

HTPT/2: +Performance

versão 1.1



- [HTPT/2: +Performance](#)
 - [Apresentação Pessoal](#)
 - [Histórico](#)
 - [Onde está no HTTP no modelo OSI?](#)
 - [+Dados sobre o uso do HTTP](#)
 - [Suporte dos Web browsers ao HTTP/2](#)
 - [Anatomia do HTTP/1.1](#)
 - [Forma de transmissão de dados](#)
 - [Frames](#)
 - [Comparações de desempenho](#)
 - [O que o HTTP/2 não resolve](#)
 - [Recursos para Identificação de suporte ao HTTP/2](#)
 - [Implementações](#)
 - [Conformidade](#)

Apresentação Pessoal

Sou o Eduardo Vieira, Analista Programador. Considero-me +1 curioso da Computação desde 2004. Trabalhei no desenvolvimento de vários sistemas desde órgão distrital, federal e para instituição privada com linguagens como Java, Groovy, PHP, Javascript, PL/SQL, Shell Script, AWK *et al*, além de banco de dados como MySQL, Oracle *et al*. Ex-docente

do IFB. Atualmente trabalho na TI do sistema bancário entre os projetos: o PIX. Já precisei escovar muitos bits para resolver problemas de performance ou de baixo nível.

Telegram, Twitter et al: @eduardoenemark.

E-mail: eduardoenemark@gmail.com

Histórico

"HTTP (HyperText Transfer Protocol) is the underlying protocol of the World Wide Web. Developed by Tim Berners-Lee and his team between 1989-1991..."

Developer Mozilla¹

Internet != World Wide Web.

|-> Internet é TCP/IP, enquanto que Web é a sua parte visível, de fácil acesso.²

- **HTTP/0.9 (~Ago/1991): Simples em forma:** sem headers, status ou error codes, URL basicamente referenciado o caminho do documento HTML. Apenas GET:

```
GET /mydoc.html
```

Response:

```
<html>
  A very simple HTML page
</html>
```

- **HTTP/1.0 (RFC 1945: ~Mai/1996)¹:** Redesenhado com methods, headers, status code, version information no line e suporte a outros tipos de arquivos, body com suporte a binary. Praticamente próximo dos dias de hoje:

```
GET /mypage.html HTTP/1.0
User-Agent: NCSA_Mosaic/2.0 (Windows 3.1)
```

```
200 OK
Date: Tue, 15 Nov 1994 08:12:31 GMT
Server: CERN/3.0 libwww/2.17
Content-Type: text/html
<HTML>
A page with an image
  <IMG SRC="/myimage.gif">
</HTML>
```

Response:

```
GET /myimage.gif HTTP/1.0
User-Agent: NCSA_Mosaic/2.0 (Windows 3.1)

200 OK
Date: Tue, 15 Nov 1994 08:12:32 GMT
Server: CERN/3.0 libwww/2.17
Content-Type: text/gif
(image content)
```

- **HTTP/1.1 (RFC 2068: ~Jan/1997):¹** Melhorou o HTTP/1.0 com acréscimo como reutilização de conexão, persistência (keep-alives), pipelining (esquecido), chunked, cache control e novos headers como encoding, content accept, host e securities (CORS: Cross-Origin Resource Sharing e CSP: Content Security Policy).

Temos um HTTP na versão 1.1 muito bem funcional e conhecido na Web, além de padrões de desenvolvimento como SOA (...) e REST (*Representational State Transfer*) construídos sobre ele 😊, porém não tão performático 😞.

- **Tecnologias entre versões do HTTP:** SSL da Netscape em 1994 para um HTTP +seguro que depois virou o TLS, Server-sent events. Ajax, WebSocket *et al*.
- **HTTP/2.0 (RFC 7540: Mai/2015):¹** É antecedido pelo protocolo experimental SPDY (pronuncia "speedy") desenhado pelos engenheiros Mike Belshe e Robert Peon da Google em 2009. Tinha um desempenho 64% melhor no quesito *load time*. Executava sobre o HTTP.² O foco do SPDY era tratar as questões de performance existentes no HTTP/1.1. Surgem conceitos aplicados deste experimento de protocolo:
 - *Multiplexed streams*;
 - *Request prioritization*; e
 - *Header compression*.

No redesenho do HTTP na versão 2 passa a ser um protocolo +binário do que textual, +multiplexado do que síncrono e acréscimos:

- *Flow Control*;
- *Stream prioritization*; e
- *Server push*.

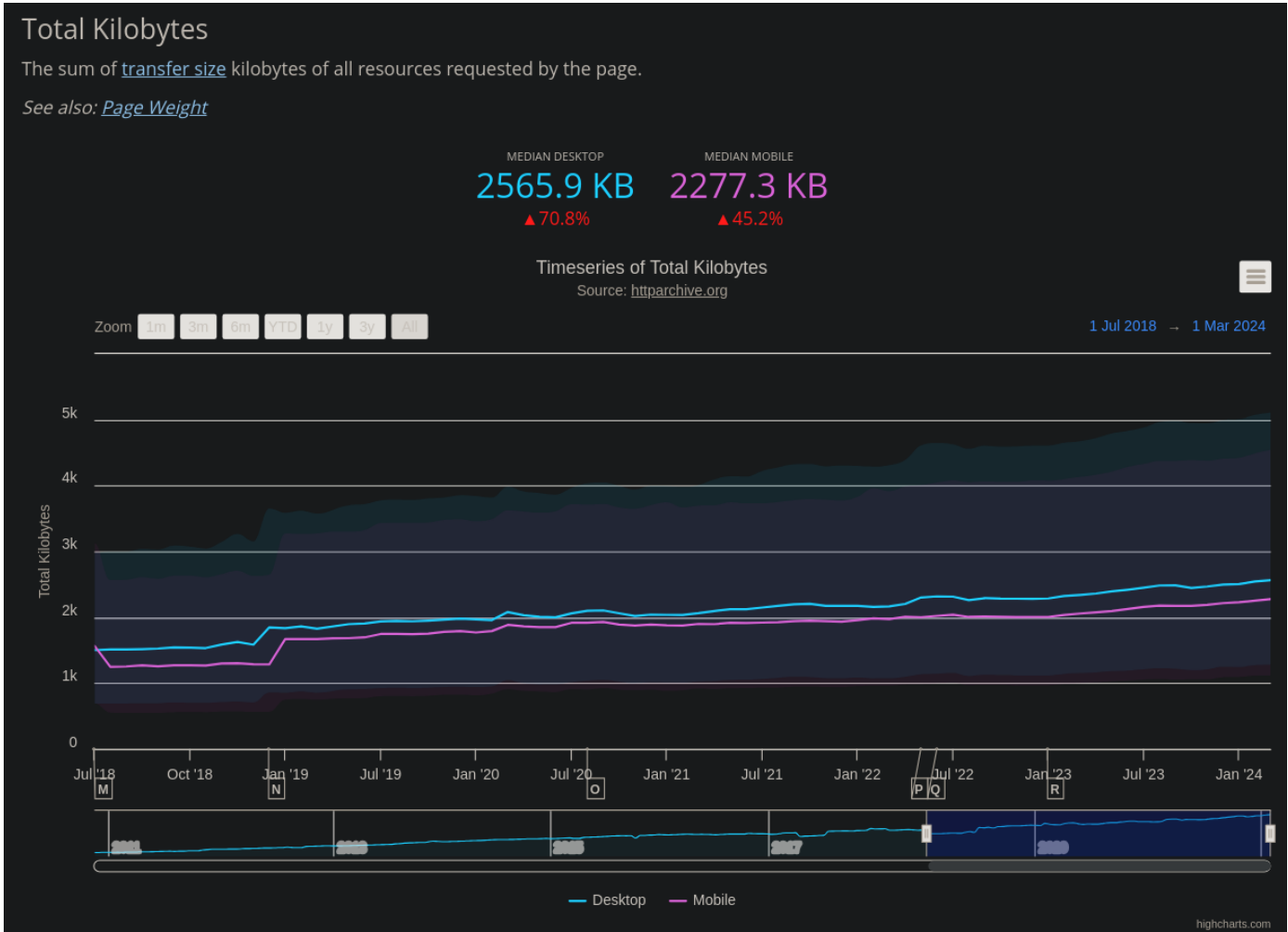
SPDY implementou conceitos do TCP no HTTP utilizando multiplexação de mensagem.

