A Question of Timing

10/17/2018, 12:00:00 PM UTC

Piers Cornwell

Photo by Aron / Unsplash

When considering website performance, the term TTFB - time to first byte - crops up regularly. Often we see measurements from cURL and Chrome, and this article will show what timings those tools can produce, including time to first byte, and discuss whether this is the measurement you are really looking for.

Timing with cURL

cURL is an excellent tool for debugging web requests, and it includes the ability to take timing measurements. Let’s take an example website www.zasag.mn (the Mongolian government), and measure how long a request to its home page takes:

First configure the output format for cURL in `~/.curlrc`:

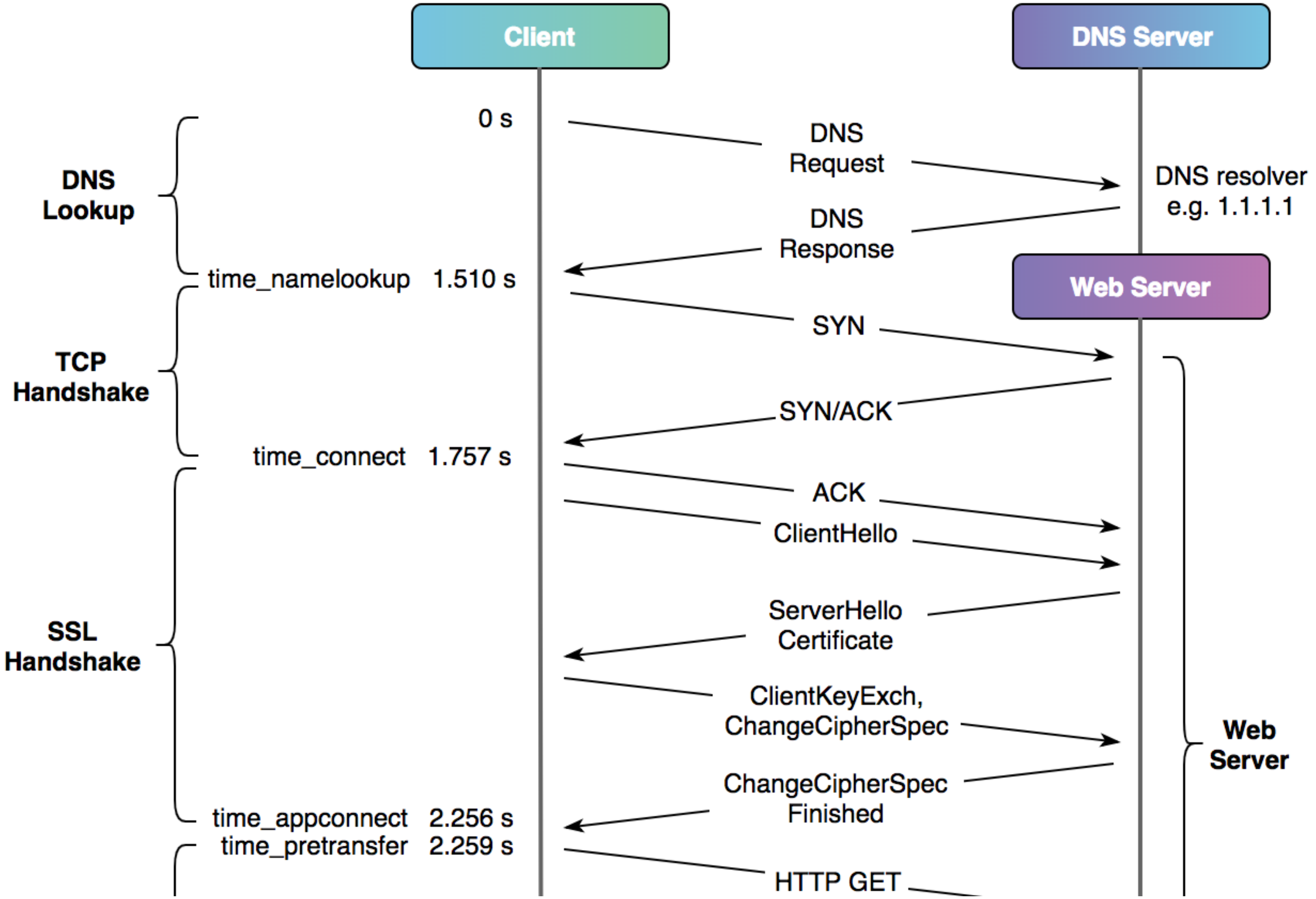
```
$ cat .curlrc
-w "dnslookup: %{time_namelookup} | connect: %{time_connect} | appconnect: %{time_appconnect}
```

