What is the HTTP/2 Protocol? Overview and Examples

The HTTP/1.1 protocol has been around for quite some time, but in recent years more web services have switched to the new HTTP/2 protocol. This article will define HTTP and the histories of HTTP/1.1 to HTTP/2, and you can use the links below to discover key differences and advantages of HTTP/2.

Why is HTTP important?

- HTTP (Hypertext Transfer Protocol) is the protocol that powers most of the web today. As you opened this page, your browser made an HTTP request towards a web server that returned the text that you are currently reading.
- HTTP is a client-server protocol. This means requests are sent by one identity, for example your browser, which acts as the client. The request is transferred to the server using the HTTP protocol and tells the server what it wants. To load this page, a request for an HTML page, some images and probably some JavaScript is sent to the server. When it receives the requests it fetches the data and sends it as responses back to your browser which will display them to you.
- HTTP is also used for things that you do not see with the naked eye. When you post something on social media, send a message to one of your friends, or open your favorite app, an HTTP request is most likely made in the background by the application you are using. These requests are made towards web services that might store your social media posts, messages, or app data.
- Another use case for HTTP is streaming data. For smaller pieces of data like HTML pages and JavaScript, the full response can be sent directly, but we need to stream the data to the client for larger files. This is when an HTTP connection is kept open for a longer period and sends data in smaller chunks.
- Without HTTP, the web would simply stop working.

The history of HTTP/1.1 to HTTP/2

In 1989, Sir Timothy John Berners-Lee invented the HTTP protocol on a NeXTcube workstation with a 25 MHz CPU and several MBs of RAM. The protocol worked on networks with port connection speeds of 10 Mbits. Today, however, we have dramatically faster CPUs and thousands of MBs of RAM, but the main WWW protocol is still only HTTP/1.1. It was last updated in 1992, so why are we still using it?

While other protocols have been updated over the years (FTP became SFTP; POP3 evolved to IMAP, and telnet became SSH), HTTP/1.1 has not changed and has developed many issues with speed and security, and user-friendliness as a result.

Google was the first to investigate issues with HTTP/1.1. At the time, they were spending millions of dollars a year to support their data centers, and the HTTP/1.1 protocol simply cost too much in terms of CPU resources and internet connection capacity. They developed SPDY as an experimental alternative to HTTP/1.1—a protocol designed for better security and improved page load times.

The HTTP Working Group of the Internet Engineering Task Force (IETF) investigated Google's SPDY protocol and Microsoft's equivalent when designing the next version of HTTP. Facebook recommended HTTP/2 to be based on SPDY in July 2012.

Based on the research done by Google, Microsoft, and Facebook, the IETF released the HTTP/2 protocol in 2015. This became the second major version of the most useful internet protocol, HTTP.

What is HTTP/2?

HTTP/2 is the second version of the HTTP protocol aiming to make applications faster, simpler, and more robust by improving many of the drawbacks of the first HTTP version.

The primary goals of HTTP/2 are:

- Enable request and response multiplexing
- Header compression
- Compatibility with the methods, status codes, URIs, and header fields defined by the HTTP/1.1 standard
- Optimized prioritization of requests, making sure that loading for optimal user experience is as fast as possible
- Support for server-side push
- Server-side backwards compatibility, making sure servers can still serve clients only supporting HTTP/1.1 without any changes
- Transforming to a binary protocol from the text-based HTTP/1.1

HTTP/2 Features

Some key features of HTTP/2 include:

- Binary: Meaning commands use 1s and 0s and not text
- Multiplex: Permits multiple requests and responses to be sent at the same time
- Compression: Compresses headers that have been requested previously to make things more efficient
- Stream prioritization: This allows for the exchange of successive streams at one time
- Server push: The server can send additional information needed for a request before it is requested
- Increased security: HTTP/2 is supported through encrypted connections

HTTP/1.1 vs. HTTP/2: Main differences

As mentioned, the aim when designing HTTP/2 was to fix many of the drawbacks experienced with HTTP/1.1. In this section, we go through the differences between HTTP/2 and its predecessor.

Compression

HTTP/2 offers built-in compression of the request headers (HPACK). Modern web applications usually accept a range of different headers, such as authorization, caching directives, and client information. While compression of these might not make much of a difference for a single request, there is a lot of data sent over the network to be saved when compressing them in high-traffic applications. HTTP/1.1 does not compress headers by default.

Performance

Many of the new features in HTTP/2 are aimed at improving performance for the end-user. One example of this is how external resources can be preemptively pushed to the client's browser before they are explicitly requested. HTTP/1.1 does not have these advanced features.