Onde está no HTTP no modelo OSI?



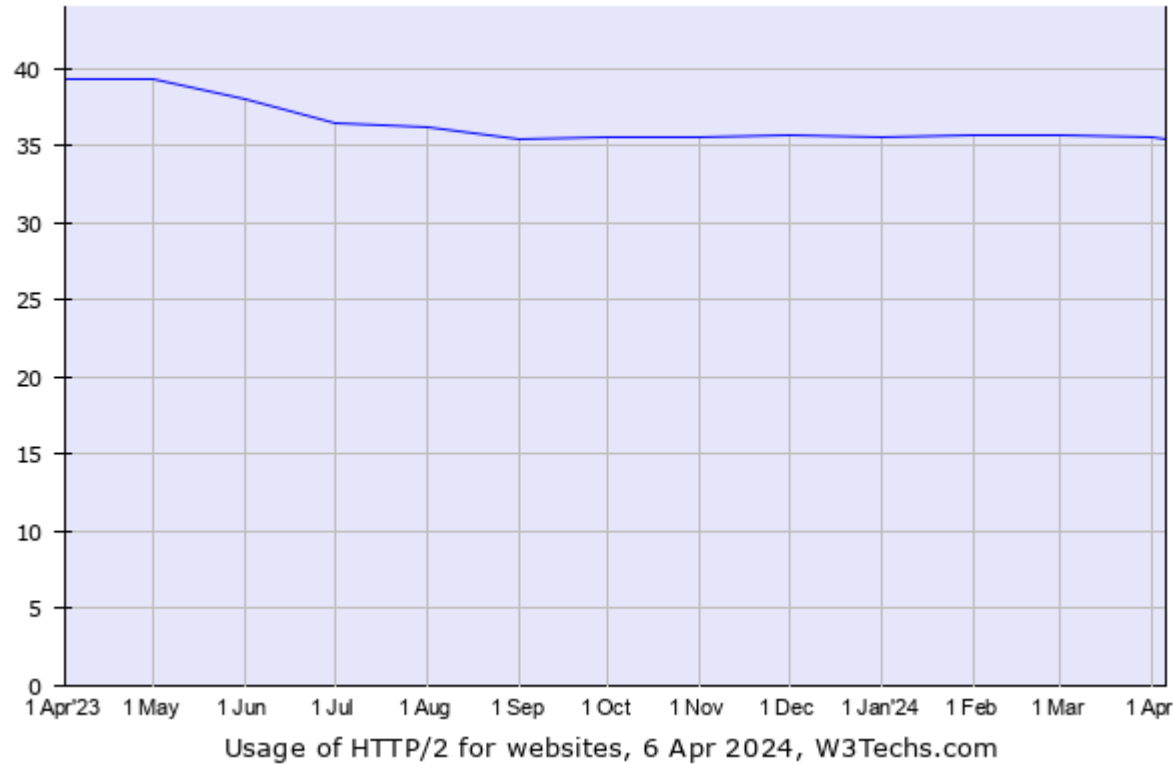
+Dados sobre o uso do HTTP

Tenha em mente que tais estatísticas citadas podem variar em comparação a outras conforme o conjunto de domínios observados ou parte da Web. Logo estas estatísticas e outras apenas dão uma noção imediata seja de uso ou demais métricas vinculadas ao protocolo.