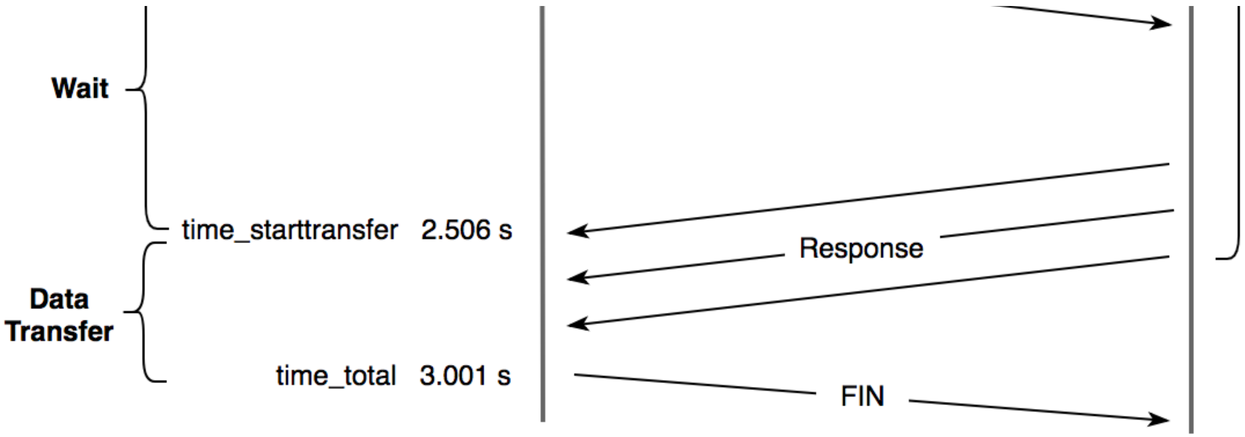
Now connect to the site dropping the output (`-o /dev/null`) since we’re only interested in the timing:

```
$ curl -so /dev/null https://www.zasag.mn
dnslookup: 1.510 | connect: 1.757 | appconnect: 2.256 | pretransfer: 2.259 |
starttransfer: 2.506 | total: 3.001 | size: 53107
```

These timings are in seconds. Depending on your version of cURL, you may get more decimal places than this example. 3 seconds is a long time, and remember this is only for the HTML from the home page - it doesn't include any JavaScript, images, etc.

The diagram below shows what each of those timings refer to against a typical HTTP over TLS 1.2 connection (TLS 1.3 setup needs one less round trip):





- **time_namelookup** in this example takes a long time. To exclude DNS resolver performance from the figures, you can resolve the IP for cURL: `--resolve www.zasag.mn:443:218.100.84.167`. It may also be worth looking for a faster resolver :).

- **time_connect** is the TCP three-way handshake from the client's perspective. It ends just after the client sends the ACK - it doesn't include the time taken for that ACK to reach the server. It should be close to the round-trip time (RTT) to the server. In this example, RTT looks to be about 200 ms.

