Difference Between HTTP V1.1 and V2

HTTP V1.1

Connection Handling:

- In HTTP/1.1, each request/response typically requires a separate connection. This can lead to increased latency as multiple connections are needed for parallel requests.

Header Compression:

- HTTP/1.1 doesn't support header compression. Headers are sent with each request and response, and if the headers are large, it can impact performance.

Resource Loading:

- Multiple requests for resources (like images, scripts, stylesheets) are handled sequentially, and browsers often open multiple connections to overcome this limitation. However, it can still result in the "head-of-line blocking" problem.

Multiplexing:

- HTTP/1.1 lacks efficient multiplexing, meaning that multiple requests and responses cannot be sent over a single connection simultaneously. This limitation affects the performance of loading complex web pages with many resources.

Server Push:

- HTTP/1.1 doesn't support server push, a feature where the server can push resources to the client before the client explicitly requests them. This can improve page load times.

HTTP V2:

Multiplexing:

- One of the significant improvements in HTTP/2 is multiplexing, allowing multiple requests and responses to be sent in parallel over a single connection. This reduces latency and improves overall performance.

Header Compression:

- HTTP/2 introduces header compression (HPACK), which significantly reduces overhead by compressing headers before sending them. This helps improve the efficiency of data transfer.

Binary Protocol:

- HTTP/2 is a binary protocol as opposed to the text-based protocol of HTTP/1.1. This makes it more efficient to parse and reduces errors related to whitespace and line endings.

Server Push:

- HTTP/2 supports server push, enabling servers to proactively push resources to the client before the client requests them. This can reduce the need for multiple round-trip requests.

Connection Handling:

- HTTP/2 allows for the reuse of connections for multiple requests and responses, reducing the need to establish new connections for each interaction. This reuse improves efficiency and lowers latency.