HTTP/2

Jeszenszky Péter
Debreceni Egyetem, Informatikai Kar
jeszenszky.peter@inf.unideb.hu

Utolsó módosítás: 2022. november 26.

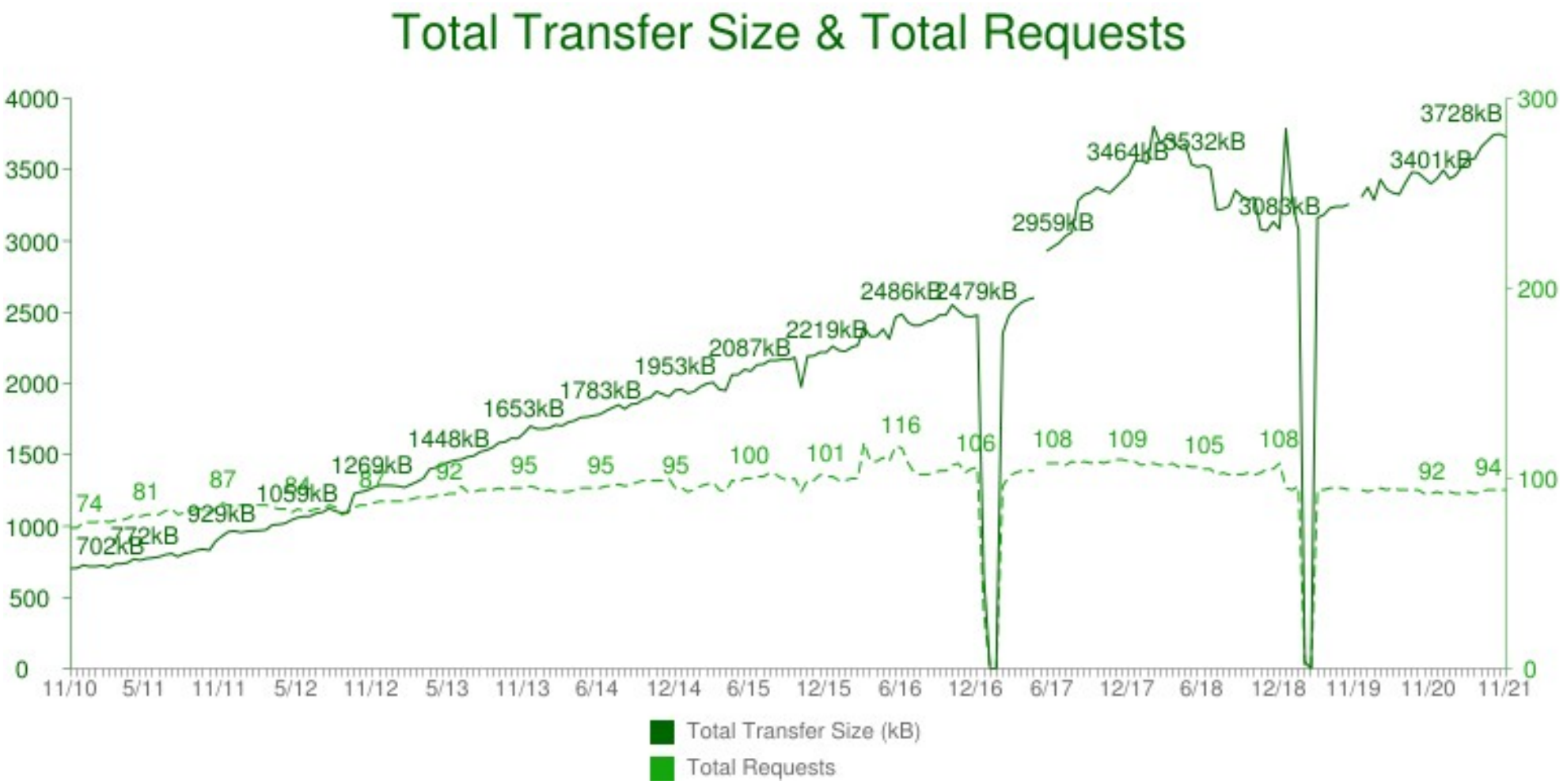
A HTTP/1.*x* a teljesítményt rontó jellemzői (1)

- A HTTP/1.0 egyetlen nem kiszolgált kérést engedett meg egy TCP kapcsolaton.