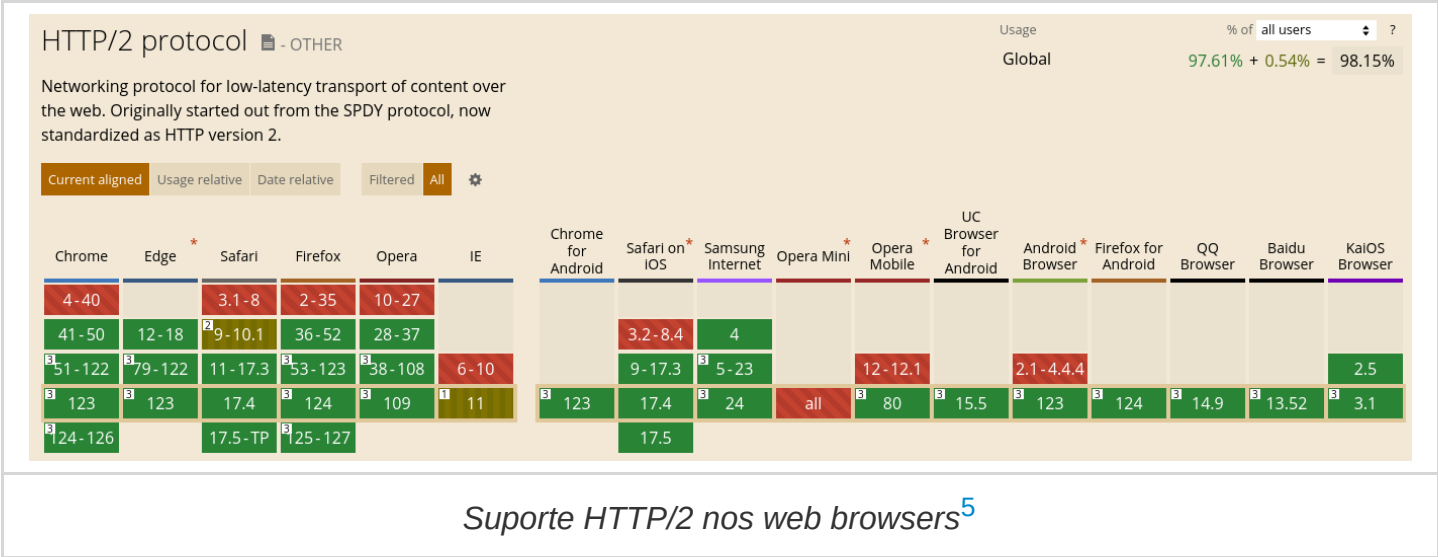


Resources por página³



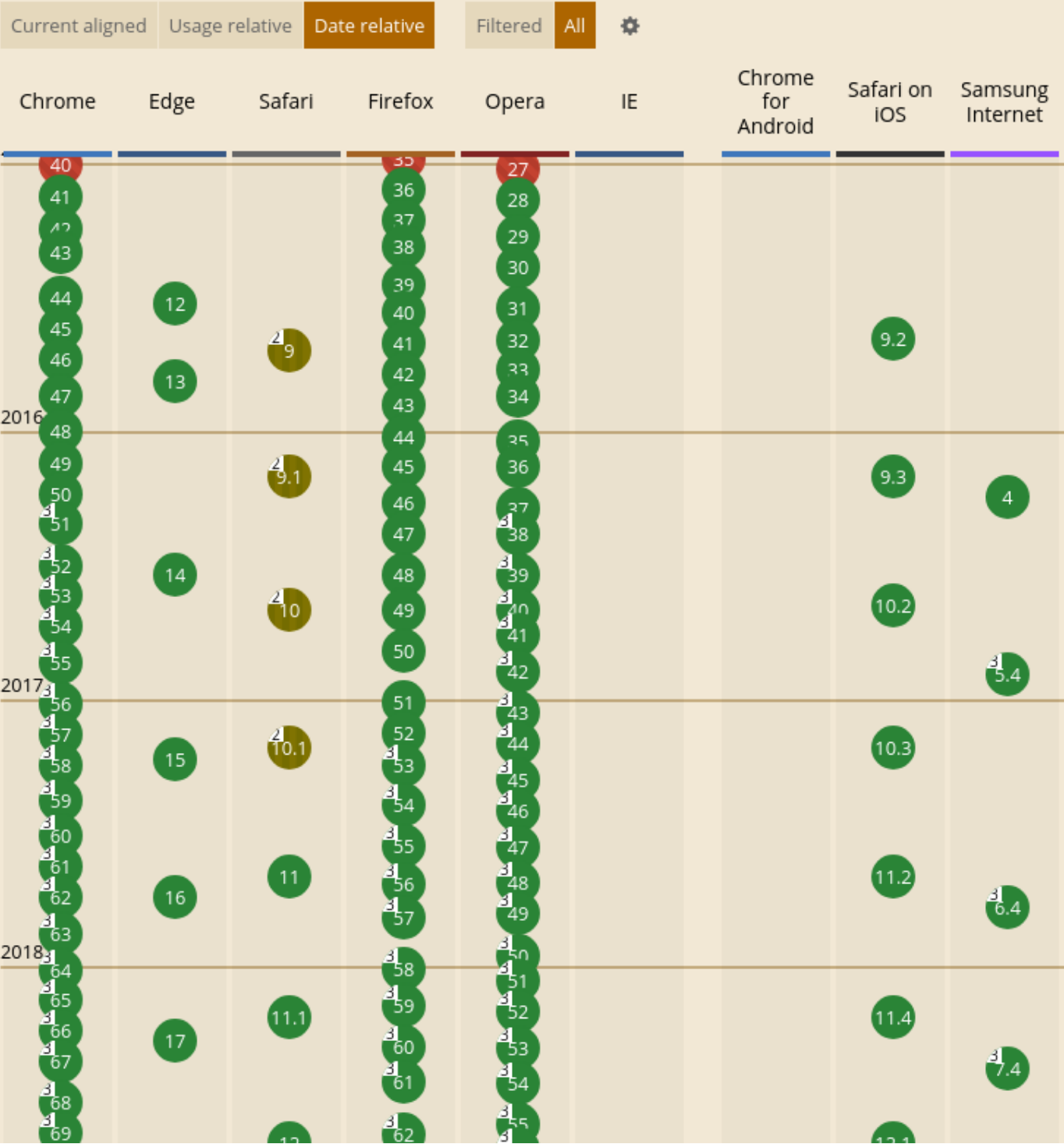
Recorte do Uso do HTTP/2⁴

Suporte dos Web browsers ao HTTP/2



HTTP/2 protocol - OTHER

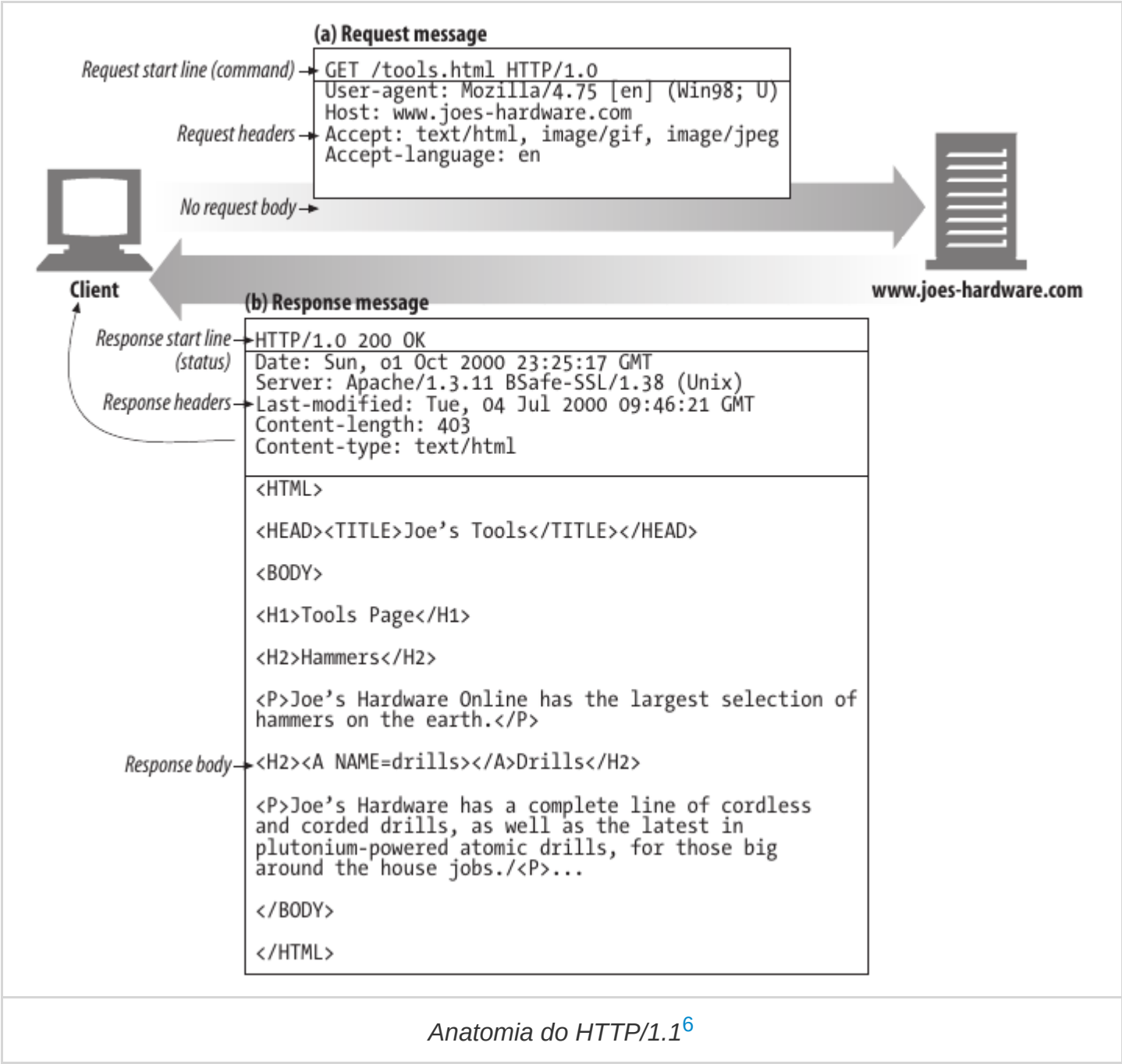
Networking protocol for low-latency transport of content over the web. Originally started out from the SPDY protocol, now standardized as HTTP version 2.

