|  |  |  |
| --- | --- | --- |
| **Feature** | **HTTP/1.1** | **HTTP/2** |
| Dependency Tree | Lacks a formalized dependency tree for resource loading. | Introduces a dependency tree structure, optimizing the order of resource loading. |
| Request Prioritization | No built-in mechanism for explicit request prioritization. | Allows for explicit request prioritization, improving the loading of critical resources. |
| Header Requests (Server-Push) | Server-push requires separate connections and has limitations. | Server-push is more efficient and can be multiplexed within the same connection. |
| TCP Slow Start | Slower initial data transfer due to TCP slow start. | Mitigates the impact of TCP slow start, accelerating data transfer. |
| Connection Header | Uses "Connection" header to manage multiple requests. | No need for the "Connection" header as multiplexing handles multiple requests over a single connection. |
| Connection Settings | Limited ability to negotiate and adjust connection settings. | Supports dynamic adjustment of connection settings for optimization. |
| Network Efficiency | Inefficient use of network resources with multiple connections. | Efficient use of network resources through multiplexing and connection reuse. |
| Compression Efficiency | Inefficient content compression due to lack of header compression. | Implements more efficient header compression (HPACK) for reduced bandwidth usage. |
| Server Load | Distributes server load unevenly due to multiple connections. | Balances server load more evenly with a single, multiplexed connection. |
| Resource Size Handling | Struggles with large resource sizes and limited parallelism. | Handles large resource sizes more effectively through parallelism and multiplexing. |
| WebSocket Overhead | Uses separate connections for WebSocket communication. | Shares the same connection for WebSocket communication, reducing overhead. |
| Browser Compatibility | Fully compatible with most browsers but may face limitations. | Enjoying increasing browser support as adoption continues to grow. |
| Header Field Restrictions | Limited flexibility in defining custom header fields. | Provides more flexibility in defining and using custom header fields. |
| Connection Header Use | Uses the "Connection" header for managing persistent connections. | Multiplexing eliminates the need for the "Connection" header in managing multiple requests. |
| Cookie Handling | Sends cookies with every request, contributing to header size. | Implements more efficient cookie handling, reducing redundant cookie transfers. |
| Page Load Time | Generally longer page load times due to sequential resource fetching. | Shortens page load times by allowing for concurrent resource fetching. |
| Standardization | Standardized in 1999 with incremental updates over the years. | Standardized in 2015 with a more modern and extensible architecture. |
| Efficiency for Mobile Networks | Inefficient for mobile networks due to multiple connections. | Optimized for mobile networks, reducing latency and improving performance. |
| Resource Fetching Control | Limited control over resource fetching from the server. | Provides more control with features like stream prioritization and server push. |
| Browser Caching Optimization | May face challenges with browser caching optimization. | Optimizes browser caching through more effective handling of cache-related headers. |