A HTTP/1.*x* a teljesítményt rontó jellemzői (2)

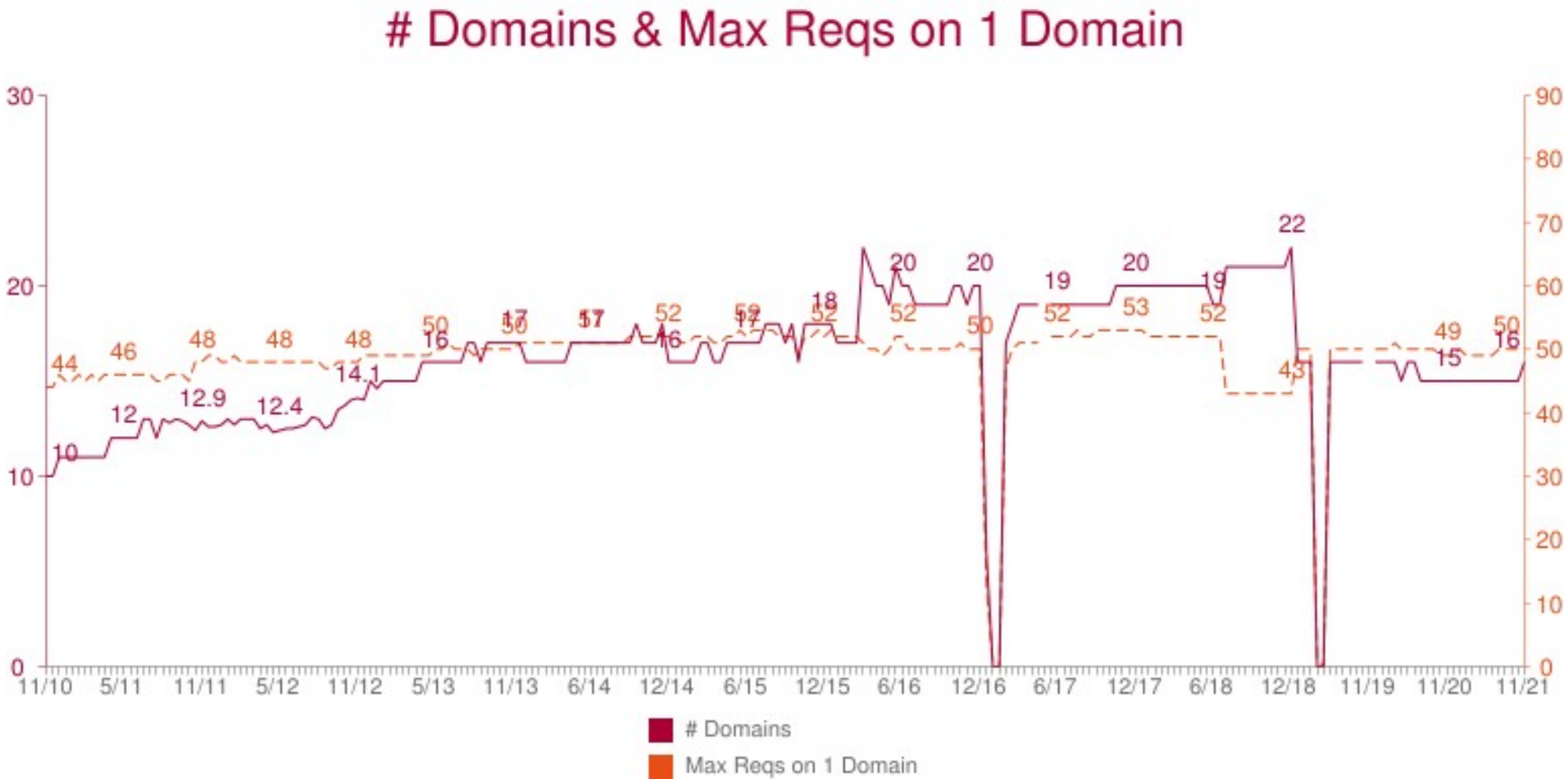
- A HTTP/1.1 bevezette a kérések sorozatban való továbbítását (*pipelining*), de ez csak részben foglalkozott a kérések párhuzamosságával és továbbra is felmerül a sor eleji blokkolás (*head-of-line blocking*) problémája.
  - Emiatt azok a HTTP kliensek, melyeknek sok kérést kell végrehajtaniuk, több kapcsolatot használnak egy szerverhez a párhuzamosság eléréséhez és a késleltetési idő csökkentéséhez.
- Ráadásul a HTTP fejlécmezők gyakran ismétlődőek és bőbeszédűek, mely szükségtelen hálózati forgalmat okoz és a kezdeti TCP torlódási ablak gyors megtelését eredményezi.

