Multiplexing: TCP vs HTTP2

Lori MacVittie, 2015-17-09

Can you use both? Of course you can! Here comes the (computer) science…

One of the big performance benefits of moving to HTTP/2 comes from its extensive use of multiplexing. For the uninitiated, multiplexing is the practice of reusing a single TCP connection for multiple HTTP requests and responses. See, in the old days (HTTP/1), a request/response pair required its own special TCP connection. That ultimately resulted in the TCP connection per host limits imposed on browsers and, because web sites today are comprised of an average of 86 or more individual objects each needing its own request/response, slowed down transfers. HTTP/1.1 let us use "persistent" HTTP connections, which was the emergence of multiplexing (connections could be reused) but constrained by the synchronous (in order) requirement of HTTP itself. So you'd open 6 or 7 or 8 connections and then reuse them to get those 80+ objects.

With HTTP/2 that's no longer the case. A single TCP connection is all that's required because HTTP/2 leverages multiplexing and allows asynchronous (parallel) requests. Many request/response pairs can be transferred over that single connection in parallel, resulting in faster transfers and less networking overhead. Because as we all know by know, TCP's three-way handshake and windowing mechanisms (slow start, anyone?) can be a drag (literally) on app performance.

So the question is, now that we've got HTTP/2 and its multiplexing capabilities on the client side of the equation, do we still see a benefit from TCP multiplexing on the server side of the equation?

Yes. Absolutely.

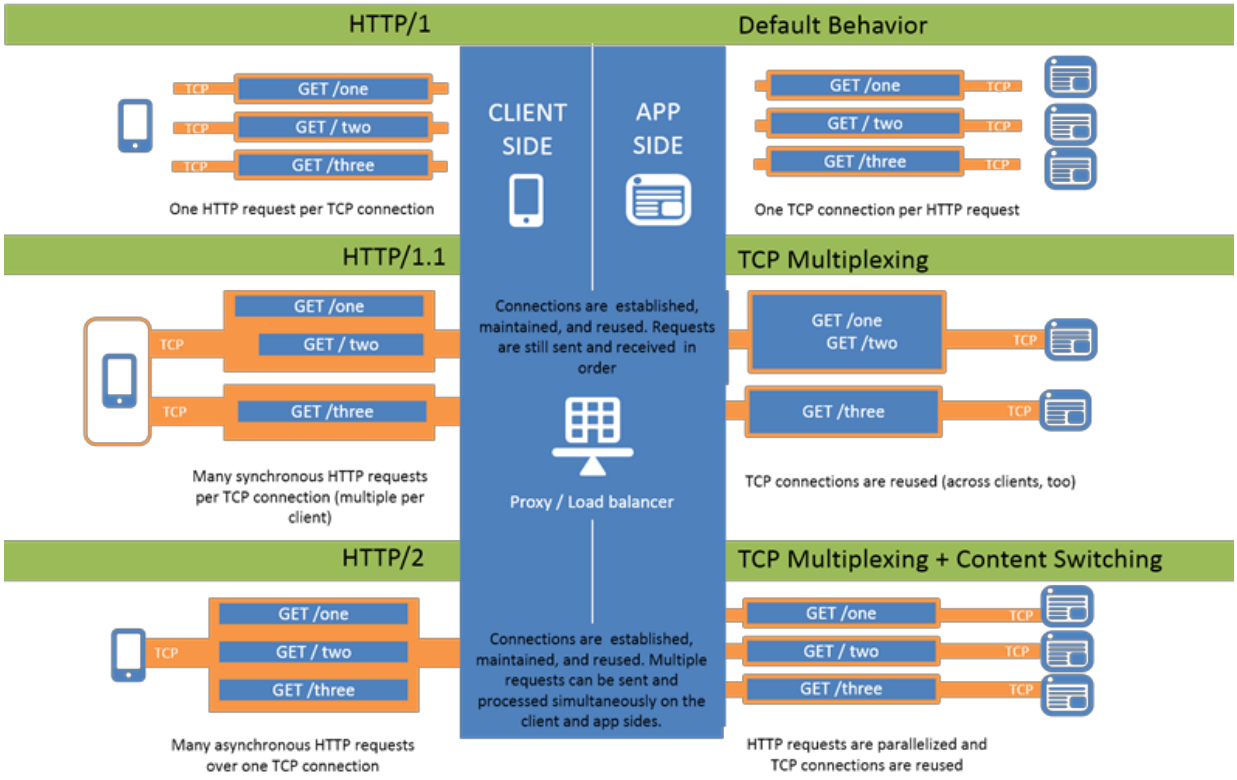
The reason for that is that is operational and directly related to a pretty traditional transition that has to occur whenever there's a significant "upgrade" to what is a foundational protocol like HTTP. Remember, IPv6 has been available and ready to go for a decade and we're still not fully transitioned. Think about that for a minute when you consider how long the adoption curve for HTTP/2 is probably going to be.

