**Write a blog on Difference between HTTP1.1 vs HTTP2**

HTTP (Hypertext Transfer Protocol) is an application protocol that has been the de facto standard for communication on the World Wide Web since its invention in 1989. HTTP/1.1 was released in 1997 and has been the primary version of the protocol used on the web until recently. In 2015, a new version called HTTP/2 was introduced, which offered several methods to decrease latency, especially when dealing with mobile platforms and server-intensive graphics and videos. [HTTP/2 has since become increasingly popular, with some estimates suggesting that around a third of all websites in the world support it.](https://www.digitalocean.com/community/tutorials/http-1-1-vs-http-2-what-s-the-difference)

**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

|  |  |
| --- | --- |
| HTTP1.1 | HTTP2 |
| **Binary framing layer:**  As opposed to HTTP/1.1, which keeps all requests and responses in plain text format | **Binary framing layer:**  HTTP/2 uses the binary framing layer to encapsulate all messages in binary format, while still maintaining HTTP semantics, such as verbs, methods, and headers. |
| **Multiplexing:** [This is in contrast to HTTP/1.1, which is ordered and blocking, meaning that requests must be sent and received in order, and each request must be completed before the next one can be sent.](https://www.digitalocean.com/community/tutorials/http-1-1-vs-http-2-what-s-the-difference) | **Multiplexing**:  HTTP/2 is fully multiplexed, meaning that multiple requests can be sent and received at the same time on a single connection. |
| **Header compression**:  HTTP/1.1, headers are sent with each request and response, which can add up to a significant amount of data. | **Header compression**:  HTTP/2 uses header compression to reduce overhead. [HTTP/2 compresses headers, which reduces the amount of data that needs to be sent over the network](https://www.digitalocean.com/community/tutorials/http-1-1-vs-http-2-what-s-the-difference). |
| **SERVER PUSH:**  HTTP/2 not allow the server to push resources to the client | **SERVER PUSH:**  HTTP/2 allows the server to push resources to the client before the client requests them, which can improve page load times. |

* In summary, HTTP/2 is a more efficient and faster version of the HTTP protocol than HTTP/1.1. It uses binary framing, multiplexing, header compression, and server push to reduce latency and improve performance. [However, it’s important to note that not all web servers and browsers support HTTP/2, so it’s important to check compatibility before implementing it on your website](https://www.digitalocean.com/community/tutorials/http-1-1-vs-http-2-what-s-the-difference)

**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**

[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 are -

**Multiplexed, instead of ordered**

Allows using 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/2 is the clear winner. Once the first few assets start loading over HTTP/2, the following assets are loaded very quickly. This is not the case with HTTP/1.1, where the image assets keep loading for a longer time one after another (typical to pipelining in HTTP/1.1) to complete the full image.