Statisztikák (1)



Az 1000 legnépszerűbb webhelyre számított statisztikák (2010. november 15. – 2021. november 1.). Forrás: https://httparchive.org/reports/state-of-the-web.

Statisztikák (2)



Az 1000 legnépszerűbb webhelyre számított statisztikák (2010. november 15. – 2021. november 1.). Forrás: https://httparchive.org/reports/state-of-the-web.

Statisztikák (3)

- A 2016 decembere és 2017 februárja között az adatokban látható rejtélyes hézagról lásd:
  - *Why is there gap between Dec 2016 and Feb 2017 trends?* https://discuss.httparchive.org/t/why-is-there-gap-between-dec-2016-and-feb-2017-trends/919

Bevált gyakorlatok a késleltetési idő csökkentésére (1)

- **Beágyazás** (***inlining***): képek közvetlen beágyazása CSS stíluslapokba.
  - `{`
    `background:`
    `url(`
    ⟨Base64-kódolt adatok⟩`) no-repeat;`
    `}`
  - Lásd: Chris Coyier, *Data URIs*, March 25, 2010. https://css-tricks.com/data-uris/

Bevált gyakorlatok a késleltetési idő csökkentésére (2)

- ***Spriting***: több kép kombinálása egyetlen képállományban.
  - Lásd: Chris Coyier, CSS Sprites: *What They Are, Why They're Cool, and How To Use Them*, October 24, 2009. https://css-tricks.com/css-sprites/
  - Példa: *MDN Web Docs* https://web.archive.org/web/20170701113221/https://developer.mozilla.org/en-US/
    - https://web.archive.org//web/20170701211407im_/https://developer.cdn.mozilla.net/static/img/logo_sprite.7d36c4a1422b.svg

Bevált gyakorlatok a késleltetési idő csökkentésére (3)

- ***Sharding***: a tartalom elosztása több szerveren.
  - Érdemes lehet például a képeket egy olyan külön webszerverről szolgáltatni, mely nem használ sütiket.
- **Összefűzés**: több CSS stíluslap és JavaScript állomány összefűzése.
- **Kicsinyítés (*minification*)**: felesleges karakterek eltávolítása CSS stíluslapokból és JavaScript állományokból anélkül, hogy az erőforrás a böngésző általi feldolgozása módosulna.
  - Például megjegyzések és *whitespace* karakterek eltávolítása.

Bevált gyakorlatok a késleltetési idő csökkentésére (4)

- **CSS kicsinyítő eszközök**:
  - *cssnano* (programozási nyelv: PostCSS; licenc: *MIT License*) https://cssnano.co/ https://github.com/cssnano/cssnano
    - Online: https://cssnano.co/playground/
  - *CSSO* (*CSS Optimizer*) (programozási nyelv: JavaScript; licenc: *MIT License*) https://github.com/css/csso
    - Online: https://css.github.io/csso/csso.html
- **JavaScript kicsinyítő eszközök**:
  - *Closure Compiler* (programozási nyelv: Java, JavaScript; licenc: *Apache License 2.0*) https://github.com/google/closure-compiler https://developers.google.com/closure/compiler/
  - *UglifyJS* (programozási nyelv: JavaScript; licenc: *Simplified BSD License*) http://lisperator.net/uglifyjs/ https://github.com/mishoo/UglifyJS2
    - Online: *JSCompress* https://jscompress.com/
- **Komplex kicsinyítő megoldások**:
  - *PageSpeed Modules* (*Apache PageSpeed*) (programozási nyelv: C++; licenc: *Apache License 2.0*) https://www.modpagespeed.com/ https://github.com/apache/incubator-pagespeed-mod https://developers.google.com/speed/pagespeed/module/
    - Apache HTTP Server és nginx modulok.

Mi a HTTP/2?

- A HTTP szemantikájának egy optimalizált kifejezésmódja.
  - Cél a hálózati erőforrások hatékonyabb használata és a végfelhasználó által megfigyelhető késleltetési idő csökkentése.
  - Az egyik fő cél, hogy lehetővé tegye a kliensek számára, hogy csupán egy kapcsolatot kelljen fenntartaniuk egy szerverhez.
- Ugyanazokat a metódusokat, állapotkódokat, fejlécmezőket és URI sémákat használja, mint a HTTP/1.1.
  - Az üzenetek formálása és átvitele történik eltérő módon.

