Caffeine explained

While having to deal with mobile networking, Caffeine resulted from years of continued frustration. It's a vertically integrated stack spanning from application-authentication to TCP. It is a protocol, a web server equivalent, and a networking library *for* iOS.

Native mobile applications are very different than websites/servers, and they should be treated as such. They don't need all the bulk on the web. When we ask ourselves what is *essential* within iOS communications, we get Caffeine. Applications can now be faster than ever thought before.

The Two Fundamental Speed Limitations in networking

- Bandwidth: bits per second of data
- Latency: time between a request and a response

Bandwidth is ultimately limited by the ISP's networking hardware. Latency is primarily limited by two features: the speed of electricity through electrical wire—very roughly 1/2 the speed of light in a vacuum (1/2c), and the geographic efficiency of the network path (straightness of line). More efficient protocols require less bandwidth and less latency.

Caffeine aggressively targets **both** the bandwidth **and** latency efficiency of the request. We send less data and incur fewer round-trips than the equivalent HTTPS implementation. The exact techniques we use to do this are proprietary, but the underlying math is very simple.

An Example

If we allow that there are 6000km from Boston to Switzerland (where the server was in image loading demo) and that the speed of electricity is 1/2c, then a round-trip request/response there and back again is about 20ms. If we further allow a 2x inefficiency because the networking cable is not in a straight line and that each of the 20 hops on the path add about 2ms of queuing latency then we are up to 80ms. This estimate is roughly consistent with empirical results from ping (using the ICMP).

There is no reason why we could not theoretically make a request in 80ms. So it is *this* figure (80ms in this scenario) that is the practical limit. When you profile Caffeine's performance along the path we are comfortably slower (~120ms in many cases). So we are not violating any fundamental speed limit; rather we are actually only achieving ~60% of theoretical speeds. Far from our performance being impossible, we have plenty of headroom to improve further.

Our performance advantage comes down to the basic fact that while we are 60% efficient, REST/HTTPS are much, much worse. In many cases the design of those protocols themselves are inefficient, in other cases the implementations are poor and have not been carefully tuned and profiled. The exact inefficiencies are many and complex, and do not apply equally in every scenario, but as long as you're not seeing 80ms speeds, some of them apply.

A more detailed walkthrough may be found here.

Datacenter-grade Communications

It's also important to note that we're not the first people to realize this; in fact our research suggests that both Facebook and Amazon are doing things like this inside their data-centers: both designing new protocols, and carefully tuning the implementations of the existing ones. But their technology is tied closely to their own products, and is kept hidden behind the firewall. Caffeine is simply "datacenter-grade" networking that allows organizations to attain product parity to Amazon.