Part of the reason for this is while many browsers already support HTTP/2, very few organizations have web or app servers that support HTTP/2. That means that while they could support HTTP on the client side, they can't on the server side. Assuming the server side ***can*** support HTTP/2, there are then business and architectural reasons why an organization might choose to delay migration – including licensing, support, and just the cost of the disruption to upgrade.

So HTTP2 winds up being a no-go. Orgs don't move to HTTP/2 on the client side even though it has significant performance benefits, especially for their increasingly mobile app user population because they can't support it on the server side. But HTTP2 gateways (proxies able to support HTTP/2 on the client side and HTTP/1 on the server side) exist. So it's a viable and less disruptive means of migrating to HTTP/2 on the client without having to go "all in" on the server side.

But of course that means you're only getting half the benefits of multiplexing associated with HTTP/2. Unless, of course, you're using TCP multiplexing on the server side.

What multiplexing offers for clients with HTTP/2, TCP multiplexing capabilities in load balancers and proxies offer for servers with HTTP/1. This is not a new capability. It's been a core TCP optimization technique for, well, a long time and it's heavily used to improve both performance and reduce load on web/app servers (which means they have greater capacity, operate more efficiently, and improve the economy of scale of any app).



On the server side, TCP multiplexing opens (and maintains) a TCP connection to each of the web/app servers it is virtualizing. When requests come in from clients the requests are sent by the load balancer or proxy over an existing (open) connection to the appropriate app instance. That means the performance of the app is improved by the time required to open and ramp up a TCP connection. It also means that the intermediary (the load balancer or proxy) can take in multiple HTTP requests and effectively parallelize them (we call this content switching). In the world of HTTP/1, that means if the client opened six TCP connections and then sent 6 different HTTP requests, the intermediary could ostensibly send out all 6 over existing TCP connections to the appropriate web/app servers, thereby speeding up the responses and improving overall app performance.

The same thing is true for HTTP/2. The difference is that with HTTP/2 those 6 different requests are coming in over the same TCP connection. But they're still coming in. That means a TCP multiplexing-capable load balancer (or proxy) can parallelize those requests to the web/app servers and achieve gains in performance that are noticeable (in a good way) to the client.

True, that gain may be measured in less than a second for most apps, but that means the user is receiving data faster. And when users expect responses (like the whole page) in less than 3 seconds. Or 5 depending on whose study you're looking at. The father of user interface design, Jakob Nielsen, noted that users will notice a 1 second delay. And that was in 1993. I'm pretty sure my 7 year old notices sub-second delays – and is frustrated by them.

The point being that every micro-second you can shave off the delivery process (receiving a request and sending a response) is going to improve engagement with users – both consumer and corporate. What HTTP/2 effectively does is provide similar TCP optimizations on the ***client side*** of the equation as TCP multiplexing offers on the ***server side.*** Thus, using both HTTP/2 and network-based TCP multiplexing is going to offer a bigger gain in performance than using either one alone. And if you couple HTTP/2 and TCP multiplexing with content switching, well.. you're going to gain some more.

So yes, go ahead. Multiplex on the app and the client side and reap the performance benefits.

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc.	F5 Networks	F5 Networks Ltd.	F5 Networks
Corporate Headquarters	Asia-Pacific	Europe/Middle-East/Africa	Japan K.K.
info@f5.com	apacinfo@f5.com	emeainfo@f5.com	f5j-info@f5.com