Fejlesztés

- Az IETF HTTP munkacsoportja (httpbis) fejleszti.
  - Honlap: https://http2.github.io/

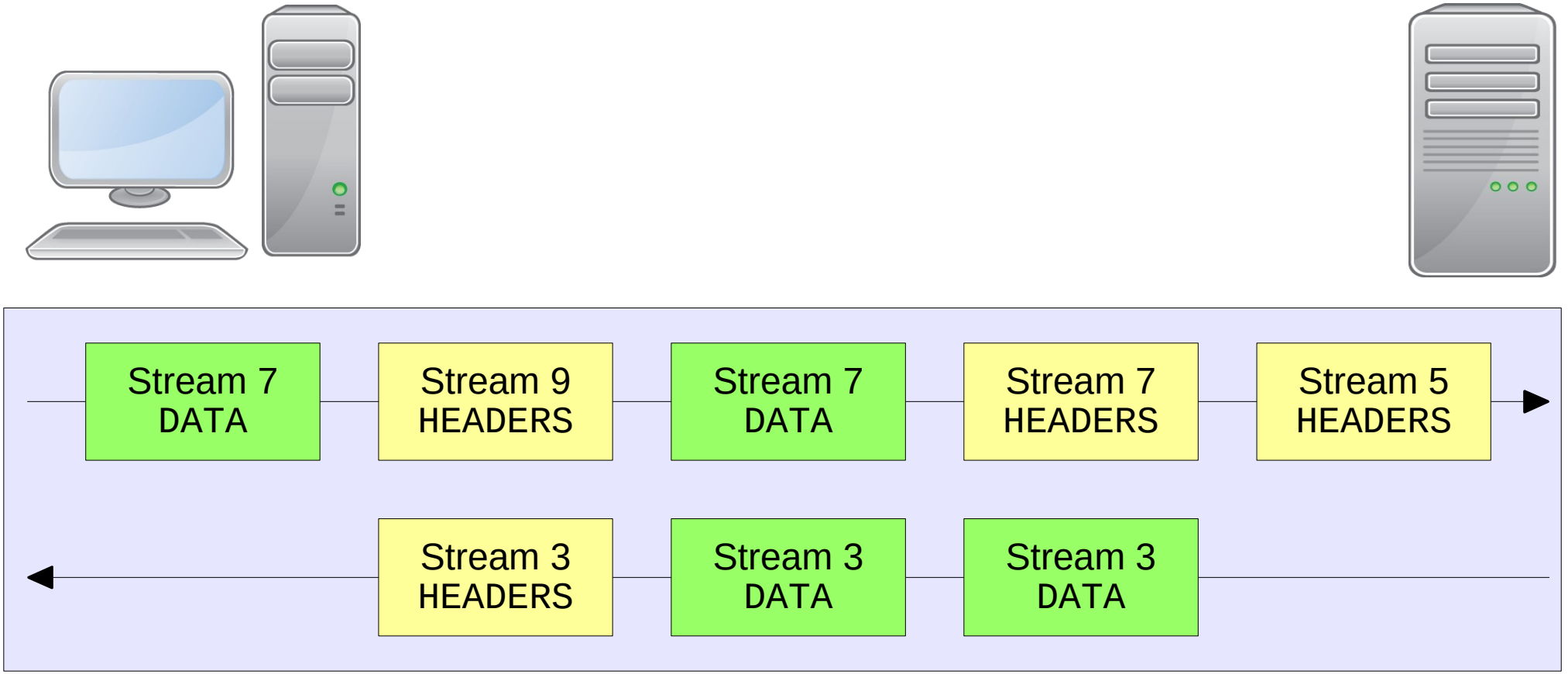
Specifikációk

- Martin Thomson (ed.), Cory Benfield (ed.). *RFC 9113: HTTP/2.* June 2022. https://www.rfc-editor.org/rfc/rfc9113
- Roberto Peon, Herve Ruellan. *RFC 7541: HPACK: Header Compression for HTTP/2.* May 2015. https://www.rfc-editor.org/rfc/rfc7541

A HTTP/2 újdonságai

- **Multiplexelés**: megvalósítás adatfolyamok használatával.
  - Az adatfolyamok túlnyomórészt egymástól függetlenek, így egy blokkolt vagy beragadt adatfolyam nem akadályozza az előrehaladást más adatfolyamokkal.
- ***Forgalomvezérlés*** és **rangsorolás**: a multiplexelt adatfolyamok hatékony használatát biztosító mechanizmusok.
  - A forgalomvezérlés biztosítja, hogy a fogadónak annyi adat kerül továbbításra, amennyit az kezelni tud.
  - A rangsorolás biztosítja, hogy a korlátozottan rendelkezésre álló erőforrások először a legfontosabb adatfolyamokhoz rendelhetők hozzá.
- **Szerver *push***: lehetővé teszi a szervernek, hogy spekulatív módon küldjön olyan adatokat egy kliensnek, melyekre annak előreláthatólag szüksége lesz.
- **Bináris protokoll**: az üzenetek hatékonyabb feldolgozását teszi lehetővé az üzenetek bináris formálása.
- **Fejlécmezők tömörítése** (**HPACK**)

