# **The Difference: HTTP/1.1 vs HTTP/2.**

# 

**Introduction:**

The breaking point in HTTP/1.1 was reached well before the 2015 introduction of HTTP/2. In fact, Google was waiting 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 tha Gmailsites, 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.

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

# **Disadvantages of HTTP/1.1:** Previously, [HTTP/1.1](https://en.wikipedia.org/wiki/HTTP/2#Differences_from_HTTP_1.1) was the major versithe on of the HTTP network protocol used by the World Wide Web, implemented across clients and servers

That worked well for 15 years. Modern-day applications and websites evolved and the amount of data to be loaded on a single page increased, the shortcomings of HTTP/1.1 became more prominent.

## **One Open Request Per Connection:** HTTP/1.1 practically allows only one outstanding request per TCP connection (though [HTTP pipelining](https://en.wikipedia.org/wiki/HTTP_pipelining) allows more than one outstanding request, it still doesn’t solve the problem completely).

The browsers, to circumvent this limit, implement multiple parallel TCP connections to every domain (the number of parallel connections varied per browser). But this head-of-line blocking nature of HTTP/1.1 is a major bottleneck for faster loading applications.

## **Duplication Of Data**

The other problem with HTTP/1.1 is the duplication of data across requests (cookies and other headers). Too many remeans means too much redundant data, which would impact performance.

This led to the development of techniques like image sprites (combining multiple image requests into a single one) and domain-sharding (splitting the requests for resources over multiple domains to increase the number of possible parallel TCP connections).

Advantages Of HTTP/2

[HTTP/2](https://http2.github.io/faq/) was built over Google’s SPDY protocol with the above shortcomings of HTTP/1.1 kept in mind. The main advantages of HTTP/2 over HTTP/1.1 as pulled from their Github page

### Multiplexed, instead of ordered

Allows using the same TCP connection for multiple parallel requests

## **Header compression using HPACK**

Compressed headers, reduced data redundancy

## **Server Push:** Instead of waiting for the client to request for assets like JS and CSS, the server can “push” the resources it believes would be required by the client. Avoids the round trip.

| **HTTP/1.1** | **HTTP/2** |
| --- | --- |
| I then usest works on the textual format. | It works on the binary protocol. |
| There is head-of-line blocking that blocks all the requests behind it until it gets all its resources. | It allows multiplexing so one TCP connection is required for multiple requests. |
| It uses requests resource Inlining for use getting multiple pages | It uses a PUSH frame by the server that collects all multiple pages |

To contextualize the specific changes that HTTP/2 made to HTTP/1.1, let’s first take a high-level look at the historical development and basic workings of each.

