**HTTP:**

HTTP stands for hypertext transfer protocol, and it is the basis for almost all web applications. More specifically, HTTP is the method computers and servers use to request and send information. HTTP is based on the Client/Server model. Client/Server model can be explained as two computers, Client (receiver of service) and Server (provider of service) that are communicating via requests and responses. For instance, when someone navigates to guvi.com on their laptop, their web browser sends an HTTP request to the guvi servers for the content that appears on the page. Then, guvi servers send HTTP responses with the text, images, and formatting that the browser displays to the user.

**HTTP versions:**

The first usable version 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 **HTTP/2** was created.

**HTTP/1.1:**

HTTP/1.1 takes uses persistent connections and pipelining. With persistent connections, HTTP/1.1 assumes that a TCP (Transmission Control Protocol) connection should be kept open unless directly told to close. This allows the client to send multiple requests along the same connection without waiting for a response to each, greatly improving the performance. Unfortunately, there is a natural bottleneck to this optimization strategy. Since multiple data packets cannot pass each other when traveling to the same destination, there are situations in which a request at the head of the queue that cannot retrieve its required resource will block all the requests behind it. This is known as head-of-line (HOL) blocking, and is a significant problem with optimizing connection efficiency in HTTP/1.1. Adding separate, parallel TCP connections could alleviate this issue, but there are limits to the number of concurrent TCP connections possible between a client and server, and each new connection requires significant resources.

**HTTP/2:**

In HTTP/2, the binary framing layer encodes requests/responses and cuts them up into smaller packets of information, greatly increasing the flexibility of data transfer. Let’s take a closer look at how this works. As opposed to HTTP/1.1, which must make use of multiple TCP connections to lessen the effect of HOL blocking, HTTP/2 establishes a single connection object between the two machines. Within this connection there are multiple streams of data. Each stream consists of multiple messages in the familiar request/response format. Finally, each of these messages split into smaller units called frames. At the most granular level, the communication channel consists of a bunch of binary-encoded frames, each tagged to a particular stream. The identifying tags allow the connection to interleave these frames during transfer and reassemble them at the other end. The interleaved requests and responses can run in parallel without blocking the messages behind them, a process called multiplexing. Multiplexing resolves the head-of-line blocking issue in HTTP/1.1 by ensuring that no message has to wait for another to finish. This also means that servers and clients can send concurrent requests and responses, allowing for greater control and more efficient connection management.

**Advantages of** **HTTP/2:**

* Protocol negotiation mechanism — protocol electing, eg. HTTP/1.1, HTTP/2 or other.
* High-level compatibility with HTTP/1.1 — methods, status codes, URIs and header fields.
* Page load speed improvements trough:
* Compression of request headers
* Binary protocol
* HTTP/2 Server Push
* Request multiplexing over a single TCP connection
* Request pipelining
* HOL blocking (Head-of-line) — Package blocking

**Difference between HTTP1.1 vs HTTP2**

* HTTP2 is binary, where as HTTP1 is 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.
* HTTP2 is secured by default.