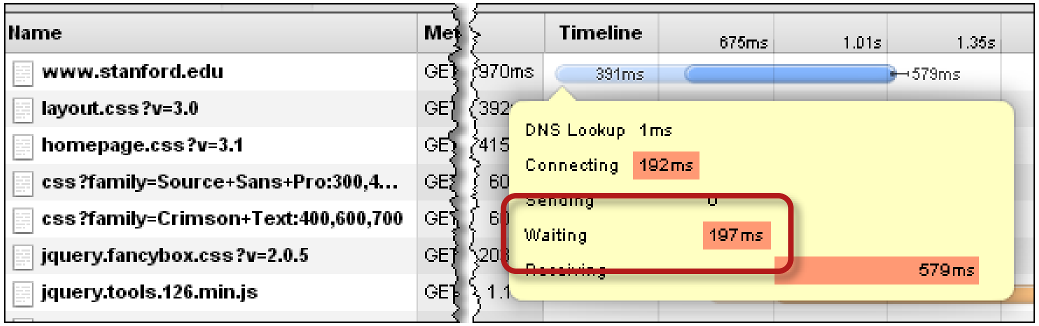
```
$ ping www.nil.com

Pinging www.nil.com [193.110.145.49] with 32 bytes of data:
Reply from 193.110.145.49: bytes=32 time=369ms TTL=244
Reply from 193.110.145.49: bytes=32 time=282ms TTL=244
Reply from 193.110.145.49: bytes=32 time=409ms TTL=244
Reply from 193.110.145.49: bytes=32 time=267ms TTL=244
Reply from 193.110.145.49: bytes=32 time=242ms TTL=244
Reply from 193.110.145.49: bytes=32 time=223ms TTL=244
Reply from 193.110.145.49: bytes=32 time=178ms TTL=244
Reply from 193.110.145.49: bytes=32 time=167ms TTL=244
Reply from 193.110.145.49: bytes=32 time=193ms TTL=244
Reply from 193.110.145.49: bytes=32 time=136ms TTL=244
Reply from 193.110.145.49: bytes=32 time=249ms TTL=244
Reply from 193.110.145.49: bytes=32 time=228ms TTL=244
Reply from 193.110.145.49: bytes=32 time=193ms TTL=244
Reply from 193.110.145.49: bytes=32 time=167ms TTL=244
```