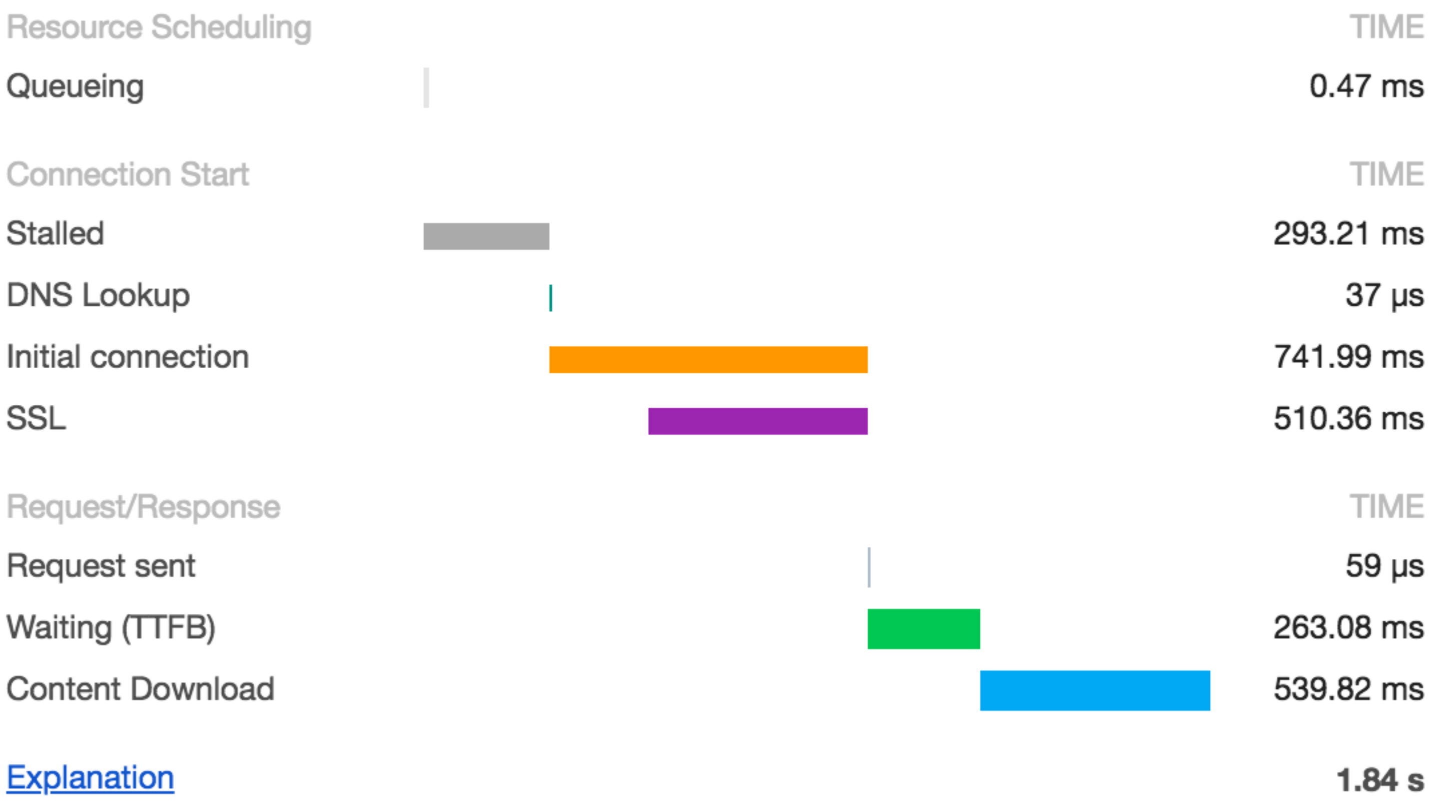
- **time_appconnect** here is TLS setup. The client is then ready to send it’s HTTP GET request.

- **time_starttransfer** is just before cURL reads the first byte from the network (it hasn't actually read it yet). `time_starttransfer - time_appconnect` is practically the same as Time To First Byte (TTFB) from this client - 250 ms in this example case. This includes the round trip over the network, so you might get a better guess of how long the server spent on the request by calculating `TTFB - (time_connect - time_namelookup)`, so in this case, the server spent only a few milliseconds responding, the rest of the time was the network.

- **time_total** is just after the client has sent the FIN connection tear down.

