# You Ain't SPDY

By: Joe Sak

April 14, 2012

This article is the seventh and last part in a series of RubyNation Recaps (http://www.neotericdesign.com/blog/the-rubynation-2012-recap-series). The first part was Cannibalize Yourself (http://www.neotericdesign.com/blog/cannibalize-yourself-rubynation-recap), and the introductory post contains a list of all the articles posted (http://www.neotericdesign.com/blog/the-rubynation-2012-recap-series) and coming up.

# You Ain't SPDY

by Chris Strom

Chris is the author of The SPDY Book and Dart for Hipsters. He is also the co-author of Recipes with Backbone. He organizes the B'more on Rails user group. Chris is a relentless public learner as evidenced by 700+ blog posts. He is currently a contributor on both the Ruby SPDY gem as well as the node-spdy package.

**Book:** The SPDY Book (Making Websites Fly) (http://spdybook.com/)

## Why SPDY?

Deliver web pages and sophisticated web applications faster and more securely than is possible with vanilla HTTP

- SPDY is a replacement for HTTP
- It's backwards compatible with HTTP
- It is optimized for page load times.

Bandwidth doesn't matter beyond 3mbps. **It makes no difference for page load times.**

Why? Because bandwidth is for something like downloading Ubuntu, but not loading the Ubuntu download web page.

**HTTP is antiquated, and hasn't been updated for 12 years.**

SPDY is the most heavily tested protocol… ever. It's in use by Gmail.

Google Search, Chrome, Gmail, Firefox, and Twitter are all SPDY enabled.

Case study: How does Amazon load their pages so fast?

A lot, *a lot*, **a lot** of optimization. ***A lot***. An entire department of developers is dedicated to implementing all the typical rules (think YSlow) for optimizations. But those are not HTTP optimizations! They are HTTP workarounds!

Here's how HTTP works (more or less)

Data travels one way through the HTTP tube.

- Client request -> Server
- server processing.... Server response -> Client
- Client request -> Server
- server processing.... Server response -> Client
- and so on...

Every modern browser has not one, but six http tubes to download resources in parallel. Problem solved, right? Well, then why are we still here talking about this? Connections need to be warmed up. Seriously.

The tubes start off cold, and can't transfer as much data as when they've been warmed up. This is how TCP/IP works. The connection makes round trips to pick up just a few bits of information so that it doesn't congest the network. Then it learns how much more it can take on each trip. This is why it starts off cold. This is why bandwidth doesn't matter. HTTP connections have to go over TCP/IP.

## Why not start the tubes warm?

The large congestion window (http://en.wikipedia.org/wiki/Slow-start) (CWND) is initially three

With six connections, we already have 18 initial CWNDs

Google, Yahoo, Facebook, etc… are already doing this on their networks.

### Enter the SPDY tube

Any time the data is ready, it goes. SPDY conversations are crossing the streams all the time. This is not a new idea.

- Built on SSL
- Binary
- Doesn't send redundant header information
- Aggressively compresses resources
- Uses a single(!) tube
- --- Only pay the warm-up penalty once
- --- Page loading is now just like downloading an Ubuntu ISO!!! (this guy loves Ubuntu)

### SPDY support in Ruby (not 100%)

- Openssl-generated server key
- SPDY gem (it does work)
- Edge-openssl for NPN (Next Protocol Negotiation)
- Carson McDonald's NPN enabled fork of eventmachine (in the tls-npn branch) (https://github.com/carsonmcdonald/eventmachine/tree/tls-npn) - pull request #196 (https://github.com/eventmachine/eventmachine/pull/196)

For full information, please check out Chris' book: The SPDY Book (Making Websites Fly) (http://spdybook.com/)

… and this concludes the RubyNation 2012 recap series! I hope you've enjoyed it. I certainly did. I'll post links to the videos once I'm aware that they are available. Until then, you can follow @rubynation on twitter (http://twitter.com/rubynation) to stay informed! You can also find me on twitter (http://twitter.com/joemsak), as well as @neotericdesign (http://twitter.com/neotericdesign)

Joe Sak

Joe was Neoteric's lead developer. When not working, he's playing the drums, spending quality time with friends and family, and riding his bike.

« **OLDER**

The Joy of Cooking - Whip up a Rails environment with Chef (/articles/2012/4/the-joy-of-cooking-whip-up-a-rails-environment-with-chef)

**NEWER** »

Using Jasmine with the asset pipeline (/articles/2012/4/using-jasmine-with-the-asset-pipeline)