HTTP 1.1 vs HTTP 2

INTRODUCTION

The Hypertext Transfer Protocol, or HTTP, is an application protocol that has been the de facto standard for communication on the World Wide Web since its invention in 1989. From the release of HTTP/1.1 in 1997 until recently, there have been few revisions to the protocol. But in 2015, a reimagined version called HTTP/2 came into use, which offered several methods to decrease latency, especially when dealing with mobile platforms and server-intensive graphics and videos. HTTP/2 has since become increasingly popular, with some estimates suggesting that around a third of all websites in the world support it. In this changing landscape, web developers can benefit from understanding the technical differences between HTTP/1.1 and HTTP/2, allowing them to make informed and efficient decisions about evolving best practices.

HTTP 1.1

HTTP 1.1 is the latest version of Hypertext Transfer Protocol (HTTP), the World Wide Web application protocol that runs on top of the Internet's TCP/IP suite of protocols. HTTP 1.1 provides faster delivery of Web pages than the original HTTP and reduces Web traffic. let's say you are visiting a website at the domain www.example.com. When you navigate to this URL, the web browser on your computer sends an

HTTP request in the form of a text-based message, similar to the one shown here:

GET /index.html HTTP/1.1

Host: www.example.com

This request uses the GET method, which asks for data from the host server listed after Host:. In response to this request, the example.com web server returns an HTML page to the requesting client, in addition to any images, stylesheets, or other resources called for in the HTML. Note that not all of the resources are returned to the client in the first call for data. The requests and responses will go back and forth between the server and client until the web browser has received all the resources necessary to render the contents of the HTML page on your screen.

HTTP/2

Hypertext Transfer Protocol (HTTP) is a set of standards allowing internet users to exchange website information. There have been four HTTP iterations since its introduction in 1991. HTTP/2 was released in 2015 as a major revision to the HTTP/1.1 protocol.

In HTTP/2, the binary framing layer encodes requests/responses and cuts them up into smaller packets of information, greatly increasing the flexibility of data transfer.

Let's take a closer look at how this works. As opposed to HTTP/1.1, which must make use of multiple TCP connections to lessen the effect of HOL blocking, HTTP/2 establishes a single connection object

between the two machines. Within this connection there are multiple streams of data. Each stream consists of multiple messages in the familiar request/response format. Finally, each of these messages split into smaller units called frames

At the most granular level, the communication channel consists of a bunch of binary-encoded frames, each tagged to a particular stream. The identifying tags allow the connection to interleave these frames during transfer and reassemble them at the other end. The interleaved requests and responses can run in parallel without blocking the messages behind them, a process called multiplexing. Multiplexing resolves the head-of-line blocking issue in HTTP/1.1 by ensuring that no message has to wait for another to finish. This also means that servers and clients can send concurrent requests and responses, allowing for greater control and more efficient connection management.

Since multiplexing allows the client to construct multiple streams in parallel, these streams only need to make use of a single TCP connection. Having a single persistent connection per origin improves upon HTTP/1.1 by reducing the memory and processing footprint throughout the network. This results in better network and bandwidth utilization and thus decreases the overall operational cost.

A single TCP connection also improves the performance of the HTTPS protocol, since the client and server can reuse the same secured session for multiple requests/responses. In HTTPS, during the TLS or SSL handshake, both parties agree on the use of a single key throughout the session. If the connection breaks, a new session starts, requiring a newly generated key for further communication. Thus, maintaining a single connection can greatly reduce the resources required for HTTPS

performance. Note that, though HTTP/2 specifications do not make it mandatory to use the TLS layer, many major browsers only support HTTP/2 with HTTPS.

Although the multiplexing inherent in the binary framing layer solves certain issues of HTTP/1.1, multiple streams awaiting the same resource can still cause performance issues.

HTTP 1.1 USES

HTTP 1.0 introduces a number of new features, including the ability to connect to multiple servers at the same time, support for persistent connections, and support for the POST method

HTTP/2 USES

HTTP/2 enables full request and response multiplexing. In practice, this means a connection made to a web server from your browser can be used to send multiple requests and receive multiple responses. This gets rid of a lot of the additional time that it takes to establish a new connection for each request.