Timeline da versão com suporte ao HTTP/2⁵

Anatomia do HTTP/1.1

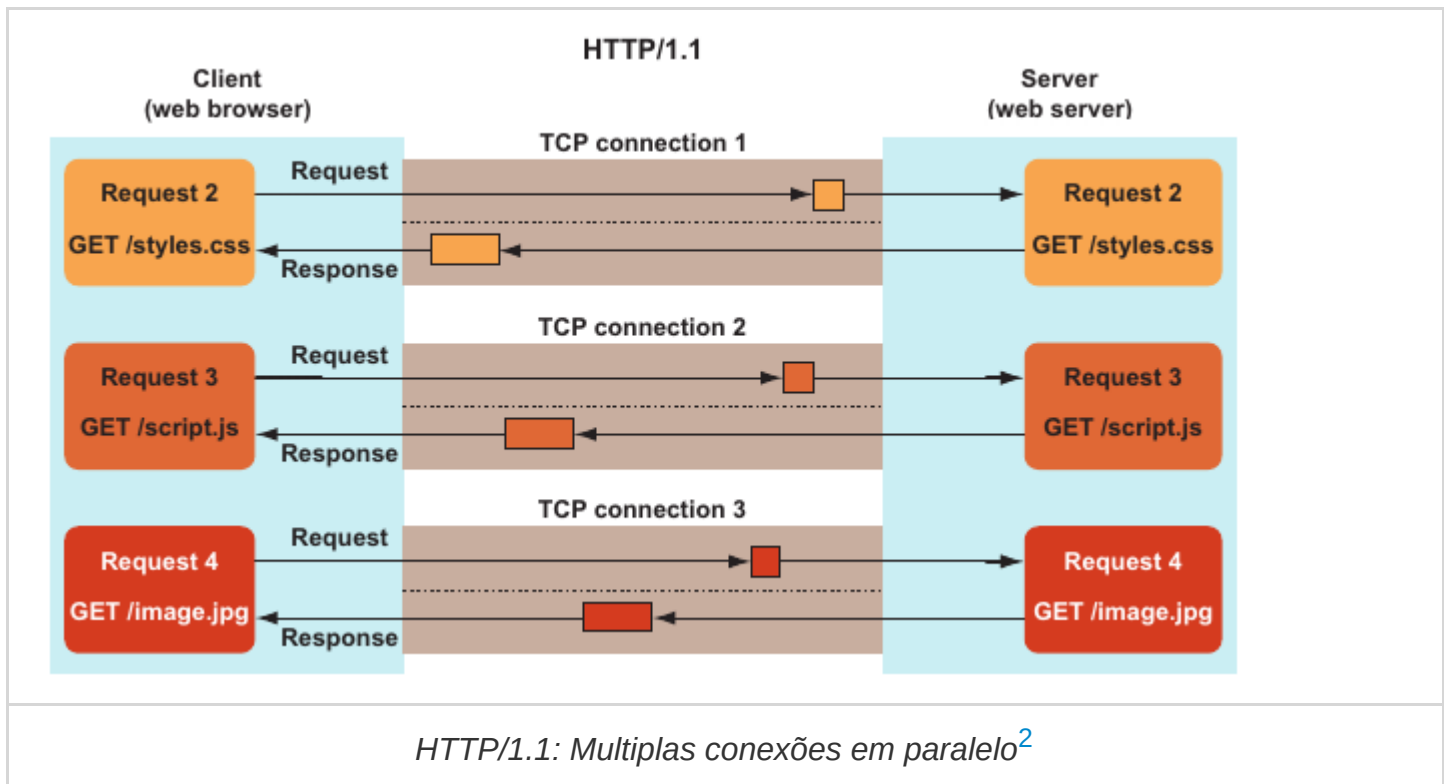
O HTTP/1 é um protocolo textual composto de 3 partes: *Start line*, *headers* e *body* separadas por CRLF (*Carriage Return*, byte 13, + *Line Feed*, byte 10).⁶ Os *headers* e *body* formam um *entity*, entidade. O que o torna um protocolo simples. No HTTP/2 tais partes continuam as mesmas.

- **Start line:** Para o *request* significa "o que fazer", enquanto que para o *response* "o que aconteceu".
- **Headers:** Consiste de campos de nome e valor separado por dois pontos.
- **Body:** É o conteúdo da entidade/mensagem.



Forma de transmissão de dados

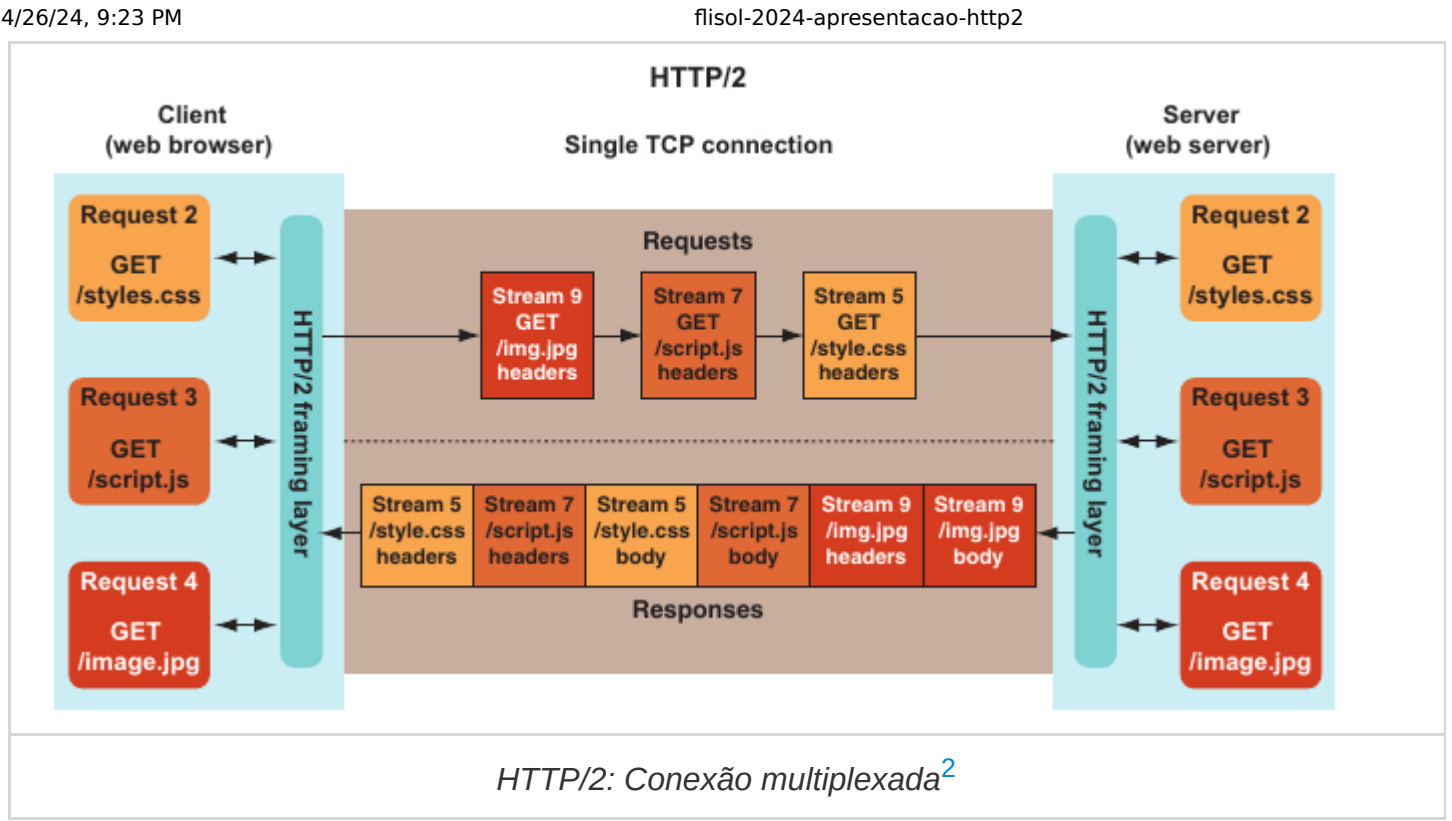
A conexão no HTTP/1.1 é síncrona do tipo *request-and-response (blocking)*, logo o ganho na performance está em múltiplas conexões em paralelo e adoção de caches/CDN (*Content Delivery Network*).



A [RFC 2616](#), Jun/1999, orienta não mais do que 2 conexões para qualquer servidor ou proxy para clientes que persistam a conexão. Esta premissa não se sustentou. Então na [RFC 7230](#), Jun/2014, orienta que o cliente seja ponderante com o uso dos recursos no sentido de evitar congestionamentos, sobrecargas ou causar um *Denial of Service*.

Os atuais web browser tendem a limitar até 6 conexões simultâneas por domínio. Então se uma página da Web fizer referência a recursos que estão em outros domínio, logo serão acrescentadas novas conexões. No Firefox na propriedade *network.http.max-persistent-connections-per-server* possui valor 6 por padrão.

