If we’re going to talk about the transition from HTTP/1.1 to HTTP/2, it’s worth taking a quick trip back to the flannel-soaked 1990s. Back then, HTTP was a simple protocol for a simple job – transporting documents from a web server to your computer as efficiently as possible. This is because the web was simple, made up of documents and a spattering of images that were no sweat for the hardware of the time to sling around.

HTTP, which stands for **Hypertext Transfer Protocol**, saw wide adoption in the early 1990s to facilitate this, transferring “hypertext” documents. These were simple documents comprising mostly text and a few images, marked up with HTML to add formatting and links to other hypertext documents. By 1996, the 1.0 specification was adopted. This version was elegant and simple for the needs of the mid-90s web – make a connection, download a file, close the connection, and repeat for each file needed to display a web page.

By the tail end of the 90s, however, things began to change. The web was rapidly evolving beyond a glorified worldwide Choose Your Own Adventure full of linked documents to click through. People began buying and selling things, necessitating security. In addition to images, people wanted to watch videos and edit documents and send goofy animated greeting cards from their browsers. The strategy of one connection, one file was quite slow for these new use-cases.

So, in 1999, the w3 overseeing body updated the HTTP spec to version 1.1. This version was a bit of a stopgap, adding features to the 1.0 specification meant to improve the speed and efficiency of web servers powering this new “hypermedia” version of the web.

And then, the innovation kind of… stalled. As I’m sure you’re aware, the web changed more between 1999 and 2015 than it did between 1996 and 1999, but for those 16 years, we lived HTTP/1.1 specification, shoehorning more and more modern applications into the aging specification.

The HTTP/1.1 To HTTP/2 Transition

The breaking point in HTTP/1.1 was reached well before the 2015 introduction of HTTP/2. In fact, Google was working on its own replacement for HTTP/1.1 since the early 2010s, called SPDY (pronounced “speedy”). This protocol used the existing infrastructure built for HTTP/1.1, but modified how the requests worked over the infrastructure. SPDY used multiplexing to download multiple resources efficiently over a single connection and could be “back-ported” to existing applications with a translation layer.

It makes sense that Google would take the lead on this, as they had been developing increasingly complex web-based applications that operated more like desktop applications than websites, like Gmail and Google Apps. In fact, the SPDY protocol was so well-designed, W3 used it as the basis for HTTP/2.

So, in 2015, w3 officially adopted the HTTP/2 specification based on SPDY, and all major browsers began supporting the protocol.

Why HTTP/2 Is Faster Than HTTP/1.1

Today, the concept of “webpages” is anachronistic, in much the same way as “videotaping” something with your smartphone. Modern websites function much more like applications, with a constant two-way stream of data being an essential part of their functionality.

For example, when you’re typing something in a Google Doc, as I am typing this article right now, every keystroke sends data to Google’s servers. Google’s servers process that data, and then send updates back to your browser with the text you’ve typed, along with other helpful information like suggestions, the last edit status of the document, and much more. Over HTTP/1.1, each of your keypresses would initiate a new connection to the server, to send each character you typed over the wire. Then, your browser would have to constantly “ping” Google’s server to see if the status of the document changed, to add the character to the screen you’re looking at. That’s a boatload of connections, and each one takes precious time.

With HTTP/2 though, it’s essentially a constant two-way stream over a single connection. Google’s server is always listening for data coming from your browser, and your browser is always listening for data to come back from Google. There’s no more send data, wait for response, update the screen, send more data, wait for a response, etc. Instead, everything happens in real-time. In this way, a web “page” like a Google Doc can update itself so frequently as to feel like a native application on your computer.

These are the high-level differences between HTTP1 and HTTP2:

* HTTP2 is binary, instead of textual
* HTTP2 is fully multiplexed, instead of ordered and blocking
* HTTP2 can, therefore, use one connection for parallelism
* HTP2 uses header compression to reduce overhead
* HTTP2 allows servers to “push” responses proactively into client caches