Websites using HTTP/2.

Website	Traffic
Geeksforgeeks.org	0.3%
Paypal.com	0.3%
Gitlab.com	0.3%
Wordpress.org	0.3%

HTTP 1.1 VS HTTP 2

HTTP2 is much faster and more reliable than HTTP1. HTTP1 loads a single request for every TCP connection, while HTTP2 avoids network delay by using multiplexing. HTTP is a network delay sensitive protocol in the sense that if there is less network delay, then the page loads faster.

HTTP/2 solves several problems that the creators of HTTP/1.1 did not anticipate. In particular, HTTP/2 is much faster and more efficient than HTTP/1.1. One of the ways in which HTTP/2 is faster is in how it prioritizes content during the loading process.

Key Features Of Http/1.1:

It was no longer required for each connection to be terminated immediately after every request was served with a response; instead, with the keep-alive header, it was possible to have persistent

connections. It allowed multiple requests/responses per TCP connection.

The Upgrade header was used to indicate a preference from the client that made it possible to switch to a more preferred protocol if found appropriate by the server.

HTTP/1.1 provided support for chunk transfers that allowed streaming of content dynamically as chunks and for additional headers to be sent after the message body. This enhancement was particularly useful in cases where values of a field remained unknown until the content had been produced. For example, when the content had to be digitally signed, it was not possible to do so before the entire content gets generated.

Other features that reinforced its stability were introduced such as:

pipelining (the second request is sent before the response to the first is adequately served)

content negotiation (an exchange between client and server to determine the media type, it also provides the provision to serve different versions of a resource at the same URI)

cache control (used to specify caching policies in both requests and responses)

Key Features Of Http/2:

It introduces the concept of a server push where the server anticipates the resources that will be required by the client and pushes them prior to the client making requests. The client retains the authority to deny

the server push; however, in most cases, this feature adds a lot of efficiency to the process.

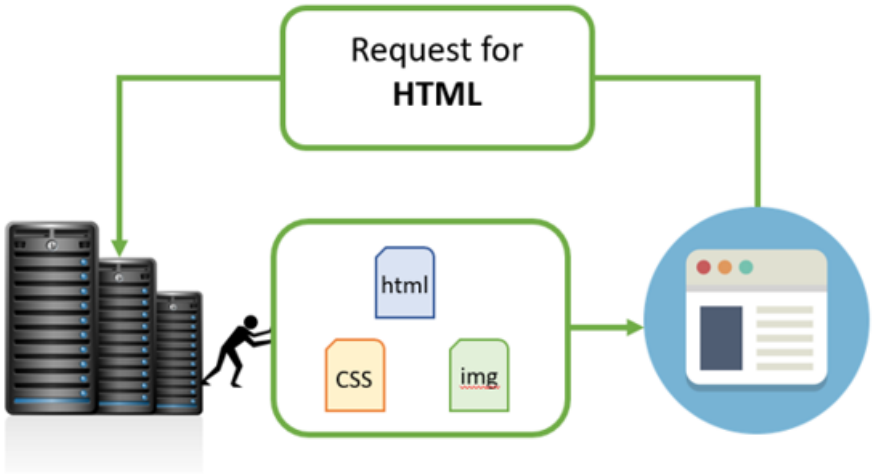


Figure 3: Server Push in HTTP/2

Introduces the concept of multiplexing that interleaves the requests and responses without head-of-line blocking and does so over a single TCP connection.

