**DIFFERENCE BETWEEN HTTP 1.1 VS HTTP 2**

1. **HTTP Format :**

HTTP 1.1 : Uses a plain text protocol, which is human-readable but less efficient.

HTTP 2 : Employs a binary protocol, which is more efficient for parsing and processing by computers.

**Example :**

Imagine you’re sending a large file from one computer to another. In HTTP 1.1, the communication would be similar to sending a handwritten letter with detailed instructions on how to assemble the file at the receiving end. This plain text format is easy for humans to understand but less efficient for computers, as they need to spend more time parsing and interpreting the text.

In contrast, HTTP 2’s binary protocol is like sending the file as a pre-packaged, machine-readable puzzle. The receiving computer can quickly and efficiently piece together the data without the need for extensive interpretation, resulting in faster and more efficient data transmission.

1. **Multiplexing**

HTTP 1.1 : It uses a request or response model, where each resource request requires a separate connection. This can lead to inefficiencies and slower page loading times.

HTTP 2 : It supports multiplexing, allowing multiple requests and responses to be sent over a single connection simultaneously. This reduces latency and speeds up page loading.

Example :

Imagine you're loading a modern web page that includes various resources like images, stylesheets, and scripts. In HTTP 1.1, each of these resources would require a separate connection to the server, similar to vehicles waiting in line at a toll booth. This can lead to delays and slower loading times as each resource is fetched one after the other.

Now, with HTTP 2's multiplexing, it's like having a high-speed, multi-lane highway. All those resources can be sent over a single connection simultaneously, just like multiple vehicles moving freely in their respective lanes. This reduces latency and speeds up the loading of the web page, resulting in a faster and more efficient user experience.

1. **Header Compression:**

HTTP 1.1 : Headers are not compressed, and they can be quite large, especially when making multiple requests.

HTTP 2 : It uses header compression, which reduces the overhead of sending headers with each request, resulting in faster data transfer.

Example :

Imagine you’re browsing an online store, and you want to view the details of several products. In HTTP 1.1, when you request information about each product, the headers accompanying each request can be quite large and redundant, similar to repeating your name and address each time you enter a store to buy a product.

With HTTP 2’s header compression, it’s like having a more efficient way to communicate. The redundant information in headers is compressed, reducing the overhead of sending these headers with each request. This results in faster data transfer because there’s less unnecessary information to transmit, making your online shopping experience more responsive and efficient.

1. **Prioritization:**

HTTP 1.1 : Requests are processed in the order they are received, without any inherent prioritization.

HTTP 2 : Supports request prioritization, allowing more critical resources to be loaded first, enhancing user experience. For example, a web page can load text content before images, improving perceived speed.

Example :

Imagine you’re using a messaging app or website where new messages, images, and videos are constantly being sent and received. In HTTP 1.1, these messages would be loaded in the order they arrive, which might lead to a delay in viewing important text messages while waiting for large media files to download.

With HTTP 2’s request prioritization, the messaging app or website can prioritize loading text messages first, ensuring that you can read and respond to important messages immediately, even if large media files are still being fetched in the background. This enhances the user experience by improving the perceived speed and responsiveness of the app.

**5) Server Push :**

HTTP 1.1 : The server cannot proactively push resources to the client; it has to wait for client requests.

HTTP 2 : Introduces server push, where the server can push resources to the client that it predicts the client will need, reducing the need for additional requests.

Example :

Imagine you're visiting an online news website. In HTTP 1.1, the server can't proactively send you resources like images or related articles until you request them. This is like having to ask a librarian for each book you want to read in a library.

With HTTP 2's server push, the website server can predict what you're likely to need next. For instance, when you open an article about a news topic, the server can proactively push related images, videos, and additional articles to your browser without you requesting them individually. This reduces the need for additional requests, making your news reading experience faster and more seamless, similar to a helpful librarian anticipating your reading interests and placing relevant books in front of you.

**6) Connection Handling :**

HTTP 1.1 : Requires multiple connections for parallel downloads, which can consume more server resources.

HTTP 2 : Uses a single connection for multiple parallel requests, reducing the number of connections and resource overhead.

Example :

Imagine you're downloading files from a cloud storage service. In HTTP 1.1, each file you request requires a separate connection to the server, much like having to open a new line at a ticket booth for every item you want to purchase at an event. This can result in a high number of connections, consuming more server resources and potentially slowing down the overall process.

With HTTP 2's single connection approach, it's like having a single efficient line at the ticket booth where you can request and receive multiple files simultaneously. This reduces the number of connections and resource overhead on both the client and server sides, leading to faster and more efficient downloads from the cloud storage service.

Overall, HTTP 2 is designed to be more efficient and faster compared to HTTP 1.1, particularly for modern web applications with many resources to load.