A transmissão de dados no HTTP/2 se dá por um conjunto de *streams* dentro de uma mesma conexão, multiplexação (*non blocking*). Cada *stream* é identificado e vinculado a um determinado recurso, *file*. O *modus operandi* é binário ao contrário da forma textual do HTTP/1.1.



Field	Length	Description
Length	24 bits	Length of the frame, not including all the header fields detailed in this table with a maximum size of $2^{24} - 1$ octets; limited by <code>SETTINGS_MAX_FRAME_SIZE</code> , which defaults to the smaller size of 2^{14} octets
Type	8 bits	Currently, 14 frame types have been defined: ^a <ul style="list-style-type: none">▪ <code>DATA</code> (0x0)▪ <code>HEADERS</code> (0x1)▪ <code>PRIORITY</code> (0x2)▪ <code>RST_STREAM</code> (0x3)▪ <code>SETTINGS</code> (0x4)▪ <code>PUSH_PROMISE</code> (0x5)▪ <code>PING</code> (0x6)▪ <code>GOAWAY</code> (0x7)▪ <code>WINDOW_UPDATE</code> (0x8)▪ <code>CONTINUATION</code> (0x9)▪ <code>ALTSVC</code> (0xa), added through RFC 7838^b▪ (0xb), not used at present but used in the past^c▪ <code>ORIGIN</code> (0xc), added through RFC 8336^d▪ <code>CACHE_DIGEST</code>, proposed^e
Flags	8 bits	Frame-specific flags
Reserved Bit	1 bit	Not currently used and must be set to 0
Stream Identifier	31 bits	An unsigned 31-byte integer identifying the frame

HTTP/2: Frame Header Format²

A visualização e análises facilitada dos frames de uma conexão HTTP/2 pode ser realizada com ferramentas como o *Wireshark* ou *nghttp*, por exemplo. Vamos utilizar o *nghttp*¹⁷ para a visualiar os frames trocados ao acesso do domínio yandex.com:

```
nghttp -v https://yandex.com | less
```

Temos os prints abaixo onde os frames são melhores observáveis dentro da massa de dados retornada. O *h2* sinaliza uma conexão HTTPS/2, o client na abertura da conexão realiza o envio, *send*, dos frames: *SETTINGS*, *PRIORITY* e *HEADERS*. No servidor por vez retorna os frames de resposta, *recv*, que podem atualizar os valores enviados: *SETTINGS* e *WINDOW_UPDATE*.

```
[ 0.304] Connected  
The negotiated protocol: h2  
[ 0.572] send SETTINGS frame <length=12, flags=0x00, stream_id=0>  
      (niv=2)  
        [SETTINGS_MAX_CONCURRENT_STREAMS(0x03):100]  
        [SETTINGS_INITIAL_WINDOW_SIZE(0x04):65535]  
[ 0.572] send PRIORITY frame <length=5, flags=0x00, stream_id=3>  
      (dep_stream_id=0, weight=201, exclusive=0)  
[ 0.572] send PRIORITY frame <length=5, flags=0x00, stream_id=5>  
      (dep_stream_id=0, weight=101, exclusive=0)  
[ 0.572] send PRIORITY frame <length=5, flags=0x00, stream_id=7>  
      (dep_stream_id=0, weight=1, exclusive=0)  
[ 0.572] send PRIORITY frame <length=5, flags=0x00, stream_id=9>  
      (dep_stream_id=7, weight=1, exclusive=0)  
[ 0.572] send PRIORITY frame <length=5, flags=0x00, stream_id=11>  
      (dep_stream_id=3, weight=1, exclusive=0)  
[ 0.572] send HEADERS frame <length=36, flags=0x25, stream_id=13>  
      ; END_STREAM | END_HEADERS | PRIORITY  
      (padlen=0, dep_stream_id=11, weight=16, exclusive=0)  
      ; Open new stream  
      :method: GET  
      :path: /  
      :scheme: https  
      :authority: yandex.com  
      accept: */*  
      accept-encoding: gzip, deflate  
      user-agent: nghttp2/1.43.0  
[ 0.834] recv SETTINGS frame <length=30, flags=0x00, stream_id=0>  
      (niv=5)  
        [SETTINGS_HEADER_TABLE_SIZE(0x01):16384]  
        [SETTINGS_MAX_CONCURRENT_STREAMS(0x03):128]  
        [SETTINGS_INITIAL_WINDOW_SIZE(0x04):262144]  
        [SETTINGS_MAX_FRAME_SIZE(0x05):16384]  
        [SETTINGS_MAX_HEADER_LIST_SIZE(0x06):524288]  
[ 0.834] recv WINDOW_UPDATE frame <length=4, flags=0x00, stream_id=0>  
      (window_size_increment=196609)  
[ 0.834] recv SETTINGS frame <length=0, flags=0x01, stream_id=0>  
      ; ACK  
      (niv=0)  
[ 0.834] send SETTINGS frame <length=0, flags=0x01, stream_id=0>  
      ; ACK  
      (niv=0)  
[ 0.896] recv (stream_id=13) :status: 200  
[ 0.896] recv (stream_id=13) content-security-policy: style-src 'unsafe-inline' https://yastat.sp.yandex.net/csp?project=morda&from=morda.big.com&showid=1714170251454690-111146304531456104-sas-130-BAL&h=stable-portal-mordago-298.sas.jp-c.yandex.net&yandexuid=4456104521714170251&&\nmedia-src https://yastatic.net;connect-src https://*.strm.yandex.net https://mc.yandex.com http\nitic.net https://yastat.net https://mc.yandex.ru https://*.mc.yandex.ru https://adstat.yandex.\ng.org https://t.verify.yandex.ru https://t.va.ru https://t.yandex.ru https://ua.ru https://us
```

Print do início da conexão HTTP/2 (h2) ao domínio yandex.com

```

Error("Failed to load chunk "+r)}))}else!function(e){return e.name.endsWith(".js")}(e)?n(n
y type")):(function(e,o){s(e),d[e].onload&&a({level:"warn",message:"onload override for modu
nload=o}(e.name,(()=>{delete h[e.name],o(p[e.name])})),function(e,o){const n=document.create
n.async=!0,n.src=e,document.head.appendChild(n)}(f+e.hash,(()=>{n(new Error("Failed to load
.home=window.home||{}},home.esBridge={get(e){const o=g(e);return p[o]},getDynamic(o){const n=g
resolve(p[n]);if(n in h)return h[n];const r=m[n];if(!r)return Promise.resolve();const[t,a]=e
(w)).then(()=>(a.forEach(w),h[n]))},set(e,o){const n=g(e);p[n]=o,define:c},c("esBridge.sta
.esBridge.set("esBridge.standalone.js",{l:a,p:e}))}());</script><script crossorigin="anonymous
3/home-static/_/nova/5f3a80a817b30c35adbe5e8ff9ad4172a768b20a288c5a6224f61aaaf82f0734.js"></s
onymous" src="https://yastatic.net/s3/home-static/_/nova/79d6ff4817a6b7d9a6f71c19e550e4c9a34
js"></script><script>home.esBridge.define("src-components-NoCrossoriginLogger--Inner-index-sp
Bridge.standalone.js","init-suggest.standalone.js"),(function(){const{w:e}=home.esBridge.get
ge.get("esBridge.standalone.js"),n=window.React.useEffect;e("42d0bc2f24cf28713712478606adef39
meout(()=>{var e;null==(e=performance.getEntriesByType("resource"))||e.filter((e=>"initiator
atorType&&"responseStatus"in e&&0===e.responseStatus)).forEach((e=>s({message:"Loading script
i-no-crossorigin-logger",level:"info",meta:{url:e.name}}))),1e4);return()=>{clearTimeout(e
/body></html>[ 1.912] rcv DATA frame <length=5003, flags=0x00, stream_id=13>
[ 1.912] rcv DATA frame <length=10, flags=0x01, stream_id=13>
; END_STREAM
[ 1.912] send GOAWAY frame <length=8, flags=0x00, stream_id=0>
(last_stream_id=0, error_code=NO_ERROR(0x00), opaque_data(0)=[])
( END )

```

Print do fim da conexão HTTP/2 (*h2*) ao domínio yandex.com

Após o recebimento dos frames do tipo *DATA* temos o encerramento dos *streams* pelo cliente ao envio do frame *GOAWAY*.