3G mobile access over Bluetooth

Remember: Latency is never zero. It could be higher than expected

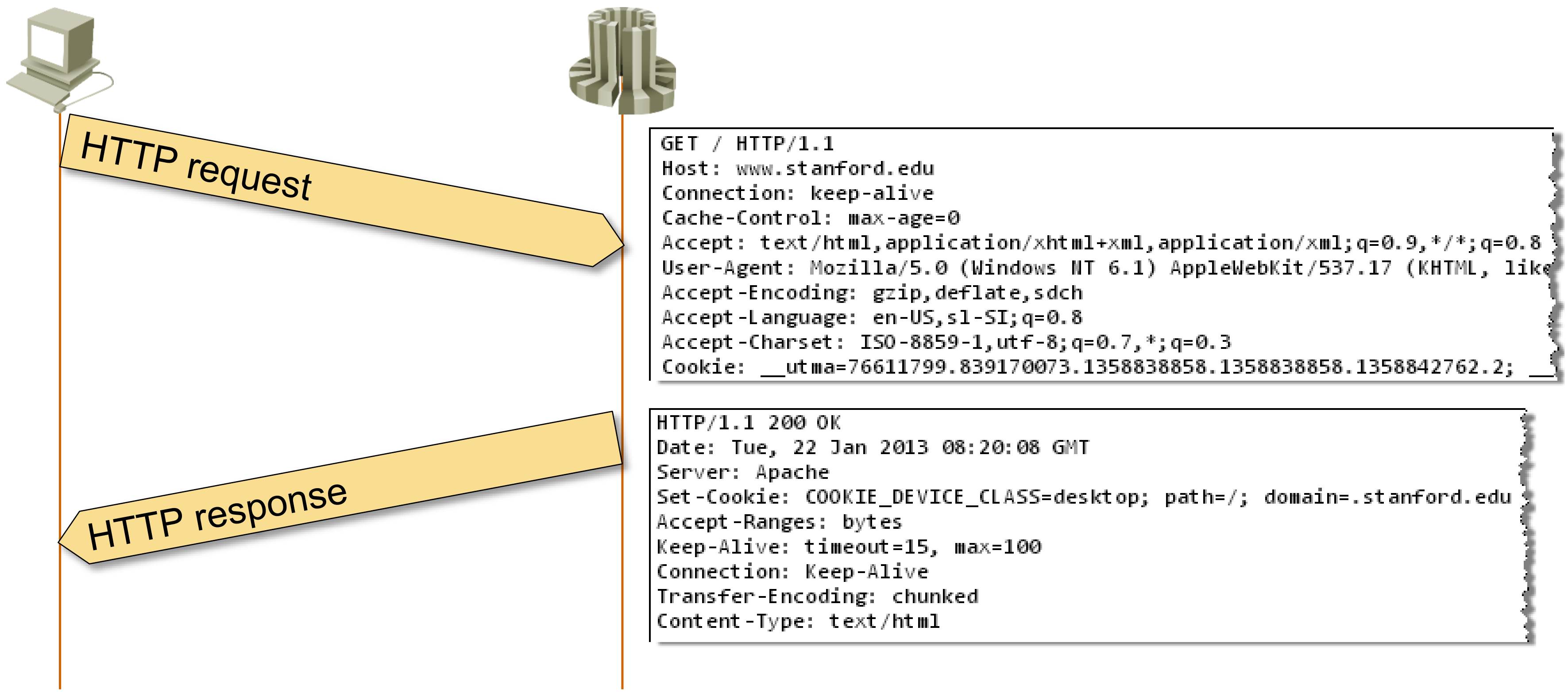
HTTP Request / Response





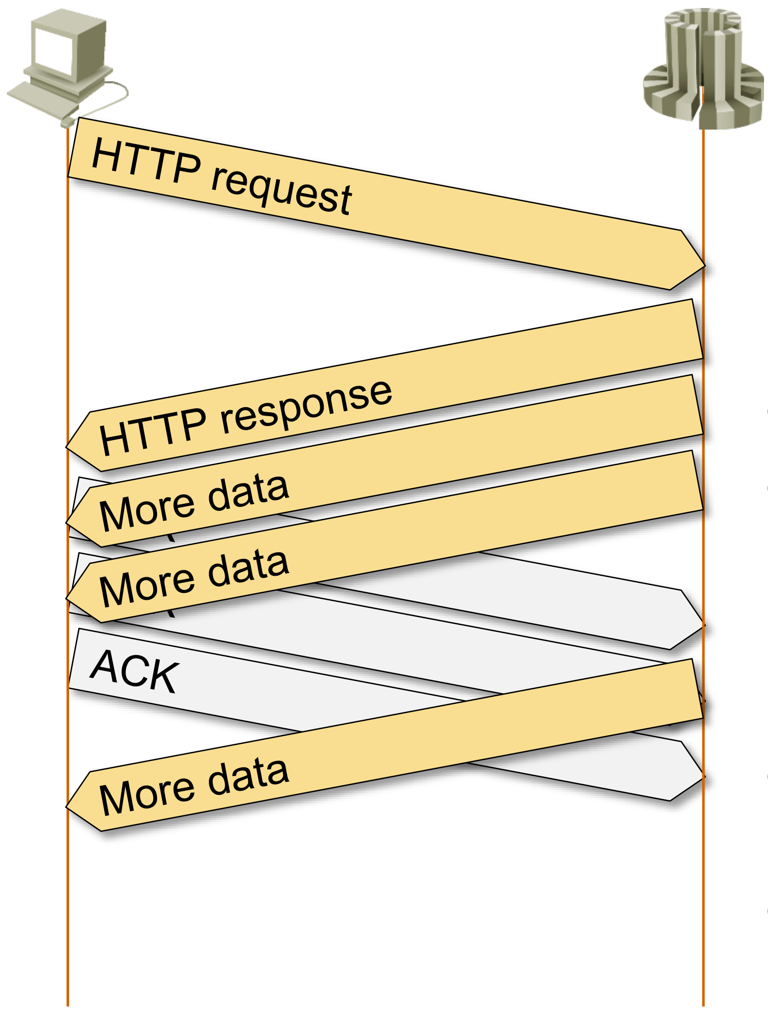
- HTTP is a request-response protocol
- Another RTT delay before first HTML data is received

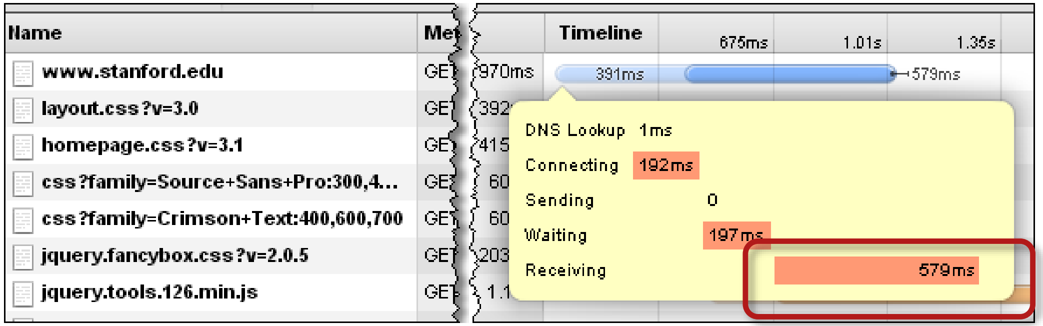
HTTP Request / Response



- HTTP is a request-response protocol
- Another RTT delay before first HTML data is received

TCP Initial Congestion Window



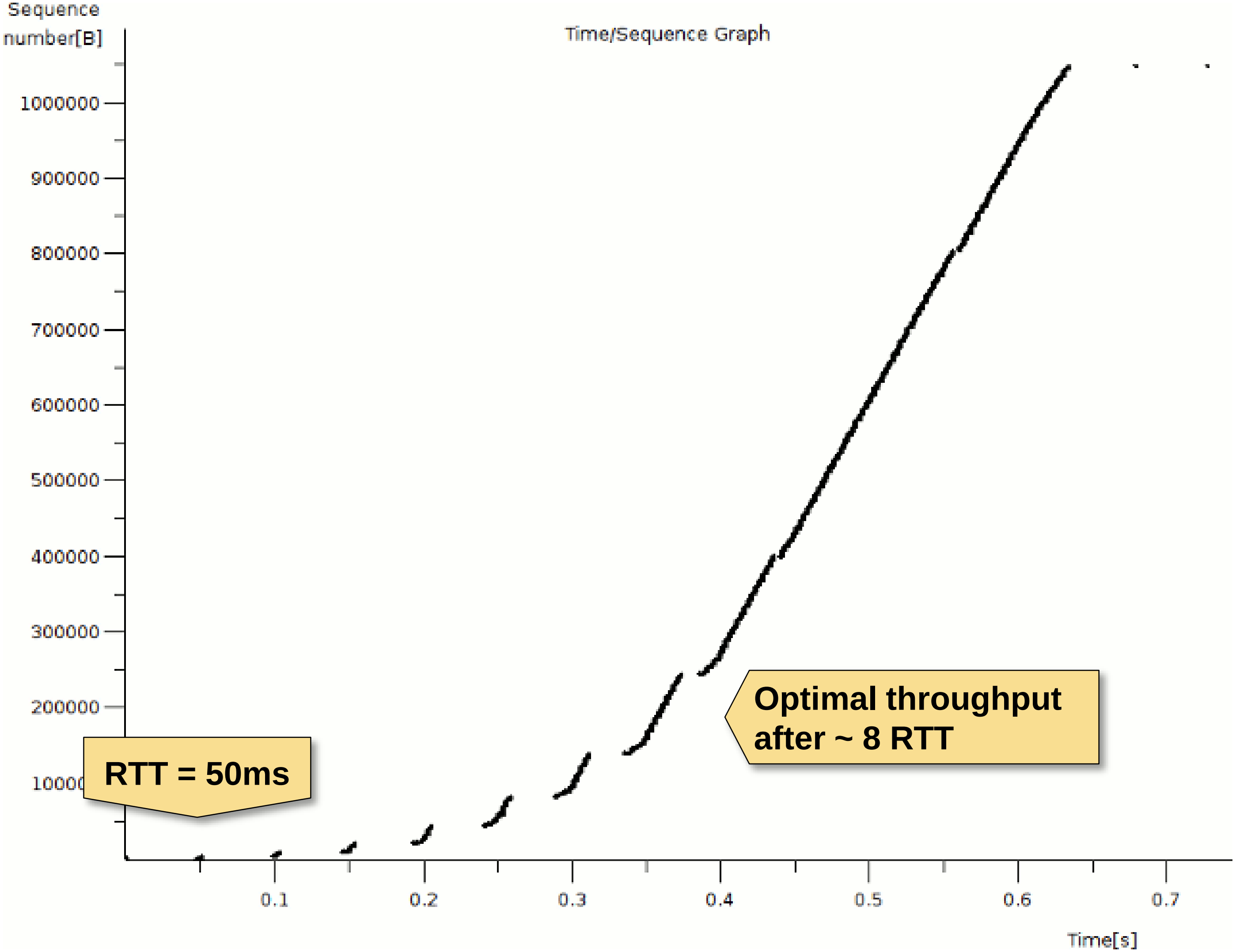


- TCP was developed in 1980s
- Major problems: congestion, buffer memory utilization

