**Difference between HTTP1.1 vs HTTP2**

Hypertext Transfer Protocol (HTTP) is an application protocol that is, currently, the foundation of data communication for the World Wide Web.

The first usable version of Introduction of HTTP was created in 1997. Because it went through several stages of development, this first version of HTTP was called HTTP/1.1. This version is still in use on the web. In 2015, a new version of HTTP called HTTP2 was created.

## High level Differences between HTTP/2 and HTTP/1.1:

* HTTP2 is binary, instead of textual
* HTTP2 is fully multiplexed, instead of ordered and blocking
* 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 version of HTTP network protocol used by the World Wide Web, implemented across clients and servers.

That worked well for 15 years. But as 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 requests 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****

## ****Binary P****rotocol **:**

HTTP1.1 used to process text commands to complete request-response cycles. HTTP/2 will use binary commands (in 1s and 0s) to execute the same tasks. This attribute eases complications with framing and simplifies implementation of commands that were confusingly intermixed due to commands containing text and optional spaces.

## Multiplexing :

HTTP/1.1 loads resources one after the other, so if one resource cannot be loaded, it blocks all the other resources behind it. In contrast, HTTP/2 is able to use a single TCP connection to send multiple streams of data at once so that no one resource blocks any other resource. HTTP/2 does this by splitting data into binary-code messages and numbering these messages so that the client knows which stream each binary message belongs to.

## Header compression:

Small files load more quickly than large ones. To speed up web performance, both HTTP/1.1 and HTTP/2 compress HTTP messages to make them smaller. However, HTTP/2 uses a more advanced compression method called HPACK that eliminates redundant information in HTTP header packets. This eliminates a few bytes from every HTTP packet. Given the volume of HTTP packets involved in loading even a single webpage, those bytes add up quickly, resulting in faster loading.

## Server Push:

Typically, a server only serves content to a client device if the client asks for it. However, this approach is not always practical for modern webpages, which often involve several dozen separate resources that the client must request. Server Push capability allows the server to send additional cacheable information to the client that isn’t requested but is anticipated in future requests. For example, if the client requests for the resource X and it is understood that the resource Y is referenced with the requested file, the server can choose to push Y along with X instead of waiting for an appropriate client request.

Take a look at [ImageKit.io’s demo page](http://imagekit.io/demo/http2-vs-http1?utm_source=blog&utm_medium=blog&utm_campaign=Blog" \t "_blank) to understand the performance boost because of HTTP/2.