|  |  |  |
| --- | --- | --- |
|  | **HTTP/1.1** | **HTTP/2** |
| **Multiplexing** | Limited to sequential processing of requests and responses, leading to potential bottlenecks and delays. | Introduces multiplexing, enabling concurrent streams of data over a single connection, significantly improving resource loading times. |
| **Header Compression** | Headers are sent in plain text, leading to redundancy and increased data transfer. | Employs header compression, reducing overhead by transmitting compressed headers, especially beneficial for small requests. |
| **Binary Protocol** | Uses a plain text protocol, which is human-readable but less efficient in terms of parsing and processing. | Adopts a binary protocol for data framing, facilitating more efficient parsing and faster communication between clients and servers. |
| **Connection Optimization** | Limited to a small number of simultaneous connections per domain, potentially slowing down the loading of multiple resources. | Eliminates the need for multiple connections per resource, as multiplexing allows multiple streams on a single connection. |
| **Server Push** | Relies on the client to request each resource individually, leading to additional round-trip delays. | Introduces server push, allowing servers to proactively send resources to clients before they are requested, optimizing page load times. |
| **Stream Prioritization** | Requests are processed in the order they are received, without explicit prioritization. | Supports stream prioritization, enabling the client to specify the importance of different resources, improving the overall user experience. |
| **Error Handling** | Errors may cause the termination of the entire connection, impacting the user experience. | Supports more granular error handling at the stream level, minimizing the impact of errors on other concurrent streams. |
| **Backward Compatibility** | Fully backward compatible with HTTP/1.0, allowing gradual adoption of new features. | Designed to be backward compatible with HTTP/1.1, ensuring a smooth transition for existing applications. |
| **Resource Prioritization** | Resources are fetched in the order they are referenced, without explicit prioritization. | Enables the client to prioritize critical resources, ensuring that important assets are loaded first for a better user experience. |
| **Header Size** | Headers are often large and redundant, contributing to increased latency. | Header compression reduces the size of headers, minimizing overhead and improving the efficiency of data transmission. |