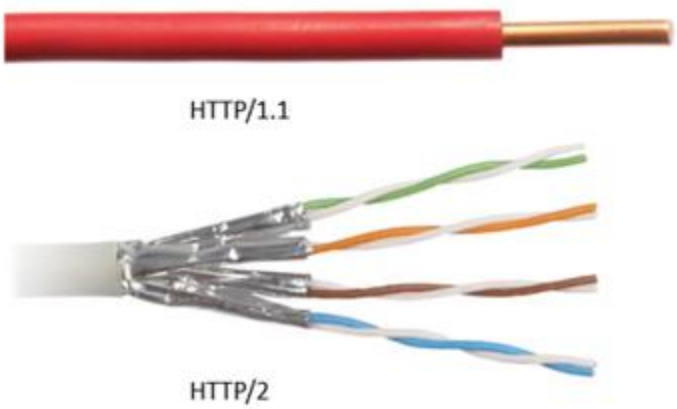
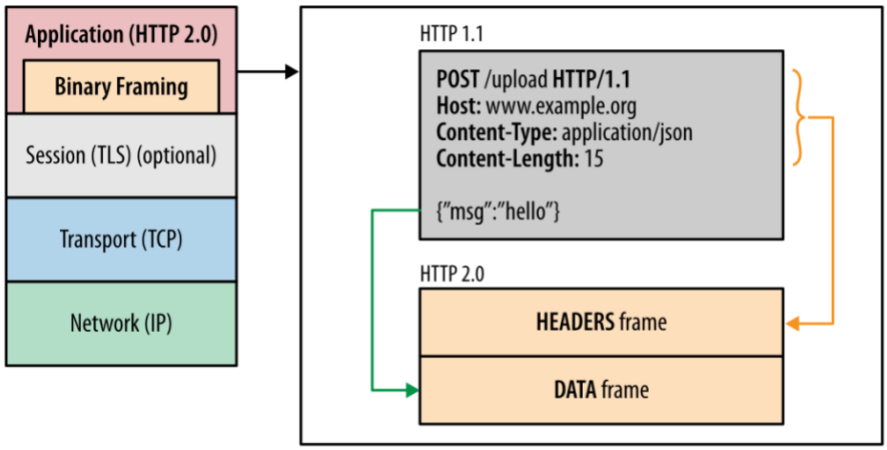


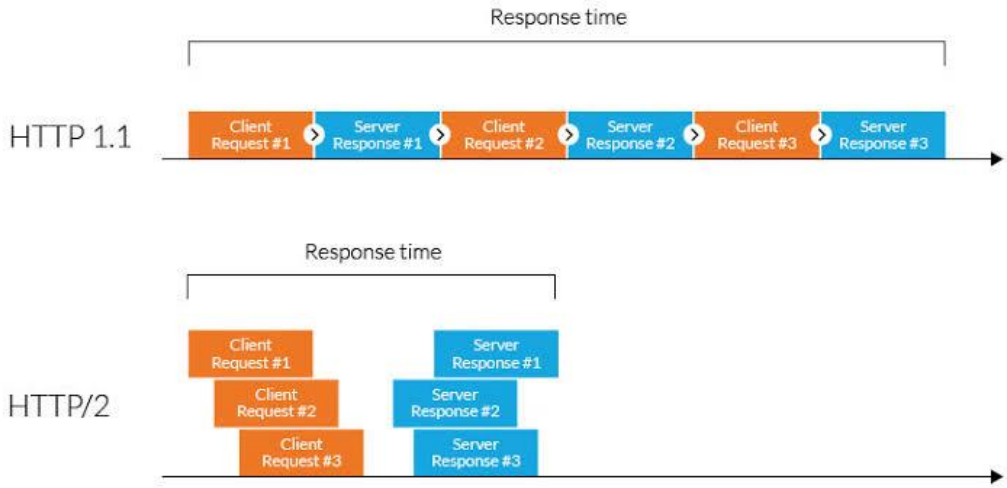
Figure 4: Multiplexing in HTTP/2

It is a binary protocol i.e. only binary commands in the form of 0s and 1s are transmitted over the wire. The binary framing layer divides the message into frames that are segregated based on their type – Data or

Header. This feature greatly increases efficiency in terms of security, compression and multiplexing.



HTTP/2 uses HPACK header compression algorithm that is resilient to attacks like CRIME and utilizes static Huffman encoding.
HTTP/3, the next version in the series, is based on Google's QUIC which, unlike its precursors is a drastic shift to UDP. Given the gradual adoption rate of HTTP/2, HTTP/3 with its security challenges (that comes into play the moment we switch from TCP to UDP) is expected to face some difficulties.



Multiplexing