**Difference between HTTP1.1 vs HTTP2**

**Introduction:**

In the fast-paced digital world, where milliseconds can make or break user experiences, the underlying protocols governing web communication play a pivotal role. Among these, Hypertext Transfer Protocol (HTTP) has been the cornerstone of data communication over the World Wide Web. As technology evolves, so do the protocols. HTTP/1.1, a long-standing standard, has recently faced a challenger in the form of HTTP/2. In this blog, we delve into the key differences between HTTP/1.1 and HTTP/2, exploring how the latter seeks to address the shortcomings of its predecessor.

**Protocol Multiplexing:**

⦁ HTTP/1.1: In HTTP/1.1, only one request can be sent on a single TCP connection at a time. This limitation often leads to inefficiencies, especially for modern web pages that require numerous resources like images, stylesheets, and scripts.

⦁ HTTP/2: HTTP/2 introduces multiplexing, allowing multiple requests and responses to be interleaved on a single TCP connection. This significantly reduces latency and improves page load times, as resources can be requested and delivered concurrently.

**Header Compression:**

⦁ HTTP/1.1: Every request and response in HTTP/1.1 is accompanied by headers containing metadata about the payload. These headers, though necessary, can be verbose and redundant, especially for subsequent requests.

⦁ HTTP/2: HTTP/2 employs header compression, reducing overhead by compressing headers into a binary format. This optimization reduces the amount of data transmitted, further improving performance, particularly for websites with numerous resources.

**Server Push:**

⦁ HTTP/1.1: In HTTP/1.1, servers can only respond to client requests. If a resource is necessary for rendering a page but not explicitly requested by the client, additional round trips are required to fetch it.

⦁ HTTP/2: HTTP/2 introduces server push, enabling servers to proactively send resources to clients before they are requested. This feature eliminates the need for additional round trips, leading to faster page rendering and improved performance.

**Stream Prioritization:**

⦁ HTTP/1.1: Requests in HTTP/1.1 are processed in the order they are received, without consideration for their relative importance. This can lead to situations where critical resources are delayed behind less essential ones.

⦁ HTTP/2: HTTP/2 allows for stream prioritization, enabling clients to assign weights and dependencies to different resources. This ensures that critical resources are prioritized, optimizing the loading process and enhancing user experience, particularly on bandwidth-constrained networks.

**Binary Protocol:**

⦁ HTTP/1.1: HTTP/1.1 relies on textual representation for messages, which are human-readable but less efficient for parsing and transmission.

⦁ HTTP/2: HTTP/2 switches to a binary protocol, which is more compact and machine-readable. This improves parsing efficiency and reduces error rates, contributing to overall performance gains.

**Conclusion:**

⦁ HTTP/2 represents a significant evolution in web communication protocols, addressing many of the limitations inherent in HTTP/1.1. By introducing features like multiplexing, header compression, server push, stream prioritization, and a binary protocol, HTTP/2 offers substantial improvements in performance, efficiency, and user experience. While the adoption of HTTP/2 continues to grow, it is essential for developers and network administrators to understand its capabilities and leverage them to optimize their web applications for the modern internet landscape.