HTTP stands for hypertext transfer protocol & it is used in client-server communication. By using HTTP user sends the request to the server & the server sends the response to the user. There are several stages of development of HTTP but mainly using HTTP/1.1 which was created in 1997 & the new one is HTTP/2 which was created in 2015.

**HTTP/1.1:** When we make a request to the server for the any(eg: Google.com) page & server responds to us as a resource of given(eg:Google.com) page. before sending the request and the response there is a TCP connection established between client & server. again, we make a request to the server for image img.jpg & the server gives a response as an image img.jpg. the connection was not lost here after the first request because we add a keep-alive header which is the part of the request so there is an open connection between the server & client. there is a persistent connection which means several requests & responses are merged in a single connection. These are the drawbacks that lead to the creation of HTTP/2: The first problem is HTTP/1.1 transfer all the requests & responses in the plain text message form. The second one is head of line blocking in which TCP connection is blocked all other requests until the response does not receive. all the information related to the header file is repeated in every request.

**HTTP/2:** HTTP/2 was developed over the SPDY protocol. HTTP/2 works on the binary framing layer instead of textual that converts all the messages in binary format. it works on fully multiplexed that is one TCP connection is used for multiple requests. HTTP/2 uses HPACK which is used to split data from header. it compresses the header. The server sends all the other files like CSS & JS without the request of the client using the PUSH frame.

**Difference between two protocols:**

**Protocol Basics:**

* HTTP 1.1: The Hypertext Transfer Protocol (HTTP) 1.1 is the older version and has been the dominant protocol for web communication for many years. It operates on a request-response model, where each request from the client typically requires its own connection to the server.
* HTTP/2: HTTP/2 is the newer version of the protocol, designed to improve upon the limitations of HTTP 1.1. It introduces several optimizations and changes to enhance performance, especially for modern web applications.

**Multiplexing:**

* HTTP 1.1: In HTTP 1.1, each request-response cycle typically requires its own connection. This can lead to inefficiencies, especially when multiple resources need to be fetched for a single webpage.
* HTTP/2: HTTP/2 supports multiplexing, which allows multiple requests and responses to be sent and received over a single connection simultaneously. This reduces latency and improves overall performance, especially for complex web pages with many resources.

**Header Compression:**

* HTTP 1.1: Headers in HTTP 1.1 are often verbose and repetitive, leading to increased overhead, especially for requests with small payloads.
* HTTP/2: HTTP/2 introduces header compression, which significantly reduces the overhead associated with sending headers. This optimization improves efficiency, especially for requests with small payloads or frequent connections.

**Server Push:**

* HTTP 1.1: In HTTP 1.1, the server cannot push resources to the client proactively. Instead, the client must request each resource individually.
* HTTP/2: HTTP/2 supports server push, allowing the server to send additional resources to the client before they are requested. This can further reduce latency and improve page load times, especially for resources that the server knows the client will need.

**Security:**

* Both HTTP 1.1 and HTTP/2 can be used over secure connections (HTTPS), which encrypts data transmitted between the client and server. However, HTTP/2 encourages the use of HTTPS by offering several performance benefits specifically designed for secure connections.

|  |  |
| --- | --- |
| **HTTP/1.1** | **HTTP/2** |
| It 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 doesn’t get its all 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 PUSH frame by server that collects all multiple pages |
| It compresses data by itself. | It uses HPACK for data compression. |