HTTP1.1	HTTP2
1.requests and responses needs to be transferred between both these machines until the client receives all the resources, essential to load a web page on the end-user's (your) screen.	1.HTTP/2 uses a binary framing layer. This layer encapsulates messages – converted to its binary equivalent – while making sure that its HTTP semantics (method details, header information, etc.) remain untamed. This feature of HTTP/2 enables gRPC to use lesser resources.
2.It uses a single connection for each request. This means that multiple requests from the same client may need to wait for previous requests to complete before they can be processed.	2.It supports multiplexing, allowing multiple requests and responses to be multiplexed over a single connection. This enables parallel processing of requests and responses, reducing latency and improving overall performance.
3.Headers are sent with each request and response, and they are not compressed. This can result in a significant amount of redundant data being transmitted, especially for multiple requests to the same server.	3.It uses header compression, which reduces the overhead of redundant header data. This helps in improving efficiency and reducing the amount of data transmitted over the network.
4.HTTP/1.1 addresses this problem by creating a persistent connection between server and client. Until explicitly closed, this connection will remain open. So, the client can use one TCP connection throughout the communication sans interrupting it again and again.	4.Considering the bottleneck in the previous scenario, the HTTP/2 developers introduced a binary framing layer. This layer partitions requests and responses in tiny data packets and encodes them. Due to this, multiple requests and responses become able to run parallelly with HTTP/2 and chances of HOL blocking are bleak.
5.It uses a text-based protocol, which is human-readable but can be less efficient for machines to process.	5.It is a binary protocol, which is more efficient for machines to parse and requires less bandwidth compared to the text-based format of HTTP/1.1.
6.There is no built-in mechanism for prioritizing requests. All requests are treated with the same priority.	6.It introduces a priority mechanism, allowing clients to assign priorities to different requests. This helps in optimizing the order in which resources are loaded, improving page load times.
7.Clients must explicitly request each resource they need, leading to suboptimal performance in certain situations.	7.It supports server push, where the server can push resources to the client before they are explicitly requested. This can result in faster page loads by reducing the need for additional round trips.
8.Each request/response pair requires a separate connection, which can lead to higher latency.	8.It allows multiple requests and responses to be multiplexed over a single connection, reducing the overhead associated with establishing multiple connections.

9.multiple TCP Connection used for every client request	9.HTTP/2 supports full multiplexing for requests as well as responses over a single TCP connection. Due to these capabilities, lower page load times are achieved by removing needless latency and improving the overall capacity of network alongside its availability.
10.HTTP/1.x uses formats like gzip to compress the data transferred in the messages.	10.To deal with this bottleneck, HTTP/2 uses HPACK compression to decrease the average size of the header. This compression program encodes the header metadata using Huffman coding, which significantly reduces its size as a result.