**Difference between HTTP1.1 vs HTTP2**

**What is HTTP?**

**HTTP is based on** the Client/Server model. Client/Server model can be explained as two computers, Client (receiver of service) and Server (provider of service) that are communicating via requests and responses.

A simple and abstract example would be a **restaurant guest and a waiter**. The guest (**Client**) asks (**sends** **request**) waiter (**Server**) for a meal, then the waiter gets the meal from the restaurant chef (**your application logic**) and brings the meal to the guest.

This is a very simplistic example, but it is also the one that will help you understand the concept.

# What is HTTP/2?

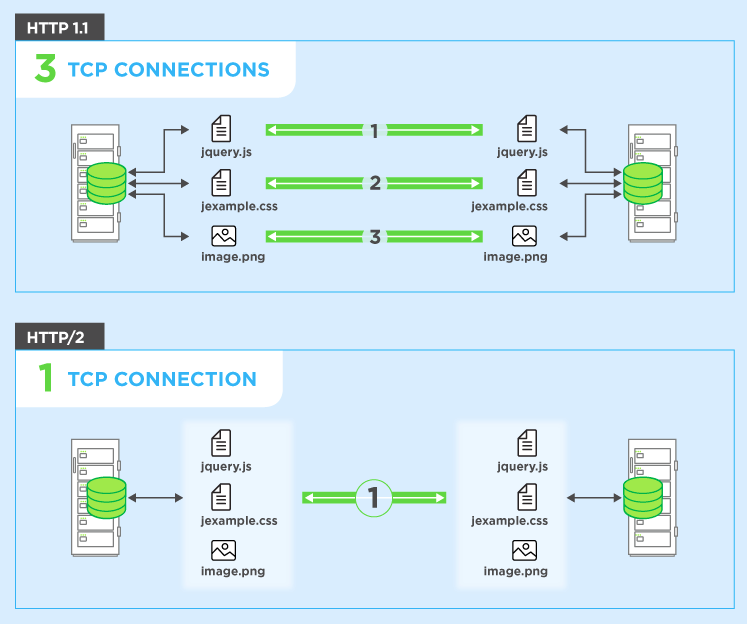
In 2015, Internet Engineering Task Force (IETF) release HTTP/2, the second major version of the most useful internet protocol, HTTP. It was derived from the earlier experimental SPDY protocol.

**Main goals of developing HTTP/2 was:**

* Protocol negotiation mechanism — protocol electing, eg. HTTP/1.1, HTTP/2 or other.
* High-level compatibility with HTTP/1.1 — methods, status codes, URIs and header fields.
* Page load speed improvements trough:
* Compression of request headers
* Binary protocol
* HTTP/2 Server Push
* Request multiplexing over a single TCP connection
* Request pipelining
* HOL blocking (Head-of-line) — Package blocking

# Request multiplexing:

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 asynchronously from one server**. Most modern browsers limit TCP connections to one server.

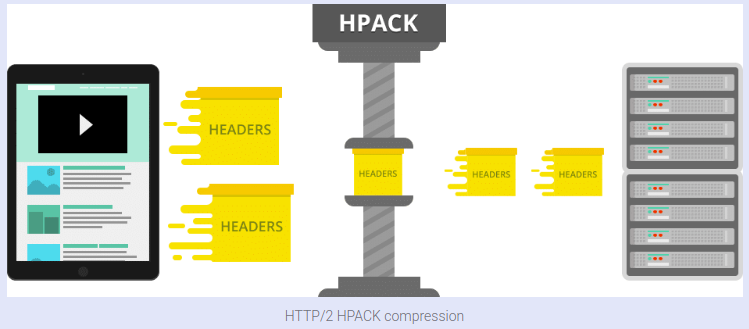


This reduces additional round trip time (RTT), **making your website load faster** without any optimization, and makes domain sharding unnecessary.

# Header compression

HTTP/2 compress a large number of redundant header frames. It uses the HPACK specification as a simple and secure approach to header compression. Both client and server maintain a list of headers used in previous client-server requests.

HPACK compresses the individual value of each header before it is transferred to the server, which then looks up the encoded information in a list of previously transferred header values to reconstruct the full header information.

****

# Binary protocol:

The latest HTTP version has evolved significantly in terms of capabilities and attributes such as transforming from a text protocol to a binary protocol. HTTP1.x used to process text commands to complete request-response cycles. HTTP/2 will use binary commands (in 1s and 0s) to execute the same tasks. This attribute eases complications with framing and simplifies implementation of commands that were confusingly intermixed due to commands containing text and optional spaces.

Browsers using HTTP/2 implementation will convert the same text commands into binary before transmitting it over the network.

