Difference between HTTP1.1 vs HTTP2

**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.

you will understand the main differences between HTTP/1.1 and HTTP/2, concentrating on the technical changes HTTP/2 has adopted to achieve a more efficient Web protocol.

The Difference

**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.

**Background**

To contextualize the specific changes that HTTP/2 made to HTTP/1.1, let’s first take a high-level look at the historical development and basic workings of each.

**HTTP/1.1**

Developed by Timothy Berners-Lee in 1989 as a communication standard for the World Wide Web, HTTP is a top-level application protocol that exchanges information between a client computer and a local or remote web server. In this process, a client sends a text-based request to a server by calling a *method* like GET or POST. In response, the server sends a resource like an HTML page back to the client.

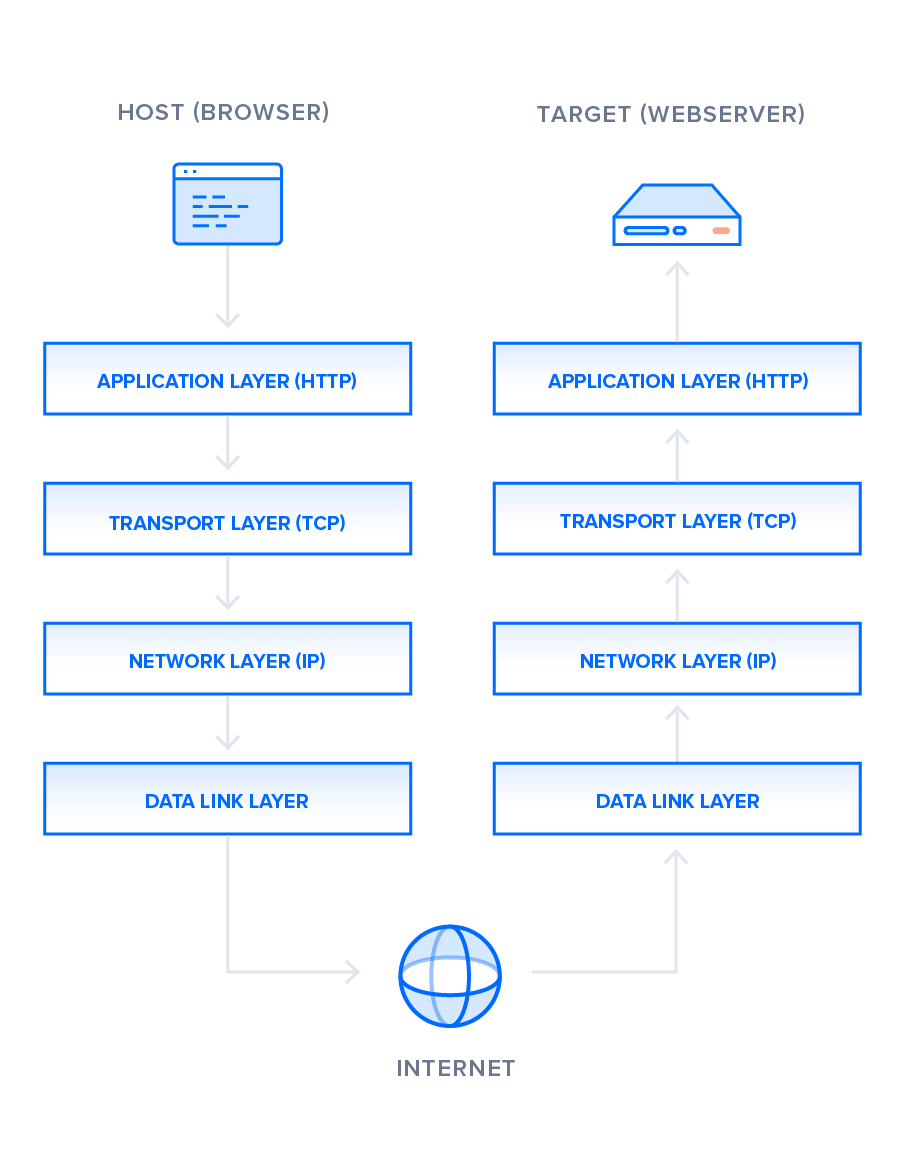
For example, let’s say you are visiting a website at the domain www.example.com. When you navigate to this URL, the web browser on your computer sends an HTTP request in the form of a text-based message, similar to the one shown here:

GET /index.html HTTP/1.1

Host: [www.example.com](http://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.

You can think of this exchange of requests and responses as a single *application layer* of the internet protocol stack, sitting on top of the *transfer layer* (usually using the Transmission Control Protocol, or TCP) and *networking layers* (using the Internet Protocol, or IP):



There is much to discuss about the lower levels of this stack, but in order to gain a high-level understanding of HTTP/2, you only need to know this abstracted layer model and where HTTP figures into it.

With this basic overview of HTTP/1.1 out of the way, we can now move on to recounting the early development of HTTP/2.

**HTTP/2**

HTTP/2 began as the SPDY protocol, developed primarily at Google with the intention of reducing web page load latency by using techniques such as compression, multiplexing, and prioritization. This protocol served as a template for HTTP/2 when the Hypertext Transfer Protocol working group httpbis of the [IETF (Internet Engineering Task Force)](https://www.ietf.org/) put the standard together, culminating in the publication of HTTP/2 in May 2015. From the beginning, many browsers supported this standardization effort, including Chrome, Opera, Internet Explorer, and Safari. Due in part to this browser support, there has been a significant adoption rate of the protocol since 2015, with especially high rates among new sites.

From a technical point of view, one of the most significant features that distinguishes HTTP/1.1 and HTTP/2 is the binary framing layer, which can be thought of as a part of the application layer in the internet protocol stack. As opposed to HTTP/1.1, which keeps all requests and responses in plain text format, HTTP/2 uses the binary framing layer to encapsulate all messages in binary format, while still maintaining HTTP semantics, such as verbs, methods, and headers. An application level API would still create messages in the conventional HTTP formats, but the underlying layer would then convert these messages into binary. This ensures that web applications created before HTTP/2 can continue functioning as normal when interacting with the new protocol.

The conversion of messages into binary allows HTTP/2 to try new approaches to data delivery not available in HTTP/1.1, a contrast that is at the root of the practical differences between the two protocols.

## Conclusion

As you can see from this point-by-point analysis, HTTP/2 differs from HTTP/1.1 in many ways, with some features providing greater levels of control that can be used to better optimize web application performance and other features simply improving upon the previous protocol. Now that you have gained a high-level perspective on the variations between the two protocols, you can consider how such factors as multiplexing HTTP/2 will affect the changing landscape of web development.