Mechanisms

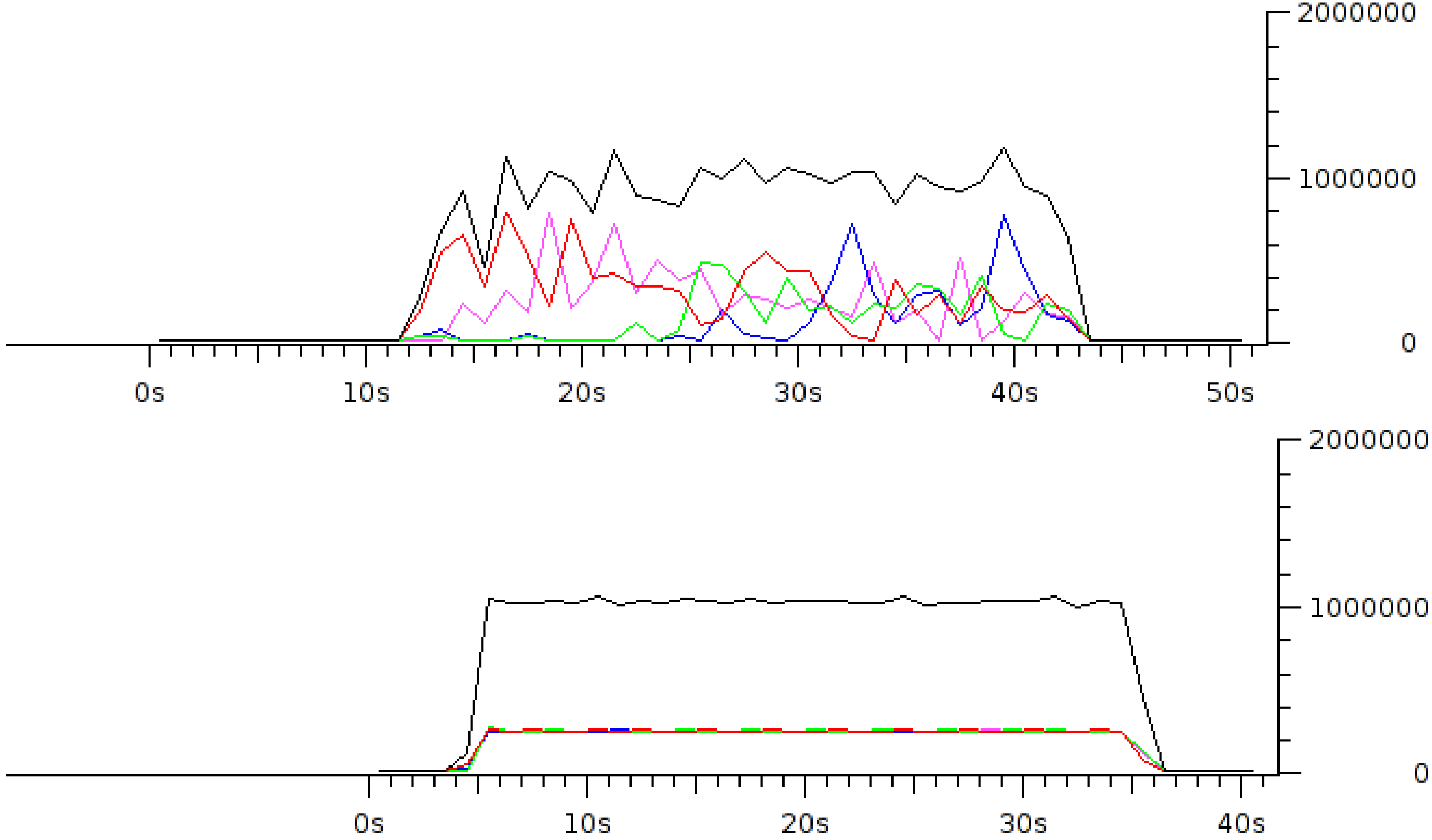
- Window = maximum amount of unacknowledged data
- Congestion window = window size reduced to avoid congestion
- Default: initial congestion window = 3 packets (3 x 1460 bytes)

TCP Slow Start



Source: http://packetlife.net/blog/2011/jul/5/tcp-slow-start/

TCP: Impact of Packet Drops



Source: http://wiki.nil.com/Policing_vs_shaping

Let’s Recap

Web load time is influenced by TCP and HTTP

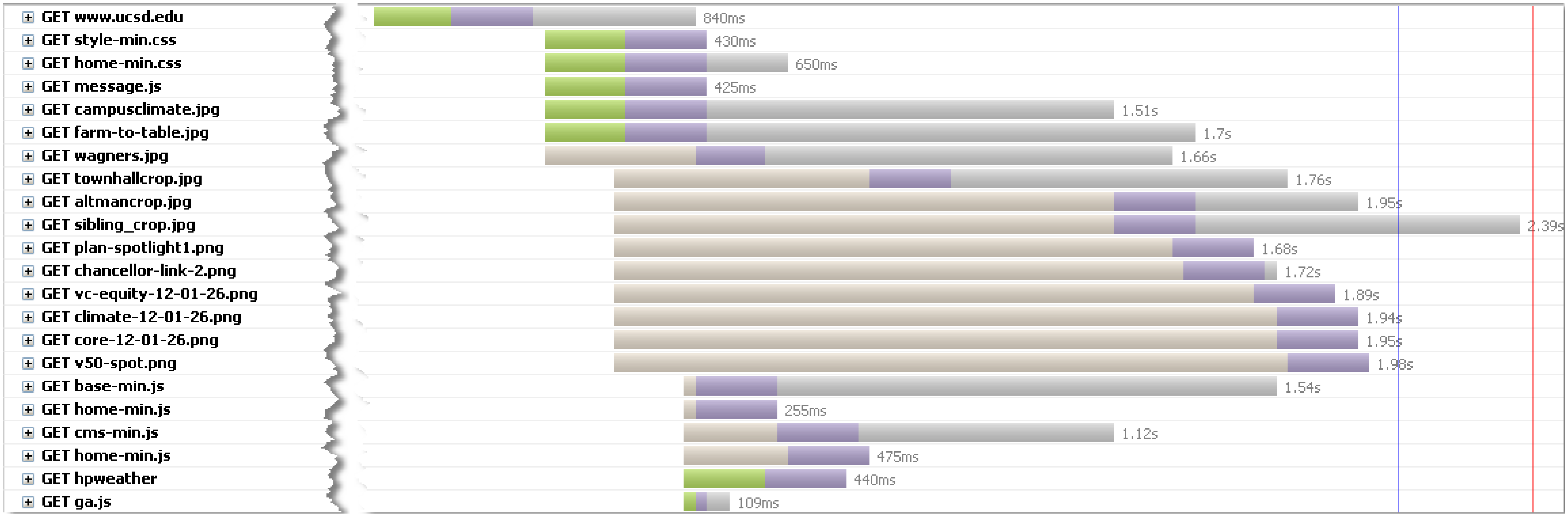
- One RTT to establish the TCP session
- Second RTT to send HTTP request and get response data
- Third RTT to get more than 3 packets of response data
- Slow down on packet loss

What can be done?

