**Task: 1**

**Difference between HTTP/1.1 and HTTP/2?**

|  |  |
| --- | --- |
| For every TCP connection there is only one request and response. But HTTP/1.1 supports connection reuse | Uses multiplexing, over a single TCP connection. It is done using streams and it also provide feature called server push. |
| Introduces a warning header field to carry additional information about status of message (24 status code) and its error reporting is quicker and efficient | Headers and status code of HTTP remains same. |
| It uses digest authentication and NTML authentication | Security concern from previous version will continue to be seen in HTTP/2. However it is better equipped to deal with them |
| It has additional header like cache-control | Does not change much in terms of caching with server push. If the client find resource present already in cache it cancel the push stream |
| Text Based protocol that is in the readable format | It is a binary protocol |

One of the most significant differences between HTTP 1.1 and 2.0

is binary framing layer. This is the part of application layer stack.

HTTP-1.1 keeps all requests and responses in plain text format while HTTP-2.0 uses the binary framing layer to encapsulate all messages in binary format

The conversion of messages into binary allows HTTP/2 to try new approaches to data delivery not available in HTTP/1.1,

What is binary framing layer?

Communication is split into smaller messages and frames, each of which is encoded in binary format.

To Understand new binary framing mechanism which is used to exchanged data between the client and server, we need to get familiarize ourselves with the HTTP/2 terminology:

Stream: A bidirectional flow of bytes within an established connection, which may carry one or more messages.

Message: A complete sequence of frames that map to a logical request or response message.

Frame: The smallest unit of communication in HTTP/2, each containing a frame header, which at a minimum identifies the stream to which the frame belongs.

The relation of these terms is described below:

All communication is performed over a single TCP connection that can carry any number of bidirectional streams.

Each stream has a unique identifier and optional priority information that is used to carry bidirectional messages.

Each message is a logical HTTP message, such as a request, or response, which consists of one or more frames.

The frame is the smallest unit of communication that carries a specific type of data, e.g. HTTP headers, message payload, and so on.

Frames from different streams may be interleaved and then reassembled via the embedded stream identifier in the header of each frame.

Request and response multiplexing

With HTTP/1.1, client uses multiple TCP connections to make multiple parallel requests to improve performance. With HTTP/1.1 only one response can be delivered at a time (response queuing) per connection.

With HTTP/2, new binary framing layer removes these limitations, and enables full request and response multiplexing, by allowing the client and server to break down an HTTP message into independent frames, interleave them, and then reassemble them on the other end.

HTTP/2 no longer needs multiple TCP connections to multiplex streams in parallel as it uses new binary framing mechanism, each stream in HTTP/2 is split into multiple frames, which can be interleaved and prioritized. Hence with HTTP/2 only one connection per origin is required, which offers numerous performance benefits.

Server push

Another powerful new feature of HTTP/2 is the ability of the server to send multiple responses for a single client request. That is, in addition to the response to the original request, the server can push additional resources to the client without the client having to request each one explicitly.

Header compression

Each HTTP transfer carries a set of headers that describe the transferred resource and its properties. In HTTP/1.1, this metadata is always sent as plain text and adds anywhere from 500–800 bytes of overhead per transfer, and sometimes kilobytes more if HTTP cookies are being used. To reduce this overhead and improve performance, HTTP/2 compresses request and response header metadata using the HPACK compression format that uses two simple but powerful techniques.

It allows the transmitted header fields to be encoded via a static Huffman code, which reduces their individual transfer size.

It requires that both the client and server maintain and update an indexed list of previously seen header fields (in other words, it establishes a shared compression context), which is then used as a reference to efficiently encode previously transmitted values.

Source: https://www.digitalocean.com/community/tutorials/http-1-1-vs-http-2-what-s-the-difference#:~:text=As%20opposed%20to%20HTTP%2F1.1,verbs%2C%20methods%2C%20and%20headers.

Source: https://medium.com/walmartglobaltech/introduction-to-http-2-d3e3b4f4d662#:~:text=HTTP%2F2%20introduces%20new%20binary,is%20encoded%20in%20binary%20format.