**Hypertext transfer protocol(HTTP**

**INTRODUCTION:**

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. This is the foundation for data communication for the World Wide Web (i.e. internet) since 1990. HTTP is a generic and stateless protocol which can be used for other purposes as well using extensions of its request methods, error codes, and headers.

Basically, HTTP is a TCP/IP based communication protocol, that is used to deliver data (HTML files, image files, query results, etc.) on the World Wide Web. The default port is TCP 80, but other ports can be used as well. It provides a standardized way for computers to communicate with each other. HTTP specification specifies how clients' request data will be constructed and sent to the server, and how the servers respond to these requests.

**HISTORY:**

HTTP (HyperText Transfer Protocol) is the underlying protocol of the World Wide Web. Developed by Tim Berners-Lee and his team between 1989-1991, HTTP has seen many changes, keeping most of the simplicity and further shaping its flexibility.

In 1989, while he was working at CERN, Tim Berners-Lee wrote a proposal to build a hypertext system over the Internet. Initially calling it the Mesh, it was later renamed to World Wide Web during its implementation in 1990. Built over the existing TCP and IP protocols, it consisted of 4 building blocks:

* A textual format to represent hypertext documents, the HyperText Markup Language (HTML).
* A simple protocol to exchange these documents, the HypertText Transfer Protocol (HTTP).
* A client to display (and accidentally edit) these documents, the first Web browser called WorldWideWeb.
* A server to give access to the document, an early version of httpd.

HTTP/0.9 – The one-line protocol:

The initial version of HTTP had no version number; it has been later called 0.9 to differentiate it from the later versions. HTTP/0.9 is extremely simple: requests consist of a single line and start with the only possible method GET followed by the path to the resource (not the URL as both the protocol, server, and port are unnecessary once connected to the server).

GET /mypage.html

Unlike subsequent evolutions, there were no HTTP headers, meaning that only HTML files could be transmitted, but no other type of documents.

HTTP/1.0 – Building extensibility:

HTTP/0.9 was very limited and both browsers and servers quickly extended it to be more versatile:

* Versioning information is now sent within each request (HTTP/1.0 is appended to the GET line)
* A status code line is also sent at the beginning of the response, allowing the browser itself to understand the success or failure of the request and to adapt its behavior in consequence (like in updating or using its local cache in a specific way)
* The notion of HTTP headers has been introduced, both for the requests and the responses, allowing metadata to be transmitted and making the protocol extremely flexible and extensible.
* With the help of the new HTTP headers, the ability to transmit other documents than plain HTML files has been added

HTTP/1.1 – The standardized protocol:

In parallel to the somewhat chaotic use of the diverse implementations of HTTP/1.0, and since 1995, well before the publication of HTTP/1.0 document the next year, proper standardization was in progress. The first standardized version of HTTP, HTTP/1.1 was published in early 1997, only a few months after HTTP/1.0.

HTTP/1.1 clarified ambiguities and introduced numerous improvements:

* A connection can be reused, saving the time to reopen it numerous times to display the resources embedded into the single original document retrieved.
* Pipelining has been added, allowing to send a second request before the answer for the first one is fully transmitted, lowering the latency of the communication.
* Chunked responses are now also supported.
* Additional cache control mechanisms have been introduced.
* Content negotiation, including language, encoding, or type, has been introduced, and allows a client and a server to agree on the most adequate content to exchange.
* Thanks to the Host header, the ability to host different domains at the same IP address now allows server colocation.

HTTP/2 – A protocol for greater performance:

Over the years, Web pages have become much more complex, even becoming applications in their own right. The amount of visual media displayed, the volume and size of scripts adding interactivity, has also increased: much more data is transmitted over significantly more HTTP requests. HTTP/1.1 connections need requests sent in the correct order.

The HTTP/2 protocol has several prime differences from the HTTP/1.1 version:

* It is a binary protocol rather than text. It can no longer be read and created manually. Despite this hurdle, improved optimization techniques can now be implemented.
* It is a multiplexed protocol. Parallel requests can be handled over the same connection, removing the order and blocking constraints of the HTTP/1.x protocol.
* It compresses headers. As these are often similar among a set of requests, this removes duplication and overhead of data transmitted.
* It allows a server to populate data in a client cache, in advance of it being required, through a mechanism called the server push.

**WORKING:**

As a request-response protocol, HTTP gives users a way to interact with web resources such as HTML files by transmitting hypertext messages between clients and servers. HTTP clients generally use Transmission Control Protocol (TCP) connections to communicate with servers.

The client initiates a transaction by sending a request message to the server. The server replies to the request message by sending a response message.

HTTP messages are of two types: request and response. Both the message types follow the same message format.

**Request Message**: The request message is sent by the client that consists of a request line, headers, and sometimes a body.

**Response Message**: The response message is sent by the server to the client that consists of a status line, headers, and sometimes a body.



Request-Line:

The Request-Line begins with a method token, followed by the Request-URI and the protocol version, and ending with CRLF. The elements are separated by space SP characters.

Request-Line = Method SP Request-URI SP HTTP-Version CRLF

Request Method:

The request method indicates the method to be performed on the resource identified by the given Request-URI. The method is case-sensitive and should always be mentioned in uppercase.

**GET**-The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data.

**HEAD**-Same as GET, but it transfers the status line and the header section only.

**POST**-A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms.

**PUT**-Replaces all the current representations of the target resource with the uploaded content.

**DELETE**-Removes all the current representations of the target resource given by URI.

**CONNECT**-Establishes a tunnel to the server identified by a given URI.

**OPTIONS**-Describe the communication options for the target resource.

**TRACE**-Performs a message loop back test along with the path to the target resource.

**Uniform Resource Locator (URL):**

A client that wants to access the document in an internet needs an address and to facilitate the access of documents, the HTTP uses the concept of Uniform Resource Locator (URL).The Uniform Resource Locator (URL) is a standard way of specifying any kind of information on the internet.The URL defines four parts: method, host computer, port, and path.



**Method:** The method is the protocol used to retrieve the document from a server. For example, HTTP.

**Host:** The host is the computer where the information is stored, and the computer is given an alias name. Web pages are mainly stored in the computers and the computers are given an alias name that begins with the characters "www". This field is not mandatory.

**Port:** The URL can also contain the port number of the server, but it's an optional field. If the port number is included, then it must come between the host and path and it should be separated from the host by a colon.

**Path:** Path is the pathname of the file where the information is stored. The path itself contain slashes that separate the directories from the subdirectories and files.

**CURRENT STATUS:**

HTTP/3 is the next protocol for network communication across the Web, which is meant to partially replace HTTP/1 and HTTP/2. One month before the next QUIC Working Group meeting, to be held in Zurich next February, it may be useful to recap what HTTP/3 promises and what its current client/server support looks like.

HTTP/3 promises to make Internet connections faster, more reliable, and more secure. Born as "HTTP over QUIC", an effort to adapt the HTTP protocol to run on top of Google's own transport layer protocol, QUIC, it was later proposed as an IETF standard and it is currently an Internet Draft. In October 2018, IETF HTTP & QUIC Working Groups co-chair Mark Nottingham proposed to rename HTTP over QUIC as HTTP/3 to clarify its true nature and its independence from QUIC.

QUIC is a key element of HTTP/3, since it provides the foundations for its main features. Built on top of UDP, QUIC attempts to solve the major issues experienced when using the TCP protocol, i.e., connection-establishment latency and multi-stream handling in the presence of packet loss. As mentioned, HTTP/3 is still being defined by IETF, with no official release date set yet. Meanwhile, adoption of HTTP/3 is growing worldwide, with almost 300,000 services using it across the world.