- Parallel TCP sessions
- Reuse TCP sessions (persistent HTTP connections, SPDY)
- Pre-establish TCP sessions
- Increase initial congestion window on servers (Google: 10)
- Send HTTP GET request with TCP SYN (TCP Fast Open)
- Use CDN to reduce RTT

TCP and HTTP Improvements

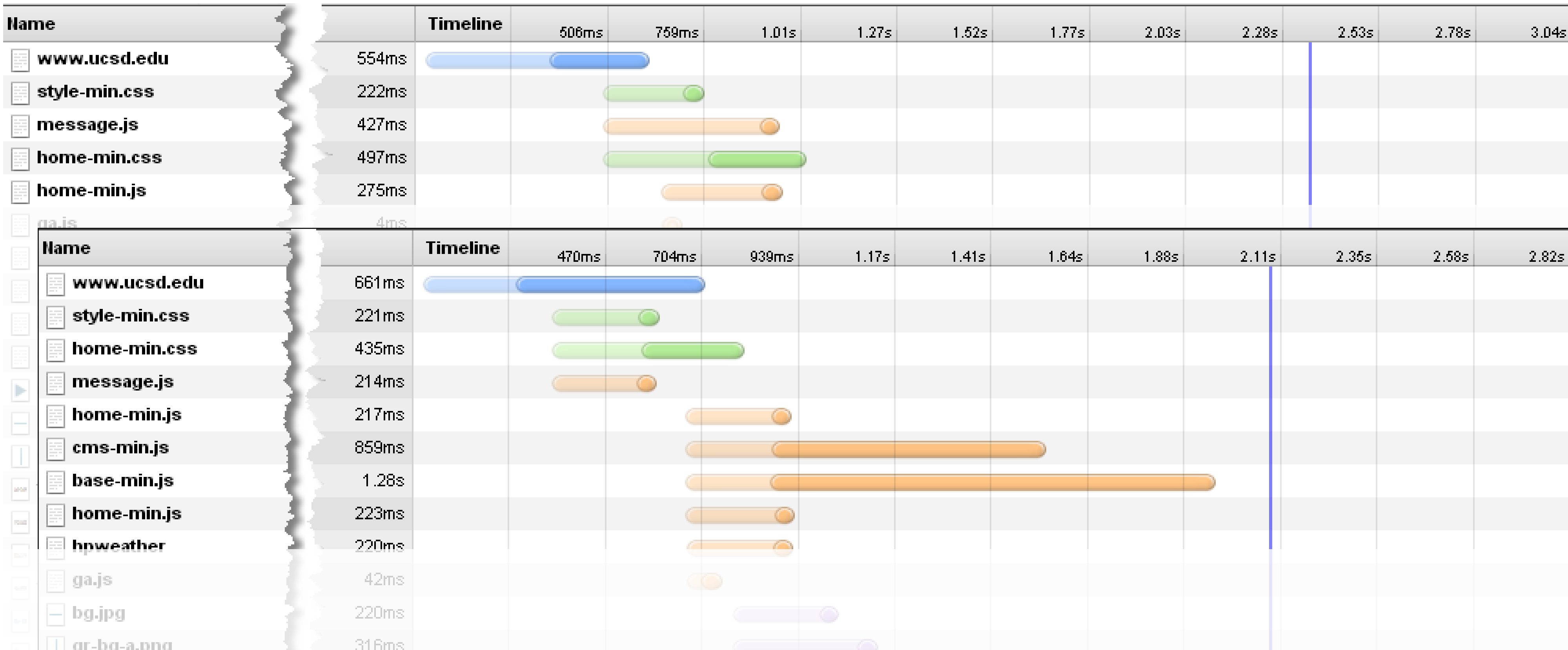
Parallel TCP Sessions: ucsd.edu On Firefox 9.0.1



- 6 sessions per hostname
- Additional sessions are established after the initial response is parsed
- JavaScript is loaded before images

More details: http://www.browserscope.org/?category=network&v=top

Parallel TCP Sessions: ucsd.edu On Chrome 24.0



- 6 sessions per hostname
- Additional sessions are pre-established on second access
- JavaScript is loaded before images

Persistent HTTP Sessions

Response Headers view source

Accept-Ranges	bytes
Connection	Keep-Alive
Content-Encoding	gzip
Content-Type	text/html; charset=UTF-8
Date	Tue, 22 Jan 2013 09:54:17 GMT
Etag	"6948-4d395240"
Keep-Alive	timeout=15, max=99
Last-Modified	Fri, 18 Jan 2013 20:53:21 GMT
Server	Apache/2.0.63 (Unix) DAV/2 mod_perl/2.0.4 Perl/v5.8.4
Transfer-Encoding	chunked
Vary	Accept-encoding

- Persistent sessions introduced in HTTP 1.1
- TCP session is not closed after HTTP response is sent
- Enabled by default on all major web servers and browsers