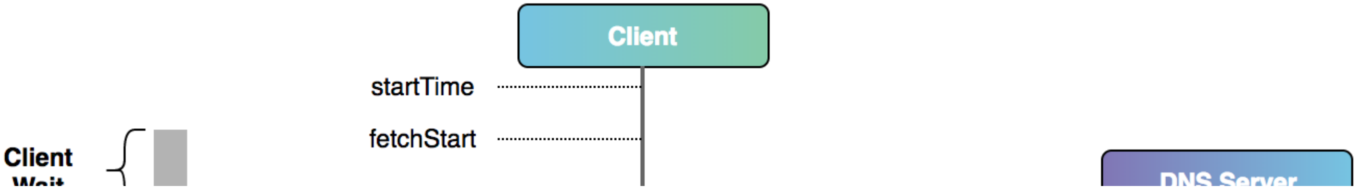
Timing with Chrome

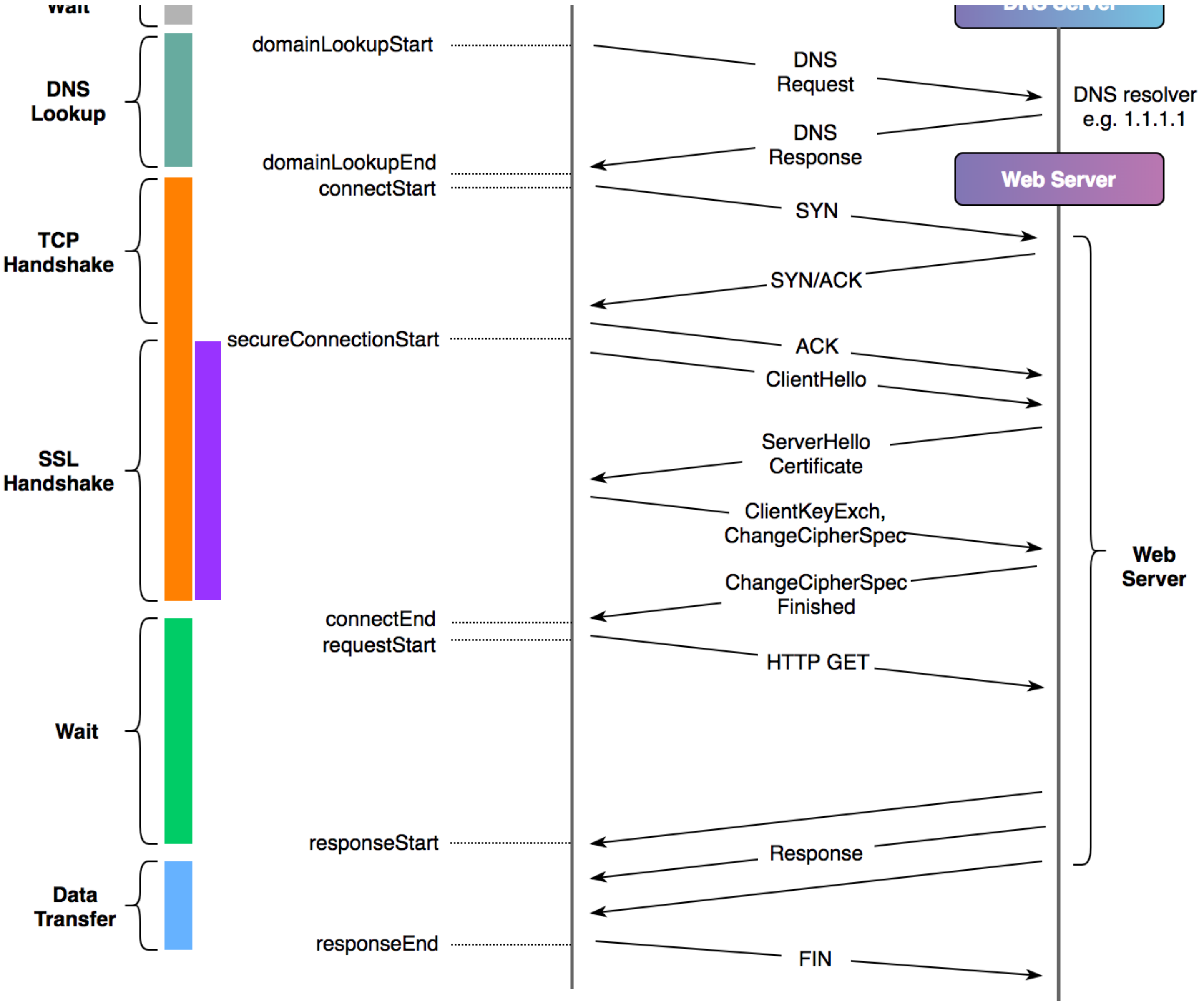
Chrome, and some other testing tools, use the W3C Resource Timing standard for measurements. In Chrome developer tools this looks like this:



Resource Scheduling	TIME
Queueing	0.47 ms
Connection Start	**TIME**
Stalled	293.21 ms
DNS Lookup	37 µs
Initial connection	741.99 ms
SSL	510.36 ms
Request/Response	**TIME**
Request sent	59 µs
Waiting (TTFB)	263.08 ms
Content Download	539.82 ms
Explanation	1.84 s

Again, here's how this maps onto a typical HTTP over TLS 1.2 connection, also showing the Resource Timing attribute names:




Wait
DNS Lookup
TCP Handshake
SSL Handshake
Wait
Data Transfer
domainLookupStart
domainLookupEnd
connectStart
secureConnectionStart
connectEnd
requestStart
responseStart
responseEnd
DNS Server
DNS Request
DNS Response
DNS resolver e.g. 1.1.1.1
Web Server
SYN
SYN/ACK
ACK
ClientHello
ServerHello Certificate
ClientKeyExch, ChangeCipherSpec
ChangeCipherSpec Finished
HTTP GET
Response
FIN
Web Server

- **Stalled** (fetchStart to domainLookupStart) is the browser waiting to start the connection, e.g. allocating cache on disk, if there are higher priority requests, or if there are already 6 connections open to this host.

