* The Hypertext Transfer Protocol (HTTP) is the foundation of the World Wide Web, and is used to load webpages using hypertext links.
* HTTP is an application layer protocol designed to transfer information between networked devices and runs on top of other layers of the network protocol stack.
* A typical flow over HTTP involves a client machine making a request to a server, which then sends a response message.

**EVOLUTION OF HTTP**

* **HTTP/0.9** – The one-line protocol
  + GET /mypage.html
* **HTTP/1.0** – Building extensibility
  + HTTP headers
* **HTTP/1.1** – The standardized protocol
* More than 15 years of extensions
* **HTTP/2** – A protocol for greater performance
* **HTTP/3** - HTTP over QUIC

**DIFFERENCE BETWEEN HTTP 1 AND HTTP 2**

**HTTP/1.1**

HTTP protocol was developed in 1989 as the common language that enables client and server machines’ interaction. Process steps are as enlisted:

1. The client (browser) has to send a request to the server using the method (GET/POST).
2. Server responds with the requested resource, for example – image, alongside the status of what it did to the client’s request.

**HTTP/2**

**a.Introduction**

* HTTP/2 was released at Google as the significant improvement of its predecessor. It was initially modeled after the SPDY protocol and went through significant changes to include features like multiplexing, header compression, and stream prioritization to minimize page load latency. After its release, Google announced that it would not provide support for SPDY in favor of HTTP/2.
* The major feature that differentiates HTTP/2 from HTTP/1.1 is the binary framing layer. Unlike HTTP/1.1, 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.

**b.Delivery models**

**HTTP/1.1**

* 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.
* This approach surely ensures good performance, but it also is problematic.

**HTTP/2**

* 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.
* Not only has it solved the HOL blocking problem in HTTP/1.1, but it also concurrent message exchange between the client and the server. This way, both of them can have more control while the connection management quality is boosted too.

**c.status code**

**HTTP/1**

Can define 16 status codes; the error prompt is not specific enough.

**HTTP/2**

Introduces a warning header field to carry additional information about the status of a message. Can define 24 status codes, error reporting is quicker and more efficient.

**D.web traffic**

* HTTP/1.1 provides faster delivery of web pages and reduces web traffic as compared to HTTP/1.0. However, TCP starts slowly and with domain sharding (resources can be downloaded simultaneously by using multiple domains), connection reuse and pipelining, there is an increased risk of network congestion.

|  |  |
| --- | --- |
|  |  |

* HTTP/2 utilizes multiplexing and server push to effectively reduce the page load time by a greater margin along with being less sensitive to network delays.