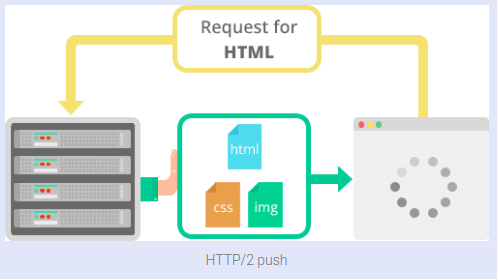
****

# Benefits:

* Low overhead in parsing data — a critical value proposition in HTTP/2 vs HTTP1.
* Less prone to errors.
* Lighter network footprint.
* Effective network resource utilization.
* Eliminating security concerns associated with the textual nature of HTTP1.x such as response splitting attacks.
* Enables other capabilities of the HTTP/2 including compression, multiplexing, prioritization, flow control and effective handling of TLS.
* Compact representation of commands for easier processing and implementation.
* Efficient and robust in terms of processing of data between client and server.
* Reduced network latency and improved throughput.

# HTTP/2 Server Push:

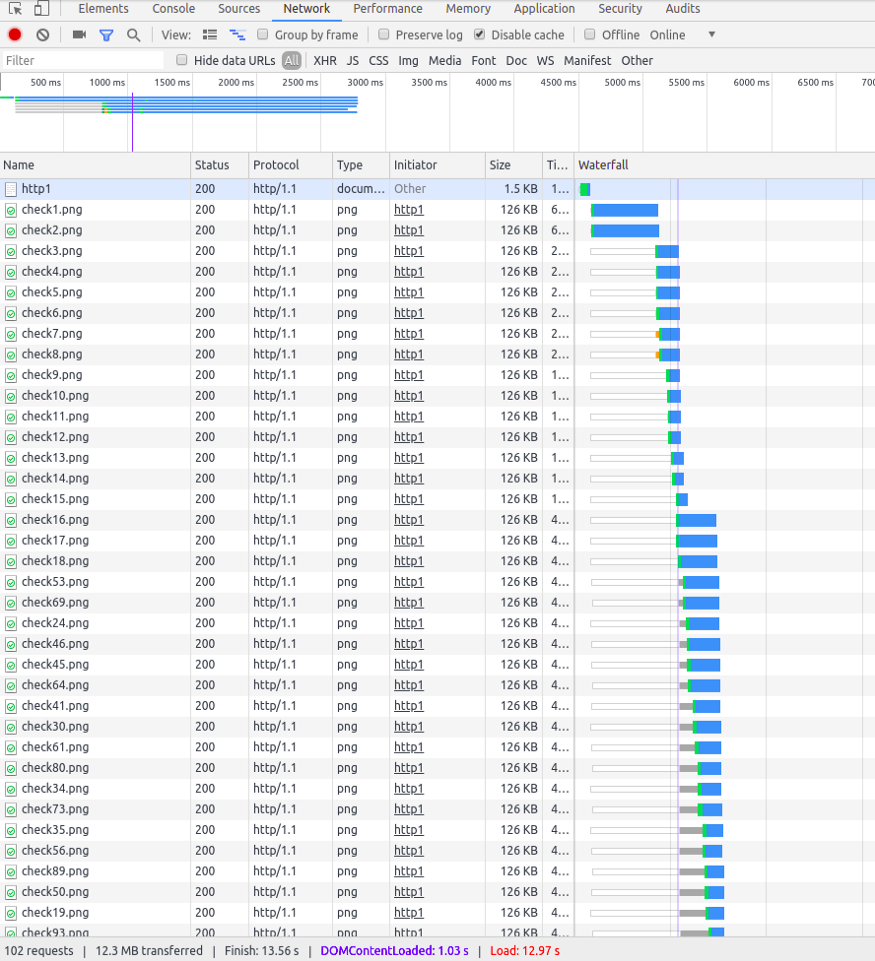
This capability allows the server to send additional cacheable information to the client that isn’t requested but is anticipated in future requests. For example, if the client requests for the resource X and it is understood that the resource Y is referenced with the requested file, the server can choose to push Y along with X instead of waiting for an appropriate client request.

****

# Benefits:

* The client saves pushed resources in the cache.
* The client can reuse these cached resources across different pages.
* The server can multiplex pushed resources along with originally requested information within the same TCP connection.
* The server can prioritize pushed resources — a key performance differentiator in HTTP/2 vs HTTP1.
* The client can decline pushed resources to maintain an effective repository of cached resources or disable Server Push entirely.
* The client can also limit the number of pushed streams multiplexed concurrently.