Binary protocol

HTTP/2 is binary instead of textual as HTTP/1.1. In practice, this means simplified implementation of commands that previously could be mixed up due to optional whitespace when using the text format. Browsers that support HTTP/2 will convert textual commands into binary before sending them over the network.

Security

Because of the binary format used by HTTP/2, there is no longer a risk with so-called response splitting attacks that are possible with HTTP/1.1. This enables an attacker to manipulate the response headers by injecting whitespace into a textual response. This is no longer possible with the binary format of HTTP/2.

Delivery models

While the HTTP/1.1 protocol delivers responses based on a single request, HTTP/2 uses multiplexing and server push features to increase the delivery performance.

Buffer overflow

The buffer is the space used by the client and server that holds the requests that have not yet been processed. In HTTP/1.1, the flow control used to manage the available buffer space is implemented at the transport layer. In HTTP/2, the client and server can implement their own flow controls to communicate the available buffer space.

Multiplexing

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. HTTP/1.1 does not support multiplexing.

Faster encrypted connections

HTTP/2 uses the new ALPN extension, which allows for faster-encrypted connections since the application protocol is determined during the initial connection. Using HTTP/1.1 without ALPN needs additional round trips for the encryption handshake.

No need for HTTP/1.1 workarounds

In order to bypass some of the drawbacks with HTTP/1.1, multiple workarounds have been invented. Two examples of these are:

Domain sharding is a common performance workaround used with HTTP/1.1 to trick browsers into opening more simultaneous connections than would normally be allowed.

Another common workaround for HTTP/1.1 is content concatenation used to reduce the number of requests for different resources. To achieve this, web developers often combine all the CSS and JavaScript into single files.

These are no longer needed with the built-in multiplexing in HTTP/2.

Benefits of HTTP/2

What are some pros to using HTTP/2?

- HTTP/2 is binary instead of textual.
- HTTP/2 is fully multiplexed. This means that HTTP/2 can send multiple requests for data in parallel over a single TCP connection. This is the most advanced feature of the HTTP/2 protocol because it allows you to download web files via ASync mode from one server. Most modern browsers limit TCP connections to one server.
- It uses header compression HPACK to reduce overhead.
- It allows servers to “push” responses proactively into client caches instead of waiting for a new request for each resource.
- It uses the new ALPN extension, which allows for faster-encrypted connections since the application protocol is determined during the initial connection.
- It reduces additional round trip times (RTT), making your website load faster without any optimization.

Domain sharding and asset concatenation are no longer needed with HTTP/2.

It’s widely supported by browsers. As a basic internet technology, protocol HTTP/2 must be supported by the current version of your browser to work well.

Disadvantages of HTTP/2

First of all, there really isn’t an available alternative today that’s superior to HTTP/2. Browsers refused to support SPDY in favor of HTTP/2, with only Mozilla Firefox still supporting SPDY. But, as an IT professional, you should still know the protocol’s weak points. Some experts believe that these issues may be fixed in the future with the release of the HTTP/3 protocol, but for now, these are a few of the cons.

It’s not very fast or modern.

Most experts around the world expected a lot of new features in the HTTP/2 protocol, but these were not included in the final version. The main reason is simple: HTTP/2 must maintain backwards compatibility with the old HTTP/1.1 by using the same POST and GET requests, codes of status (200, 301, 404, and 500), etc. Also, several new features like compression page headers are vulnerable to BREACH and CRIME attacks.

Encryption is not required.

Years ago, data encryption was mostly limited to financial transactions. But hacks and even government surveillance are more viable threats, with implications on your private life and business.

Encryption may guard you against those types of threats, but disputes inside the team of HTTP/2 developers led to a decision to “leave encryption as is in HTTP/1.1.” This means that owners of a website may still choose a lower level of security, potentially putting users at risk. Developers of browsers can “fix” this issue in a way—currently, they turn on HTTP/2 only if SSL/TLS protocols are available.

Cookie security is still an issue.

One of the biggest security failures of HTTP/1.1 is cookies—.txt files containing client data that may be included in an HTTP request to a server (website). By using these files, the server can identify the client and, for example, keep the client session open, so it doesn’t have to ask for a login and password. Some ad providers like Google can store information about the user in cookies (e.g., gender, age, interests, etc.). That information is then used to target personalized ads to that user.

This may not sound too bad, but the reality is that cookies may be stolen or tampered with by hackers. This means that hackers can access your email, social networks, and other websites that contain your personal data, even without passwords. This is called a cross-site scripting (XSS) attack.

Experts hoped that cookies would be replaced with new technologies, but so far, they have not.