|  |  |
| --- | --- |
| **HTTP 1.1** | **HTTP 2.0** |
| **Development:** It was developed in the year 1997 | **Development:** It was developed in the year 2015. |
| **Compression:** It compresses data by itself. | **Compression:** It uses HPACK for data compression. |
| **Security:** The client sends a request to a server, and the server sends a response back to the client. | **Security:**  The client sends a request to a server, and the server sends a response back to the client. |
| **Predicting Resource Requests:** HTTP 1.1, the client-server initiates all requests for resources such as images, stylesheets, and scripts. The server can only respond to requests that it receives. This means that the client must first request the HTML of a web page, parse it, and then make additional requests for any additional resources it needs to render the page. This results in delayed page load times. | **Predicting Resource Requests:**  HTTP 2 allows for server push, so the server proactively pushes resources to the client without the client needing to request them. This speeds up page load times as the client starts processing and rendering resources as soon as they are received. |
| Buffer Overflow:  In HTTP 1.1, a buffer overflow can occur when a client sends a request with a header that is larger than the server’s buffer size. This can cause the server to crash or become unresponsive. To prevent buffer overflow, servers typically have a maximum buffer size for incoming requests and reject any requests exceeding this limit. | **Buffer Overflow:**  HTTP 2, on the other hand, uses a more sophisticated approach to prevent buffer overflow. It uses a flow control mechanism that allows the server to send data to the client in small chunks rather than sending all the data at once. |
| **Multiplexing:**  HTTP 1.1 uses a separate connection for each request and response, resulting in increased latency and reduced performance. | Multiplexing:HTTP 2 supports multiplexing, allowing multiple requests and responses to be sent over a single connection simultaneously. This helps to reduce the latency and increase the overall performance of the connection. |
| Binary Protocol:HTTP 1.1 uses plain text to encode and transmit data. Though it is easy for humans to read and understand the data, it can be less efficient than a binary protocol. | Binary Protocol:HTTP 2 uses a series of binary codes to encode and transmit data rather than plain text. Binary protocols are generally more efficient than text-based protocols because they can transmit data more compactly. |
| ****Performance:**** HTTP 1.1 does not include any in-built features, so the performance it delivers is less efficient. | ****Performance:**** HTTP 2 is designed to be more efficient and performant than HTTP 1.1. This is because HTTP 2 includes several features like multiplexing, binary protocol and header compression. |