# For HTTP/1.1:

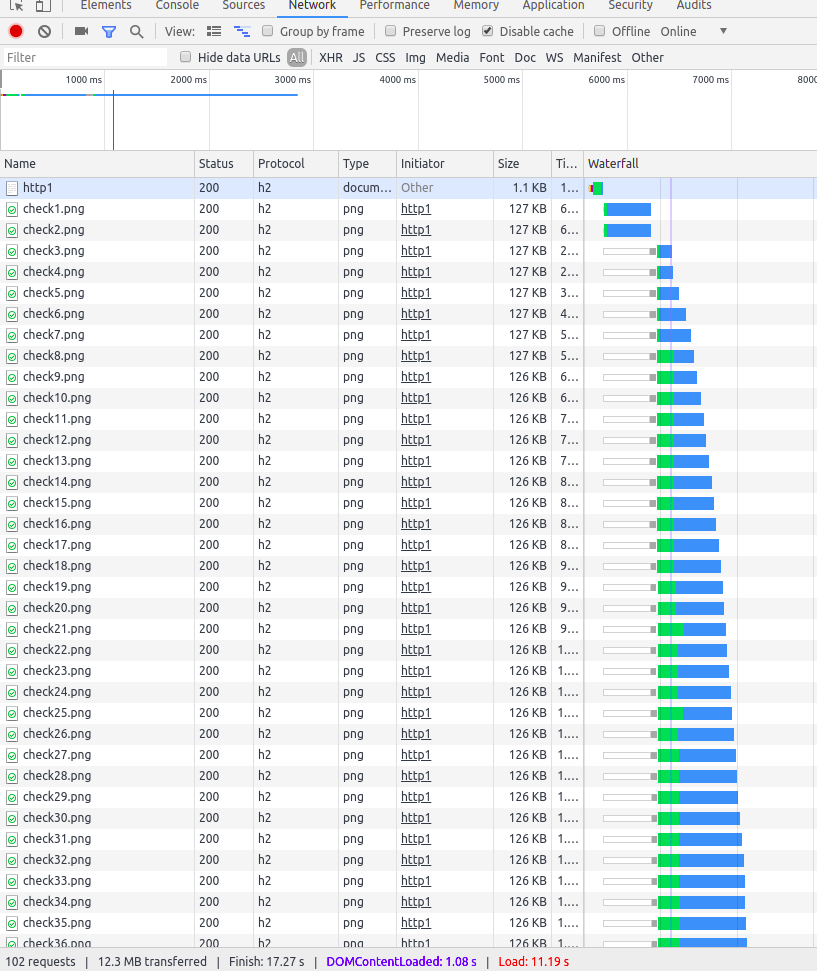
****

Number of requests: 102

Load time: 12.97s

Protocol: “http/1.1

# For HTTP/2:

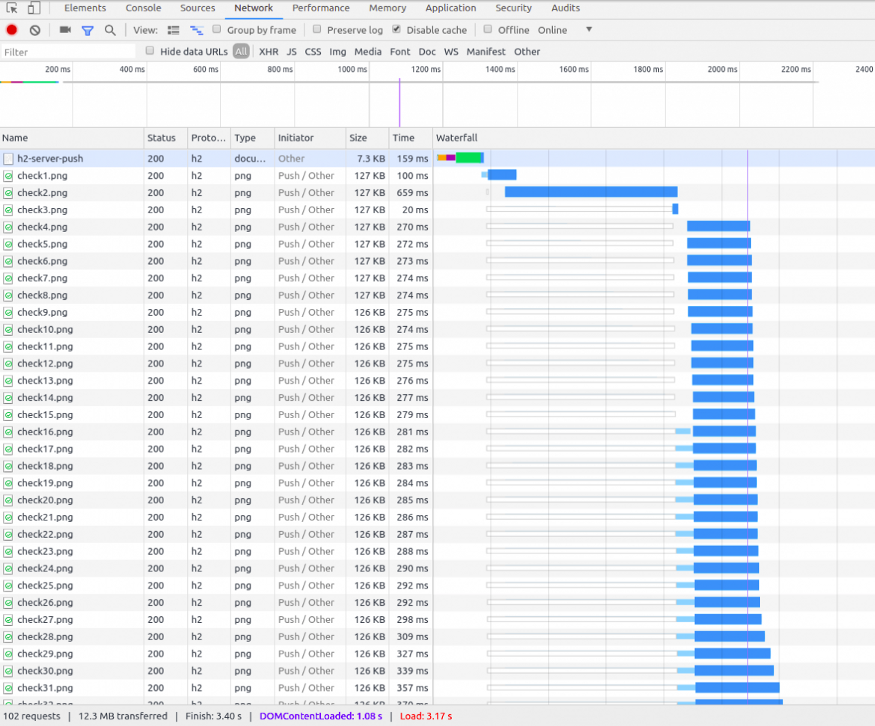
****

Number of requests: 102

Load time: 11.19s

Protocol: “h2” (HTTP/2)

# HTTP/2 server push:



Number of requests: 102

Load time: 3.17 s

Protocol: “h2” (HTTP/2).