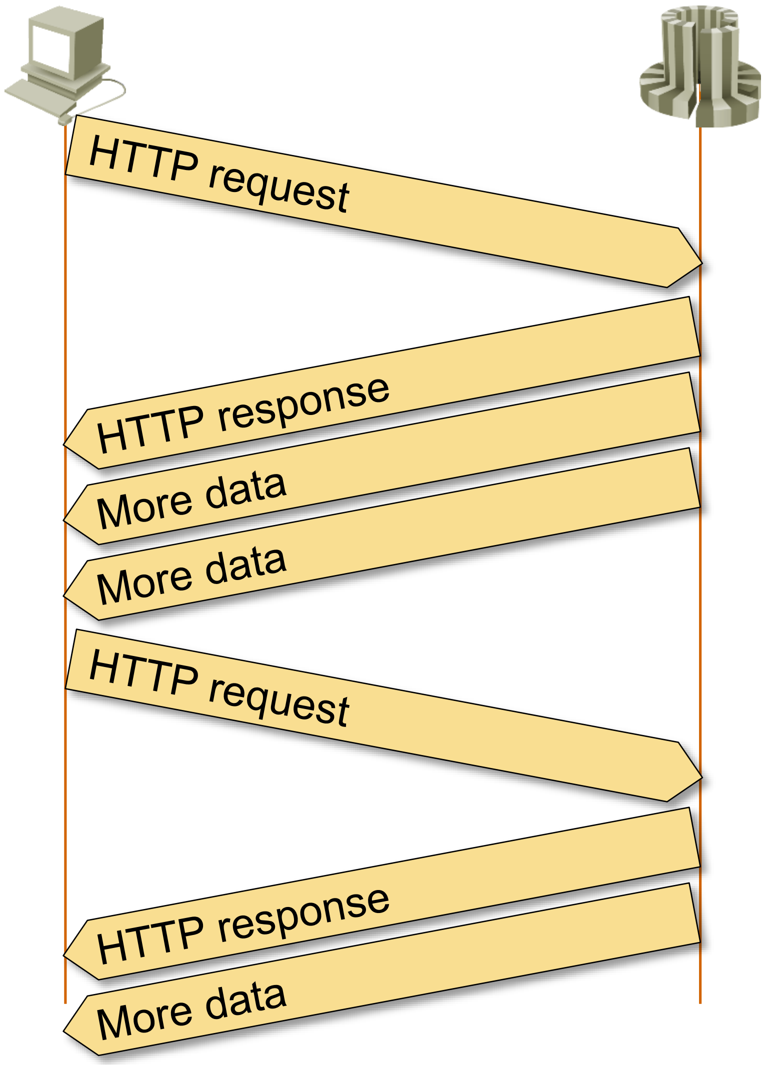
Benefits

- One RTT is saved on subsequent HTTP requests

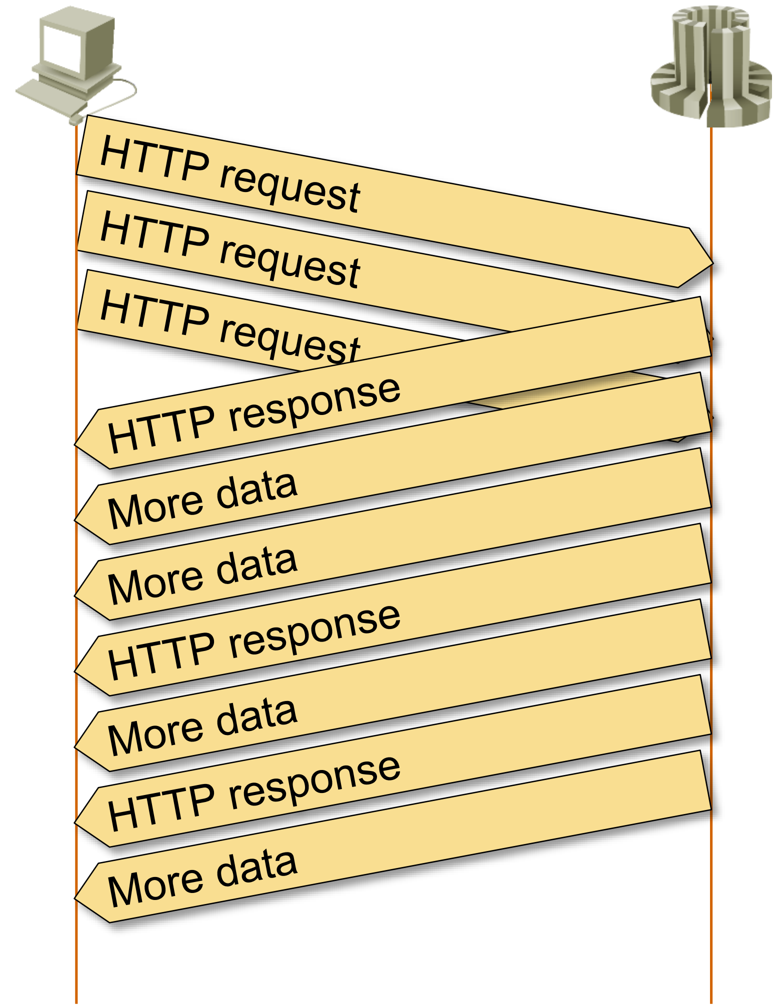
Drawbacks

- Each persistent HTTP session consumes a thread or worker process

HTTP Pipelining



Persistent HTTP session



HTTP pipelining

HTTP Pipelining

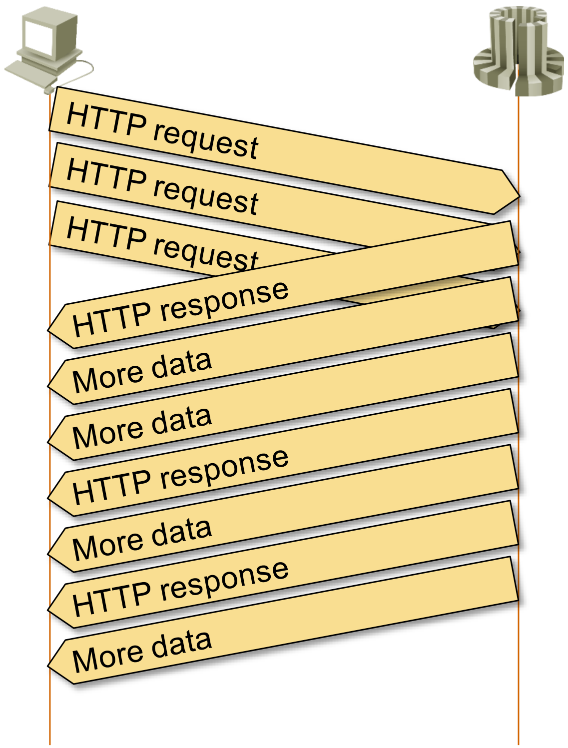
- Multiple HTTP requests are sent without waiting for HTTP response
- Not widely used, has to be enabled manually

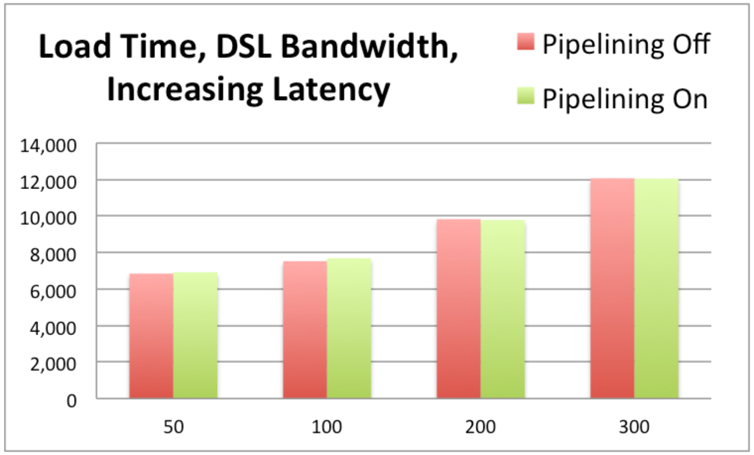
Benefit

- One RTT is saved for all subsequent requests

Drawback

- Head-of-line blocking of response data
- Hard to select optimal sequence of requests





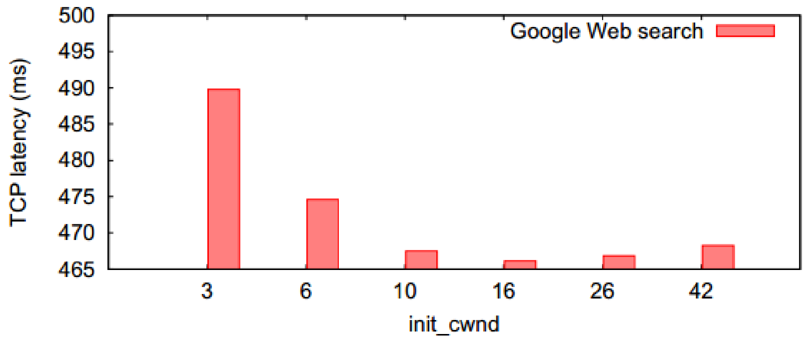
Source: http://www.guypo.com/technical/http-pipelining-not-so-fast-nor-slow/

Increase Initial Congestion Window

Google's proposal:

- Set initial congestion window (cwnd) to 10
- Up to ~15K of HTTP response (reasonably-big web pages) delivered in a single RTT
- Does not impact existing L4+ middleboxes
- Minimal impact on the Internet
- Already used by Google and some large CDN
- Easy to configure on a Linux server

```
# ip route change default via 192.168.200.193
   dev eth0 initcwnd 10
```