Comparações de desempenho

Para fins didáticos de comparação, inicialmente vamos utilizar o site [Tune The Web](https://tunetheweb.com)⁷ para verificar resultados entre o HTTP/1.1 e HTTP/2. O teste aplicado⁸ é apenas solicitar a mesma imagem 360 vezes utilizando um contador presente na *query* da URL. Iniciamos pelo HTTP/1.1 de tempo total de 61.321s:

Tune The Web⁷: HTTP Performance Test⁷

The screenshot displays the Chrome DevTools Network tab. The top panel shows a list of requests, all of which are 200 OK status codes. The bottom panel shows the details of a selected request, including the Request URL, Request Method, Status Code, Remote Address, and Referrer Policy. The 'Response Headers' section is expanded, showing various headers such as 'Accept-Ranges', 'Cache-Control', 'Connection', 'Content-Length', 'Content-Security-Policy', 'Content-Type', 'Date', 'Expires', 'Keep-Alive', 'Last-Modified', 'Referrer-Policy', 'Server', 'X-Content-Type-Options', and 'X-Frame-Options'. The 'Request Headers' section is also expanded, showing headers like 'Accept', 'Accept-Encoding', 'Accept-Language', 'Connection', 'Cookie', 'Host', 'Referer', 'Sec-Opcode', and 'User-Agent'.

Inspeção pelo DevTools do HTTP Performance Test⁷

15/22

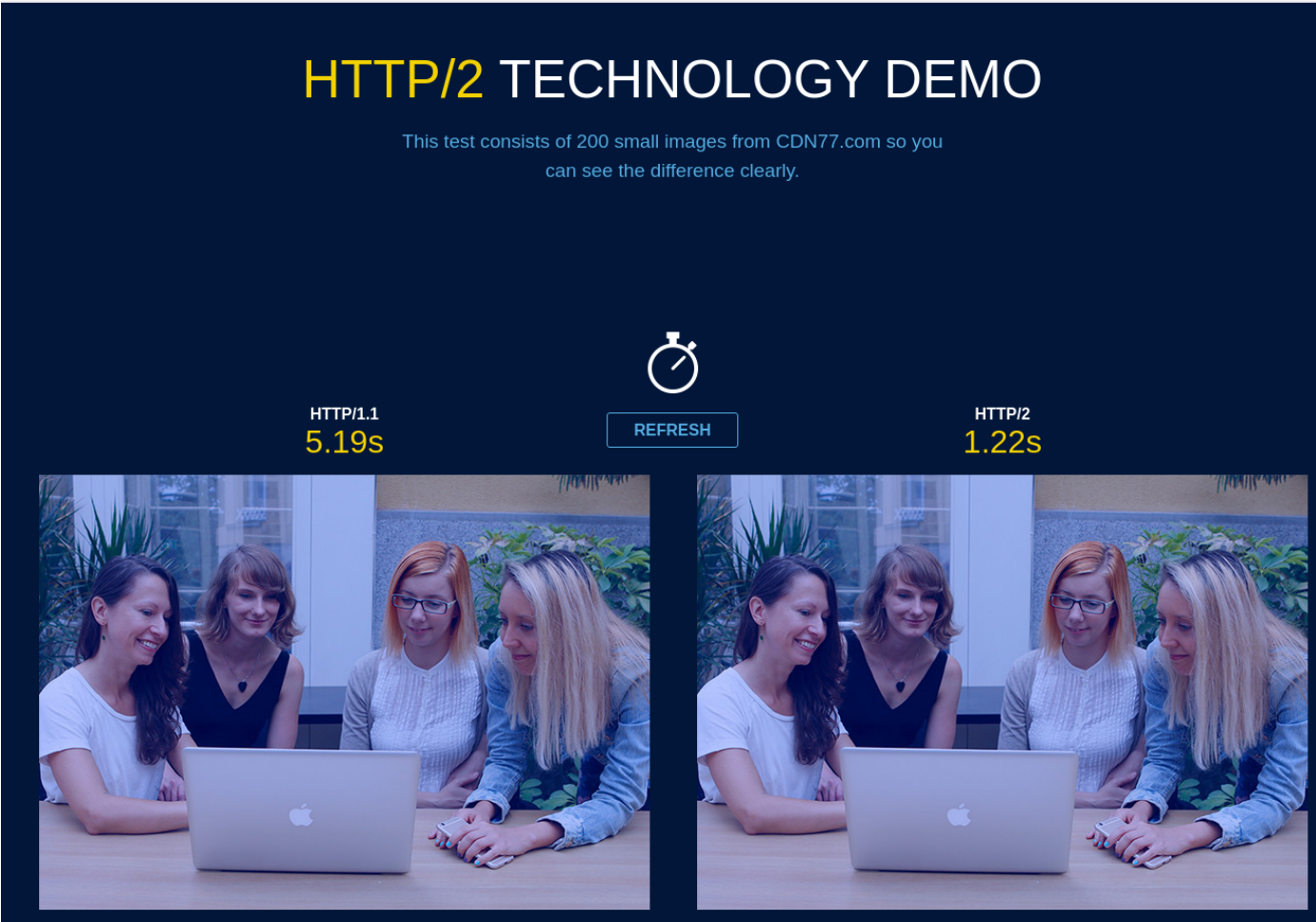
Tune The Web⁷: HTTP/2 Performance Test⁷

The screenshot shows the Chrome DevTools Network tab. The top panel displays a list of requests, with the first 30 requests being 'wrench-icon-24-202352.png?count=262' through 'wrench-icon-24-202352.png?count=301'. The second panel shows the details of the selected request, 'wrench-icon-24-202352.png?count=271'. The 'General' tab is active, showing the request URL, method (GET), status code (200 OK), and remote address (52.24.10.188:443). The 'Response Headers' tab is also visible, showing headers such as 'Accept-Ranges', 'Cache-Control', 'Content-Length', 'Content-Security-Policy', 'Content-Type', 'Date', 'Expires', 'Last-Modified', 'Referer-Policy', 'Server', 'Strict-Transport-Security', 'X-Content-Type-Options', 'X-Frame-Options', 'Request Headers', 'Authority', 'Method', 'Path', 'Scheme', 'Accept', 'Accept-Encoding', 'Accept-Language', 'Cookie', 'Referer', 'Sec-Ch-Ua', 'Sec-Ch-Ua-Mobile', 'Sec-Ch-Ua-Platform', 'Sec-Fetch-Dst', 'Sec-Fetch-Mode', 'Sec-Fetch-Site', and 'Sec-Gpc'.

Tune The Web⁷: Inspeção pelo DevTools do HTTP Performance Test⁷

No teste acima o HTTP/2 não teve ainda melhor performance devido a camada TLS que é submetida a requisição e mensagem de resposta. Os *web browsers* utilizam preferencialmente comunicações HTTP/2 sobre TLS.

O site HTTP2 Demo⁹, também, faz um teste bem semelhantes ao visto do Tune The Web⁷. Quando a página é carregada ou utilizado o botão *Refresh* na mesma é disparado uma sequência de requisições do tipo HTTP/1.1 para compor a imagem maior de exemplo constituída de outras 170 imagens menores.