- **Initial connection** shown by Chrome is connectStart to connectEnd. Unlike cURL timings, this includes SSL connection setup, so if you want a fair estimate of RTT, this would be `Initial connection - SSL`. If an existing connection is being reused, then DNS Lookup, Initial connection and SSL won't be shown.

- **Request sent** is `connectEnd - requestStart`, which should be negligible.

- Similarly to cURL, if we subtract the TCP handshake time from TTFB, we can guess the amount of time the server really spent processing (again, we don't have an exact RTT timing, so this is a approximation).

What are we looking for again?

These measurements, including TTFB, can be helpful in diagnosing problems, and might help you to delve into a specific problem, but do they actually tell you about how well a website is performing? Ultimately, if you are looking to measure the experience of users, the time it takes for the first byte of some HTML to return isn’t effective. A web page might contain hundreds of images, it might have JavaScript and styles that need to load before you can interact. To reflect real user experience, you need to time how long until the web page becomes useful, and to take those measurements from representative sample of where your users are accessing the site from. And that's a topic for another day :)

Speed & Reliability Chrome

RELATED POSTS

December 31, 2019 7:13PM

Adopting a new approach to HTTP prioritization

HTTP prioritization is important for web performance. This is the story behind a new approach recently adopted for further work in the IETF....

By Lucas Pardue

Speed & Reliability, Product News

August 13, 2019 2:01PM

Magic Transit makes your network smarter, better, stronger, and cheaper to operate

Today we're excited to announce Cloudflare Magic Transit. Magic Transit provides secure, performant, and reliable IP connectivity to the Internet. Out-of-the-box, Magic Transit deployed in front of your on-premise network protects it from DDoS attack...

By Rustam Lalkaka

Speed & Reliability, Magic Transit, Product News

August 13, 2019 2:00PM

Magic Transit: Network functions at Cloudflare scale

Today we announced Cloudflare Magic Transit, which makes Cloudflare's network available to any IP traffic on the Internet. Up until now, Cloudflare has primarily operated proxy services: our servers terminate HTTP, TCP, and UDP sessions...

By Nick Wondra

Speed & Reliability, Magic Transit, Network, Anycast, TCP

May 14, 2019 5:00PM

Parallel streaming of progressive images

Image-optimized HTTP/2 multiplexing makes all progressive images across the page appear visually complete in half of the time....

By Andrew Galloni, Kornel Lesiński

Speed & Reliability, Speed Week, Optimization

2 Comments   Cloudflare Blog   🔒 Disqus' Privacy Policy   1 Login

♡ Recommend 5   Tweet   Share   Sort by Best

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Daniel Kanchev • 2 years ago

Piers, thanks for the interesting post!

I totally agree with you that people should measure when a site becomes usable and not when the first byte is delivered to the client.

Still, very few people understand the technical details and often end users blindly trust performance testing tools. One such tool is Google PageSpeed Insights - it is very popular and people use it all the time.

I checked the same site with Google PageSpeed Insights and according to the tool the server needed 1.3 seconds to "load the necessary HTML":

https://developers.google.c...

Google defines this as Server Response Time and more info is available here:

https://developers.google.c...

Do you have any idea what this Server Response Time actually means and what is included in it. It would be nice to be able to reverse engineer such tools so we can better understand them.

^ | ⌄ • Reply • Share ›

Šime Vidas • 2 years ago

Sorry for going off topic, but would you say that ~1750 ms is a realistic TTFB value over slow 3G? It seems out of reach according to my tests on WebPageTest. I’ve started a discussion on their forum here:

https://www.webpagetest.org...

^ | ⌄ • Reply • Share ›

✉ **Subscribe** **Add Disqus to your site** ⚠ **Do Not Sell My Data** **DISQUS**