	All	Web search	Maps	Photos
AvgDC				
Exp	2.29 [6.26]	1.73 [5.63]	4.17 [7.78]	2.64 [11.14]
Base	1.98 [6.24]	1.55 [5.82]	3.27 [7.18]	2.25 [10.38]
Diff	0.31 [0.02]	0.18 [-0.20]	0.90 [0.60]	0.39 [0.76]
SlowDC				
Exp	4.21 [8.21]	3.50 [10.44]	5.79 [9.32]	6.10 [22.29]
Base	3.54 [8.04]	2.98 [10.17]	3.94 [7.36]	4.97 [19.99]
Diff	0.67 [0.17]	0.52 [0.26]	1.85 [1.97]	1.12 [2.30]

Percentage of retransmissions

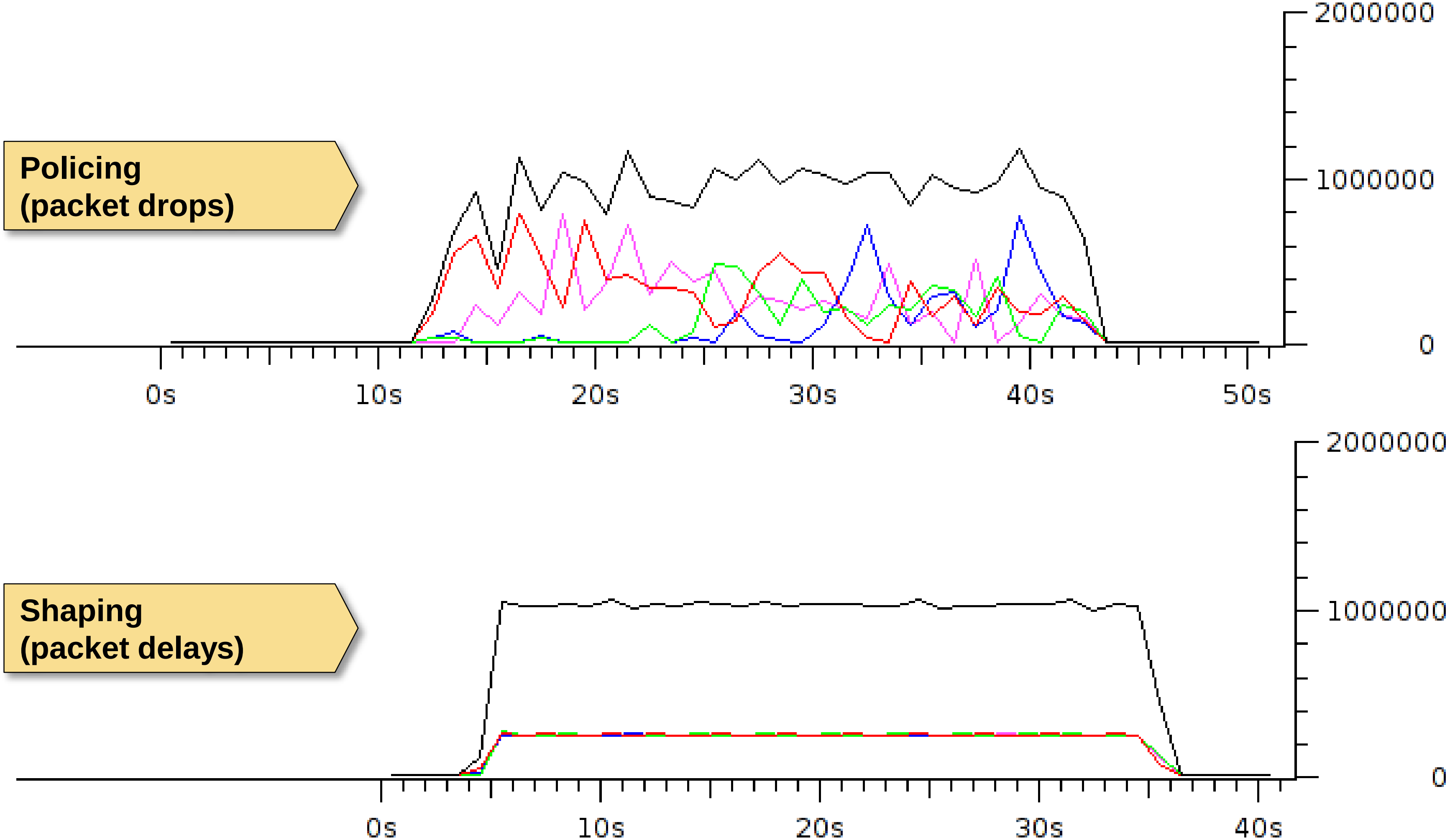
Sources:

https://developers.google.com/speed/articles/tcp_initcwnd_paper.pdf

http://www.cdnplanet.com/blog/initcwnd-settings-major-cdn-providers/

http://www.cdnplanet.com/blog/tune-tcp-initcwnd-for-optimum-performance/

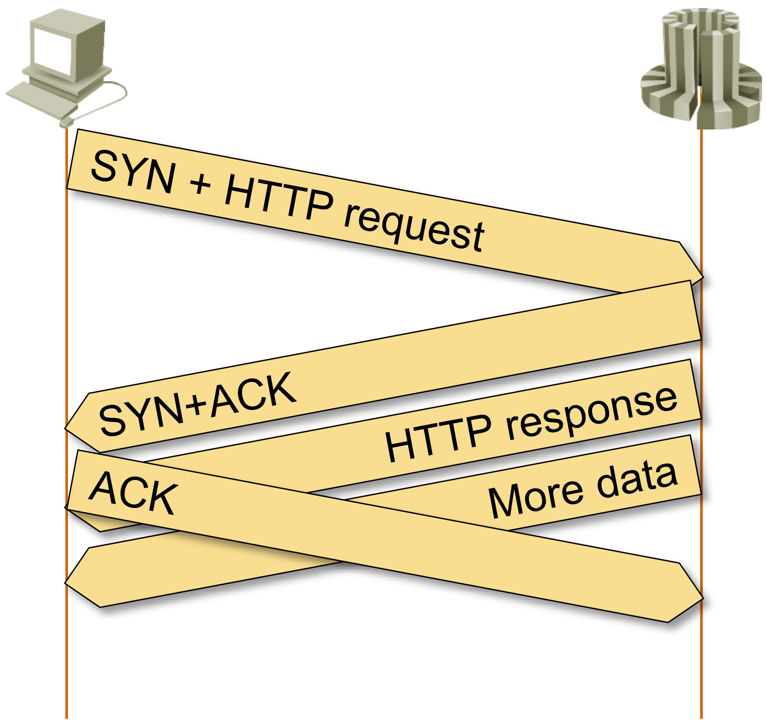
Retransmissions Matter a Lot



Source: http://wiki.nil.com/Policing_vs_shaping

TCP Fast Open (Experimental)

- HTTP request is sent in SYN packet
- Server processes HTTP request before 3-way handshake completes
- Response data is sent before initial client ACK



Benefit: One RTT is saved

Drawbacks

- Duplicate SYN packets
  ➔ works only for idempotent transactions
  ➔ wasted server resources
- SYN floods are more harmful
  ➔ protection with fast open cookie
- L4+ firewalls might intercept and drop packets with SYN+data or TFO options

