HTTP/1.1

HTTP/1.1, the first standardized version of HTTP, was introduced in 1997. It presented significant performance optimizations (over HTTP/0.9 and HTTP/1.0) and transformed the way requests and responses were exchanged between clients and servers.

Key Features of HTTP/1.1:

* It was no longer required for each connection to be terminated immediately after every request was served with a response; instead, with the keep-alive header, it was possible to have persistent connections. It allowed multiple requests/responses per TCP connection.
* The Upgrade header was used to indicate a preference from the client that made it possible to switch to a more preferred protocol if found appropriate by the server.
* Other features that reinforced its stability were introduced such as:
  + pipelining (the second request is sent before the response to the first is adequately served)
  + content negotiation (an exchange between client and server to determine the media type, it also provides the provision to serve different versions of a resource at the same URI)
  + cache control (used to specify caching policies in both requests and responses)

HTTP/2

At the beginning of 2010, Google introduced an experimental protocol, SPDY, which supported multiplexing (multiple requests/responses sent and received asynchronously over a single TCP connection) but as it gained traction IETF’s HTTP Working Group came up with HTTP/2 in 2015, which is based on the SPDY protocol.

Key Features of HTTP/2:

* It introduces the concept of a server push where the server anticipates the resources that will be required by the client and pushes them prior to the client making requests. The client retains the authority to deny the server push; however, in most cases, this feature adds a lot of efficiency to the process.
* Introduces the concept of multiplexing that interleaves the requests and responses without head-of-line blocking and does so over a single TCP connection.
* It is a binary protocol i.e. only binary commands in the form of 0s and 1s are transmitted over the wire. The binary framing layer divides the message into frames that are segregated based on their type – Data or Header. This feature greatly increases efficiency in terms of security, compression and multiplexing.
* HTTP/2 uses HPACK header compression algorithm that is resilient to attacks like CRIME and utilizes static Huffman encoding.

HTTP2 Vs. HTTP1.1 –

HTTP/1.1 has been around for more than a decade. With Google’s SPDY leading the way in 2015, the IETF (Internet Engineering Task Force) gave us HTTP/2, which introduces several features to reduce page load times.

HTTP/2 achieves faster webpage loading without performance optimizations that require extensive human efforts in terms of development. It significantly reduces the complexities that had crept into HTTP/1.1 and gives us a robust protocol which, though not without its flaws, will perhaps stand the test of time.

|  |  |  |
| --- | --- | --- |
| Status Code | 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. | Underlying semantics of HTTP such as headers, status codes remains the same. |
| Authentication Mechanism | It is relatively secure since it uses digest authentication, NTLM authentication. | Security concerns from previous versions will continue to be seen in HTTP/2. However, it is better equipped to deal with them due to new TLS features like connection error of type Inadequate Security. |
| Caching | Expands on the caching support by using additional headers like cache-control, conditional headers like If-Match and by using entity tag | HTTP/2 does not change much in terms of caching. With the server push feature if the client finds the resources are already present in the cache, it can cancel the pushed stream |
| Header Compression | Headers are sent on every request leading to a lot of duplicate data being sent uncompressed across the wire. | Header compression is included by default in HTTP/2 using HPACK. |

|  |  |  |
| --- | --- | --- |
| Performance Optimization | Spriting, concatenating, inlining, domain sharding are some of the optimizations used as a workaround to the ‘six connections per host’ rule. | Removes the need for unnecessary optimization hacks. |
| Protocol Type | Text based protocol that is in the readable form. | It is a binary protocol (HTTP requests are sent in the form of 0s and 1s). Needs to be converted back from binary in order to read it. |
| Security | SSL is not required but recommended. Digest authentication used in HTTP1.1 is an improvement over HTTP1.0. HTTPS uses SSL/TLS for secure encrypted communication. | Though security is still not mandatory, it is mostly encrypted (though it is not enforced) since almost all clients require traffic to be encrypted. It also has some minimum standards, such as minimum key size for encryption. TLS 1.2 etc. |