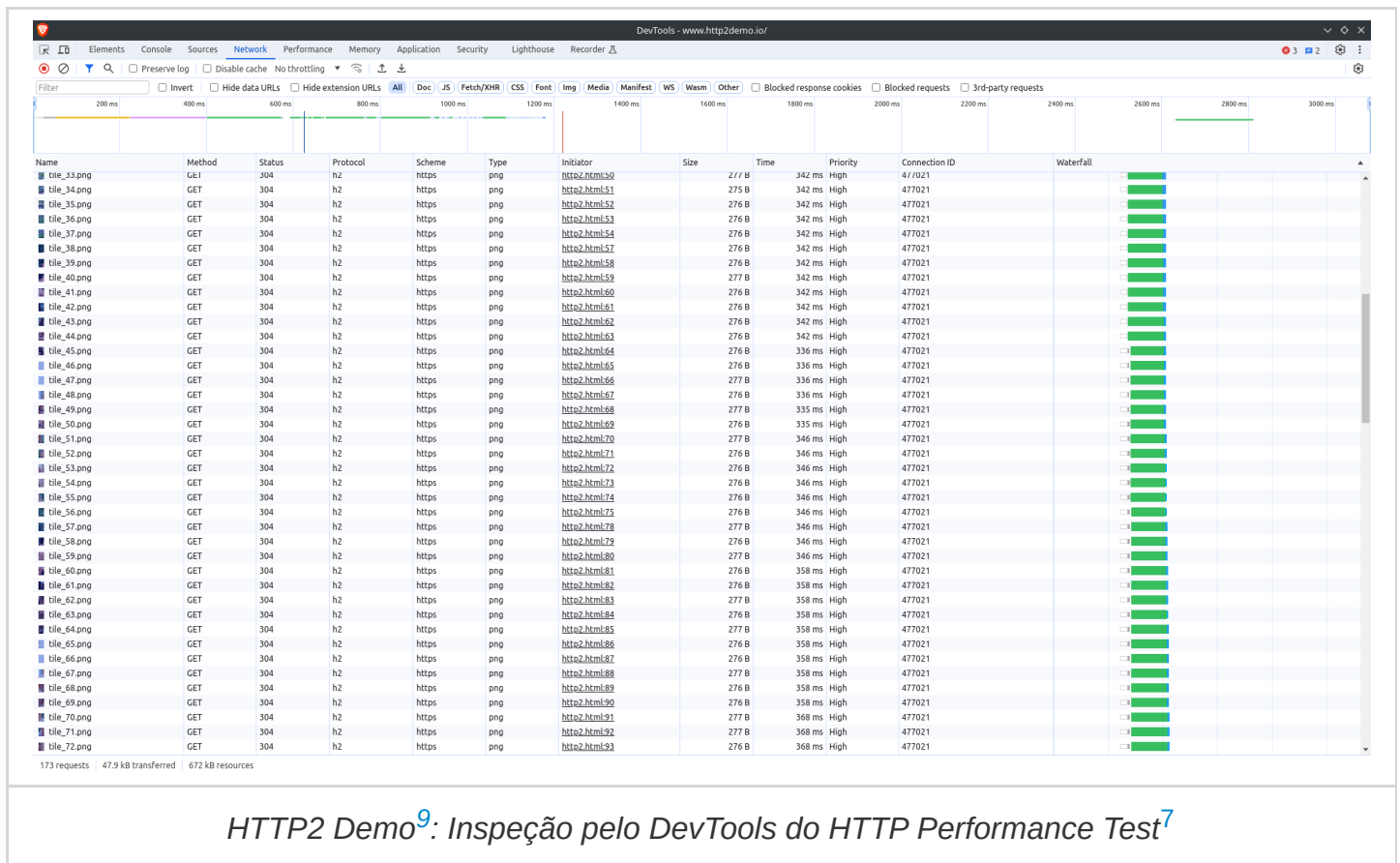


The screenshot shows a web browser window with the address bar displaying "http2demo.io". The page has a dark blue background with the title "HTTP/2 TECHNOLOGY DEMO" in large yellow and white letters. Below the title, it says "This test consists of 200 small images from CDN77.com so you can see the difference clearly." In the center, there is a stopwatch icon and a "REFRESH" button. On either side of the button, the load times are displayed: "HTTP/1.1 5.19s" on the left and "HTTP/2 1.22s" on the right. Below these times are two side-by-side images of four women sitting around a table with a laptop, which are the result of the image requests.

HTTP2 Demo⁹: Teste comparativo de performance entre HTTP/1.1 e 2

Nesta nova inspeção habilitamos para display as colunas: *Method*, *Protocol*, *Time* e *Connection ID*. Existem três pontos a ser notar nesta inspeção do DevTools, imagem abaixo:

- A coluna *Protocol* tem o valor *h2* que sinaliza que a requisição realizada é do tipo HTTP/2;
- A coluna *Connection ID* é o mesmo de número 477021; e
- *Waterfall* possui uma diferença mínima de tempo entre os streams quando visto pelo conjunto de *Queueing* (retângulo branco), *Stalled* (cinza), *Waiting* (verde) e *Content Download* (azul)¹⁰.



O que o HTTP/2 não resolve

- Páginas Web mal estruturadas, por exemplo: com imagens gigantes, muitos arquivos CSS e JS sem uso;
- Serviço de backend de processamento dos dados é lento para responder requisições da Web;
- Falta de estratégias de uso de cache;
- Não uso de compactação para a transmissão de dados quando suportado pelo cliente;
- Quantidade enorme de *headers* sem uso de estratégias de compactação ou desnecessários; e
- Requisição única para um único resource, *file*. A diferença de tempo não é significativa.

Recursos para Identificação de suporte ao HTTP/2

- HTTP Dev: <https://http.dev/2/test>
- KeyCDN: <https://tools.keycdn.com/http2-test>
- HTTP2 Pro: <https://http2.pro>
- HTTP Header Upgrade, exemplo pela CLI (curl¹¹) onde é realizado uma requisição HTTP/1.1, porém o servidor informa no *header response* que suporta o HTTP/2 (*Upgrade: h2,h2c*):

```
curl -kvo /dev/null --http1.1 -L https://debian.org
```

Response:

```
...  
< HTTP/1.1 200 OK  
< Date: Wed, 10 Apr 2024 03:05:06 GMT  
< Server: Apache  
< Content-Location: index.en.html  
< Vary: negotiate, accept-language, Accept-Encoding, cookie  
< TCN: choice  
< X-Content-Type-Options: nosniff  
< X-Frame-Options: sameorigin  
< Referrer-Policy: no-referrer  
< X-Xss-Protection: 1  
< Permissions-Policy: interest-cohort=()  
< Strict-Transport-Security: max-age=15552000  
< Upgrade: h2,h2c  
< Connection: Upgrade  
...
```

Poderíamos usar a opção `--http2` no curl em um outro domínio qualquer e haveria resposta como *HTTP/1.1 200 OK* no *header response* em caso de não suporte ao HTTP/2.

Implementações

No Github do IETF HTTP Working Group¹² temos o repositório *http2-spec*¹³ que trata das implementações do HTTP/2, além de outras *tools*¹⁴ que poderão nos auxiliar em demais testes:

github.com/httpwg/http2-spec/wiki/Implementations																																																																							
IssuesActionsProjects1WikiSecurityInsights																																																																							
<h1>Implementations</h1> <p>sanket0731 edited this page on Jun 17, 2020 · 344 revisions</p> <p>This wiki tracks known implementations of HTTP/2. See also our Tools listing.</p> <p>Please add your implementation below.</p> <table><thead><tr><th>name</th><th>language</th><th>version</th><th>role(s)</th><th>negotiation(s)</th><th>protocol id(s)</th></tr></thead><tbody><tr><td>Ace</td><td>Elixir</td><td></td><td>client, server</td><td>ALPN</td><td>h2</td></tr><tr><td>Aerys</td><td>PHP</td><td></td><td>server</td><td>ALPN, Upgrade, direct</td><td>h2, h2c</td></tr><tr><td>Akamai GHost</td><td>C++</td><td></td><td>intermediary</td><td>ALPN, NPN</td><td>h2, h2-14</td></tr><tr><td>Apache HTTP Server 2.4.17+</td><td>C</td><td></td><td>server</td><td>ALPN, Upgrade, direct</td><td>h2, h2c</td></tr><tr><td>Apache HttpComponents 5.0-beta1</td><td>Java</td><td></td><td>client,server</td><td>ALPN, Upgrade, direct</td><td>h2</td></tr><tr><td>Apache Traffic Server v5.3.0</td><td>C++</td><td></td><td>intermediary</td><td>ALPN, NPN</td><td>h2, h2-14</td></tr><tr><td>Apache Tomcat 8.5+</td><td>Java</td><td></td><td>Server</td><td>ALPN, Upgrade, direct</td><td>h2, h2c</td></tr><tr><td>Armeria</td><td>Java</td><td></td><td>client, server</td><td>ALPN, Upgrade, direct</td><td>h2, h2c</td></tr><tr><td>http4s-blaze</td><td>Scala</td><td></td><td>server</td><td>ALPN</td><td>h2, h2-14</td></tr><tr><td>Brocade Traffic Manager (formerly Riverbed/Zeus)</td><td>C++</td><td></td><td>Server</td><td>ALPN, Upgrade, direct</td><td>h2, h2c</td></tr></tbody></table>						name	language	version	role(s)	negotiation(s)	protocol id(s)	Ace	Elixir		client, server	ALPN	h2	Aerys	PHP		server	ALPN, Upgrade, direct	h2, h2c	Akamai GHost	C++		intermediary	ALPN, NPN	h2, h2-14	Apache HTTP Server 2.4.17+	C		server	ALPN, Upgrade, direct	h2, h2c	Apache HttpComponents 5.0-beta1	Java		client,server	ALPN, Upgrade, direct	h2	Apache Traffic Server v5.3.0	C++		intermediary	ALPN, NPN	h2, h2-14	Apache Tomcat 8.5+	Java		Server	ALPN, Upgrade, direct	h2, h2c	Armeria	Java		client, server	ALPN, Upgrade, direct	h2, h2c	http4s-blaze	Scala		server	ALPN	h2, h2-14	Brocade Traffic Manager (formerly Riverbed/Zeus)	C++		Server	ALPN, Upgrade, direct	h2, h2c
name	language	version	role(s)	negotiation(s)	protocol id(s)																																																																		
Ace	Elixir		client, server	ALPN	h2																																																																		
Aerys	PHP		server	ALPN, Upgrade, direct	h2, h2c																																																																		
Akamai GHost	C++		intermediary	ALPN, NPN	h2, h2-14																																																																		
Apache HTTP Server 2.4.17+	C		server	ALPN, Upgrade, direct	h2, h2c																																																																		
Apache HttpComponents 5.0-beta1	Java		client,server	ALPN, Upgrade, direct	h2																																																																		
Apache Traffic Server v5.3.0	C++		intermediary	ALPN, NPN	h2, h2-14																																																																		
Apache Tomcat 8.5+	Java		Server	ALPN, Upgrade, direct	h2, h2c																																																																		
Armeria	Java		client, server	ALPN, Upgrade, direct	h2, h2c																																																																		
http4s-blaze	Scala		server	ALPN	h2, h2-14																																																																		
Brocade Traffic Manager (formerly Riverbed/Zeus)	C++		Server	ALPN, Upgrade, direct	h2, h2c																																																																		
*HTTP WG Implementations ¹³																																																																							

Conformidade

Independentemente da escolha da implementação escolhida sempre é necessário que possamos realizar verificações se está conforme descrito nas RFCs, além é claro de demais teste de integração e navegação.

E uma ferramenta, em especial, citada no repositório [http2-spec](#)¹³ que possa nos ajudar a realizar estas verificações é a [h2spec](#)¹⁵. Podemos realizar o *build* ou baixar uma versão pronta, *release*.

Na CLI executamos o comando:

```
./h2spec -t -k -S -h apache.org -p 443
```

Como visto no resultado abaixo a implementação HTTP/2 no [httpd](#)¹⁶ no domínio raiz da Apache teve ótimo desempenho, apenas 1 um teste dos 147:

```
8.1.2.3. Request Pseudo-Header Fields
✓ 1: Sends a HEADERS frame with empty ":path" pseudo-header field
✓ 2: Sends a HEADERS frame that omits ":method" pseudo-header field
✓ 3: Sends a HEADERS frame that omits ":scheme" pseudo-header field
✓ 4: Sends a HEADERS frame that omits ":path" pseudo-header field
✓ 5: Sends a HEADERS frame with duplicated ":method" pseudo-header field
✓ 6: Sends a HEADERS frame with duplicated ":scheme" pseudo-header field
✓ 7: Sends a HEADERS frame with duplicated ":path" pseudo-header field

8.1.2.6. Malformed Requests and Responses
✓ 1: Sends a HEADERS frame with the "content-length" header field which does not equal the DATA frame payload length
✓ 2: Sends a HEADERS frame with the "content-length" header field which does not equal the sum of the multiple DATA frames payload length

8.2. Server Push
✓ 1: Sends a PUSH_PROMISE frame

HPACK: Header Compression for HTTP/2
2. Compression Process Overview
2.3. Indexing Tables
2.3.3. Index Address Space
✓ 1: Sends a indexed header field representation with invalid index
✓ 2: Sends a literal header field representation with invalid index

4. Dynamic Table Management
4.2. Maximum Table Size
✓ 1: Sends a dynamic table size update at the end of header block

5. Primitive Type Representations
5.2. String Literal Representation
✓ 1: Sends a Huffman-encoded string literal representation with padding longer than 7 bits
✓ 2: Sends a Huffman-encoded string literal representation padded by zero
✓ 3: Sends a Huffman-encoded string literal representation containing the EOS symbol

6. Binary Format
6.1. Indexed Header Field Representation
✓ 1: Sends a indexed header field representation with index 0

6.3. Dynamic Table Size Update
✓ 1: Sends a dynamic table size update larger than the value of SETTINGS_HEADER_TABLE_SIZE

Failures:

Hypertext Transfer Protocol Version 2 (HTTP/2)
4. HTTP Frames
4.3. Header Compression and Decompression
using source address [redacted]
x 2: Sends a PRIORITY frame while sending the header blocks
-> The endpoint MUST terminate the connection with a connection error of type PROTOCOL_ERROR.
Expected: GOAWAY Frame (Error Code: PROTOCOL_ERROR)
Connection closed
Actual: Error: read tcp [redacted]:50360->[2a04:4e42::644]:443: read: connection reset by peer

Finished in 20.0796 seconds
147 tests, 146 passed, 0 skipped, 1 failed
```

H2spec Test¹⁵ sobre o domínio [apache.org](#)

#Início do doc

Este doc foi construído utilizando [Markdown](#).

FliSol 2024. HTTP/2: +Performance. @eduardoenemark.

1. https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP
2. HTTP/2 in Action, Barry Pollard. Manning Publications, 2019.
3. <https://httparchive.org/reports/state-of-the-web>
4. <https://w3techs.com/technologies/details/ce-http2>
5. <https://caniuse.com/?search=http2>
6. HTTP: The Definitive Guide, David Gourley & Brian Totty. O'Reilly, 2002.
7. <https://www.tunetheweb.com/performance-test-360>
8. Testes realizados entre 08/04 a 09/04/2024.
9. <https://http2demo.io>
10. https://developer.chrome.com/docs/devtools/network/reference/?utm_source=devtools#timing-explanation
11. <https://curl.se/docs/manpage.html>
12. <https://github.com/httpwg>
13. <https://github.com/httpwg/http2-spec/wiki/Implementations>
14. <https://github.com/httpwg/http2-spec/wiki/Tools>
15. <https://github.com/summerwind/h2spec>
16. <https://httpd.apache.org>
17. <https://nghttp2.org>
18. <https://httpwg.org/specs/rfc7540.html>