Sources:

http://research.google.com/pubs/pub37517.html

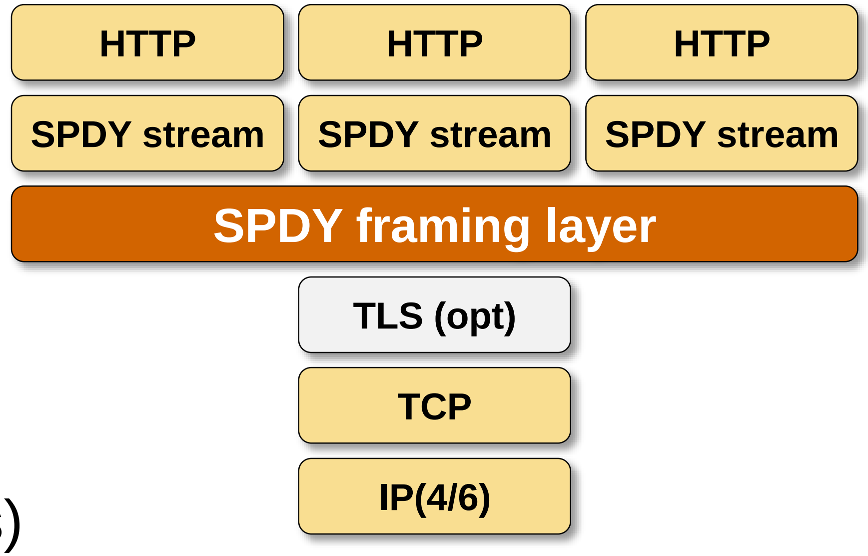
http://tools.ietf.org/html/draft-ietf-tcpm-fastopen-01

SPDY

SPDY Overview

What is it?

- Framing layer implementing streams above TCP or TLS
- Optimized for HTTP



How does it work?

- Single client-server TCP connection (based on IP addresses, not host names)
- HTTP requests and responses streamed in parallel over the TCP session

Optimizations

- Compressed HTTP headers in SYN_STREAM and SYN_REPLY requests
- Chunked responses prevent head-of-line blocking

More @ http://tools.ietf.org/html/draft-mbelshe-httpbis-spdy-00

Creating SPDY streams

SYN_STREAM creates a new stream

- Streams can be bidirectional (regular HTTP requests) or unidirectional (server push)
- *Stream-ID* is odd for client-created streams, even for server-created streams
- *Priority* can be used for priority queuing within SPDY TCP session
- *FIN* flag has the same meaning as in TCP

```
+----------------------------------+
|1|     version     |         1    |
+----------------------------------+
|  Flags (8)  |  Length (24 bits)  |
+----------------------------------+
|X|           Stream-ID (31bits)   |
+----------------------------------+
|X| Associated-To-Stream-ID (31bits) |
+----------------------------------+
| Pri|Unused | Slot |              |
+-------------------+              |
| Number of Name/Value pairs (int32) |
+----------------------------------+
|     Length of name (int32)       |
+----------------------------------+
|           Name (string)          |
+----------------------------------+
|     Length of value  (int32)     |
+----------------------------------+
|          Value   (string)        |
+----------------------------------+
|           (repeats)              |
```

HTTP layering

- HTTP headers transported in name/value pairs
- Name/values block is compressed with zlib, initial dictionary specified in SPDY draft

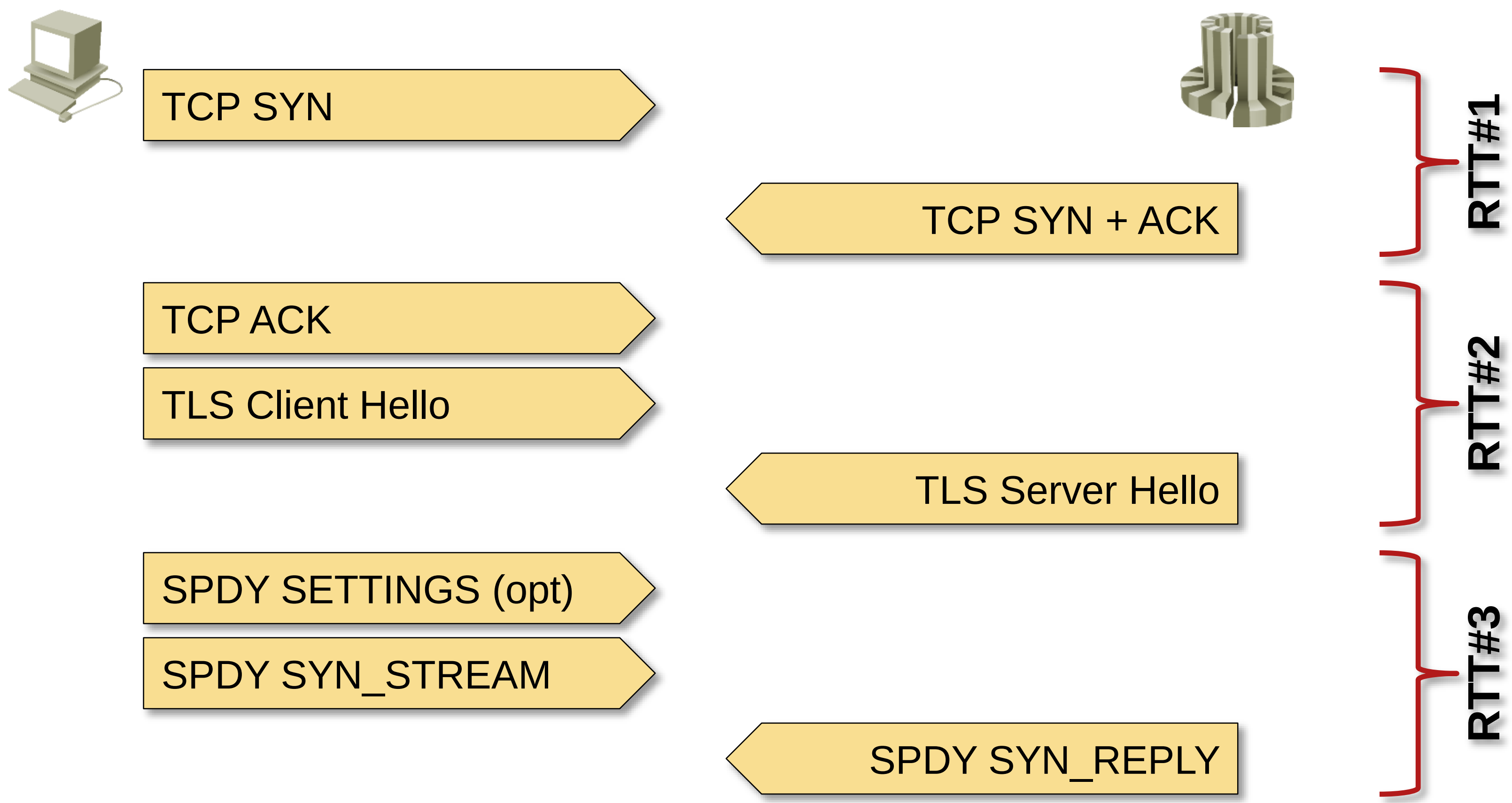
Stream Acceptance

SYN_REPLY indicates stream acceptance

- *FIN* flag sent when this is the last message in this stream
- HTTP headers sent compressed in name/value block
- Additional data sent in DATA frames

