| S.NO | HTTP 1.1 | HTTP2 |
| --- | --- | --- |
| 1. | Operates on a serialised request-response model, handling one request at a time per connection. | Introduces multiplexing, enabling multiple requests and responses to be processed concurrently on a single connection, eliminating the need for multiple connections for parallelism. |
| 2. | Headers are sent in plaintext with each request and response, resulting in redundant data transmission and increased latency. | Implements header compression, reducing the overhead associated with sending headers by compressing redundant information before transmission, leading to improved efficiency and faster load times. |
| 3. | Uses a text-based protocol, which is human-readable but has parsing complexities and larger packet sizes. | Introduces server push, allowing servers to proactively send resources to clients before they are requested, optimising page loading by anticipating needs and reducing latency. |
| 4. | Relies on clients to explicitly request each resource, resulting in additional round trips and increased latency. | Introduces server push, allowing servers to proactively send resources to clients before they are requested, optimising page loading by anticipating needs and reducing latency. |
| 5. | Often requires multiple connections to achieve parallelism, increasing resource consumption and potential bottlenecks. | Supports multiplexing, reducing the need for multiple connections. It also allows connection reuse, minimising overhead associated with establishing and maintaining multiple connections. |
| 6. | Limited by its sequential processing of requests, potentially causing delays and slower page loading times. | Offers improved performance through features like multiplexing, header compression, and server push, resulting in faster and more efficient data transfer. |
| 7. | Lacks explicit prioritisation of requests, leading to potential inefficiencies in resource loading. | Introduces request prioritisation, allowing clients to specify the importance of individual resources, ensuring critical resources are loaded first for a more optimised user experience. |
| 8. | Lacks effective flow control mechanisms, which can lead to overloading in certain situations. | Implements more robust flow control mechanisms, preventing congestion and ensuring efficient data transfer. |
| 9. | Requires a clear-text upgrade to establish secure connections (e.g., transitioning from HTTP to HTTPS). | Supports direct negotiation and doesn't require clear-text upgrade, simplifying the process of transitioning to a secure connection. |
| 10. | Has been the standard for many years and is still widely used. | Adoption has been increasing, and it is becoming the preferred choice for optimising web performance due to its advanced features. |