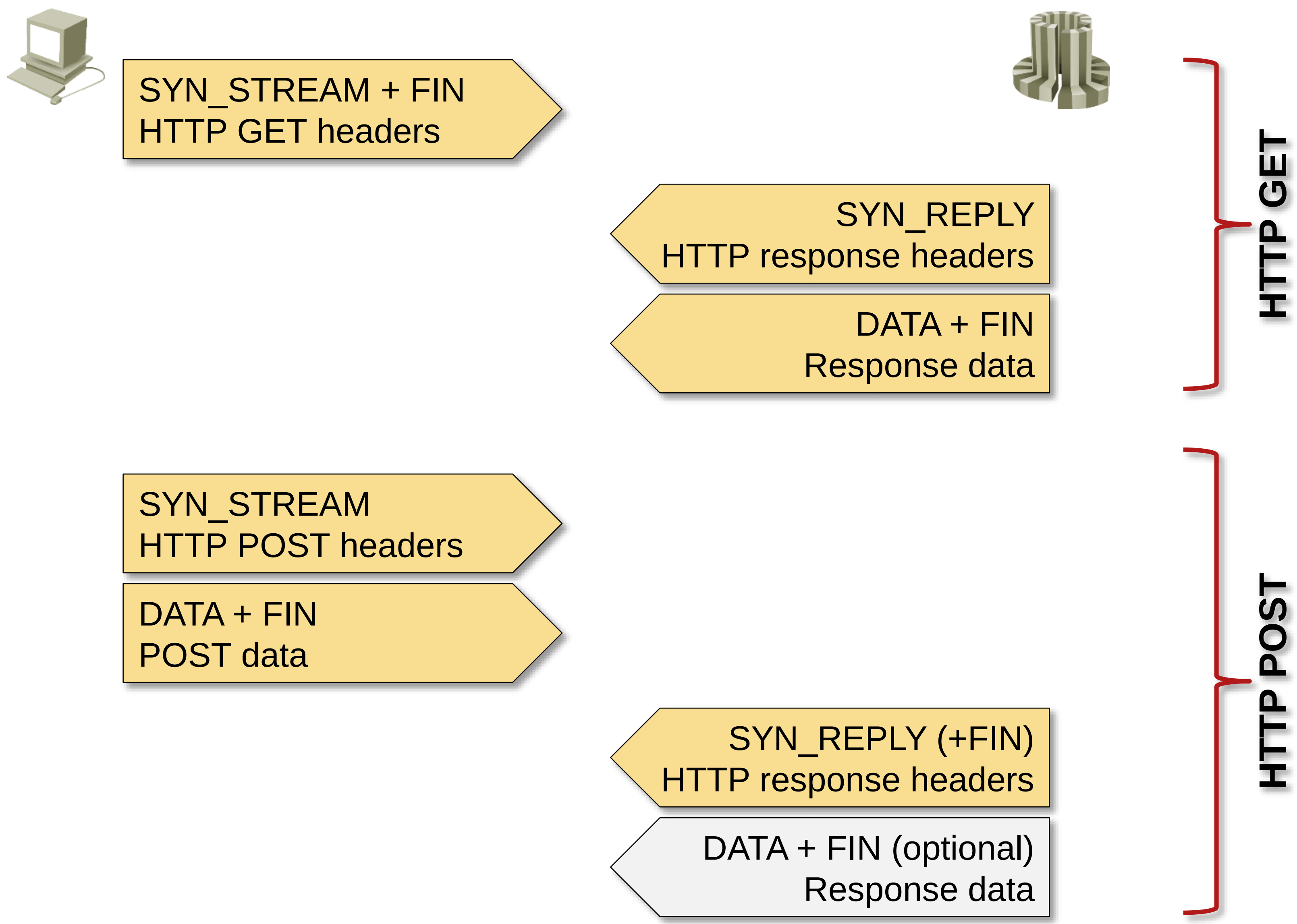
```
+----------------------------------+
|1|    version     |        2      |
+----------------------------------+
|  Flags (8)   |   Length (24 bits) |
+----------------------------------+
|X|          Stream-ID (31bits)    |
+----------------------------------+
| Number of Name/Value pairs (int32) |
+----------------------------------+
|     Length of name (int32)       |
+----------------------------------+
|          Name (string)           |
+----------------------------------+
|     Length of value  (int32)     |
+----------------------------------+
|          Value   (string)        |
+----------------------------------+
|           (repeats)              |
```

SPDY Session Setup



- SPDY is commonly used over TLS
- Three RTTs (without DNS lookup) before first data arrives

Sample SPDY Transactions



Streamlined SPDY Response Delivery

SYN_STREAM (HTML)

SYN_REPLY (HTML)

DATA (HTML)

RTT#1

SYN_STREAM PRIO = 0 (CSS)

SYN_STREAM PRIO = 0 (JS)

SYN_STREAM PRIO = 7 (PNG)

SYN_REPLY (CSS)

SYN_REPLY (JS)

SYN_REPLY (PNG)

SYN_DATA (CSS)

SYN_DATA (JS)

SYN_DATA (JS)

RTT#2

- All data delivered in two RTTs + xfer time
- Long-lived TCP session used by SPDY would have large cwnd size
- Works well when RTT >> transfer time

Is SPDY Faster Than HTTP

Google: YES

- Sites load twice as fast
- Using fine-tuned examples with lots of images
  http://blog.chromium.org/2009/11/2x-faster-web.html
  http://googledevelopers.blogspot.com/2012/04/add-spdy-support-to-your-apache-server.html

Microsoft and others: NO (OK, maybe a little)

- SPDY+TLS is comparable to HTTPS + pipeline
- SPDY + minify is approximately as fast as HTTP + pipeline + minify
- SPDY+TLS is slower than HTTP due to extra RTT
  http://research.microsoft.com/apps/pubs/default.aspx?id=170059
- SPDY is approximately as fast as HTTPS on real-life data
  http://www.guypo.com/technical/not-as-spdy-as-you-thought/

Real-Life SPDY

SPDY prerequisites:

- Web server with SSL/TLS (SPDY w/o TLS is rare)
- Next-Protocol Negotiation extensions for TLS (custom mod_ssl required for Apache)

Real-life deployment and availability:

- Starting point for HTTP 2.0
- Available in Chrome and Firefox
- mod-spdy for Apache (from Google Code), patches for nginx, development code for haproxy
- SPDY supported by F5 WebAccelerator
- Used by several large production web sites (Google, Wordpress, Cloudflare)
- Use SPDYCheck.Org to check SPDY status

Conclusions

Conclusions

- Web pages should load in < 100 msec, worst case in few seconds

Obstacles on the road to the holy grail

- Non-zero latency and non-infinite bandwidth
- Short-lived TCP sessions
- Request-response nature of HTTP

What could help:

- Parallel TCP sessions or SPDY
- HTTP pipelining

What will help:

- Good web site design
- Minification, compression and caching

Questions?