Üzenet multiplexelés



Előzmény

- A HTTP/2 a Google által fejlesztett SPDY protokollon alapul. https://www.chromium.org/spdy/
  - Célkitűzések: *SPDY: An experimental protocol for a faster web* https://www.chromium.org/spdy/spdy-whitepaper/
    - A célkitűzések között szerepelt például az oldalak betöltési idejének felére csökkentése.
  - Az SPDY/2 szolgált alapul a HTTP/2 specifikációhoz.
    - Verziótörténet: https://sites.google.com/a/chromium.org/dev/spdy/spdy-protocol

Elterjedtség (1)

- HTTP/2-képes webhelyek:
  - *Dropbox* https://www.dropbox.com/
  - *Facebook* https://www.facebook.com/
  - *Flickr* https://www.flickr.com/
  - *Google* https://www.google.com/
  - *Twitter* https://twitter.com/
  - *Wikipedia* https://www.wikipedia.org/
  - *W3C* https://www.w3.org/
  - *Yahoo* https://www.yahoo.com/
  - …

Elterjedtség (2)

- *HTTP2.Pro* https://http2.pro/
  - Online eszköz szerver oldali HTTP/2 támogatás ellenőrzéséhez.

Elterjedtség (2)

- Az elterjedtségre vonatkozó statisztikák:
  - *HTTP Archive – State of the Web – HTTP/2 Requests* https://httparchive.org/reports/state-of-the-web#h2
  - *HTTP/2 Dashboard* http://isthewebhttp2yet.com/
  - *Usage of HTTP/2 for websites* https://w3techs.com/technologies/details/ce-http2/all/all
  - https://http2.netray.io/stats.html

Implementációk (1)

- Az implementációk egy listája: https://github.com/http2/http2-spec/wiki/Implementations

Implementációk (2)

- Szerverek:
  - *Apache HTTP Server* (programozási nyelv: C; licenc: *Apache License 2.0*) https://httpd.apache.org/
    - A 2.4.17 verzióban megjelent `mod_http2` modul biztosít HTTP/2 támogatást: *Overview of new features in Apache HTTP Server 2.4* https://httpd.apache.org/docs/trunk/new_features_2_4.html
  - *Apache Tomcat* (programozási nyelv: Java; licenc: *Apache License 2.0*) https://tomcat.apache.org/
    - HTTP/2 támogatás a 8.5.*x* és 9.*x* verziókban áll rendelkezésre, lásd: https://tomcat.apache.org/whichversion.html
  - *Jetty* (programozási nyelv: Java; licenc: *Apache License 2.0*/*Eclipse Public License v1.0*) https://www.eclipse.org/jetty/ https://github.com/eclipse/jetty.project
    - HTTP/2 támogatás a központi Maven tárolóból elérhető `9.3.0.M2` verziótól.
  - *nghttp2* (programozási nyelv: C; licenc: *MIT License*) https://nghttp2.org/ https://github.com/nghttp2/nghttp2
  - *nginx* (programozási nyelv: C; licenc: *Simplified BSD License*) https://nginx.org/
    - *Faisal Memon, NGINX Open Source 1.9.5 Released with HTTP/2 Support*, September 22, 2015. https://www.nginx.com/blog/nginx-1-9-5/
  - *Undertow* (programozási nyelv: Java; licenc: GPLv2.1) https://undertow.io/ https://github.com/undertow-io/undertow
    - A *WildFly* (korábbi nevén *JBoss*) alkalmazásszerver alapértelmezett webszervere. https://www.wildfly.org/

Implementációk (3)

- Böngészők: a felsorolt böngészők mindegyikében csak TLS-en keresztüli HTTP/2 támogatás (`https` URI-k)
  - **Firefox**:
    - Lásd a `network.http.spdy.enabled.http2` opciót (`about:config`).
  - **Chromium/Google Chrome/Opera**:
    - Lásd: https://chromestatus.com/feature/5125863656652 80
  - **Microsoft Edge**:
    - Lásd: https://developer.microsoft.com/en-us/microsoft-edge/status/http2/
- Lásd: https://caniuse.com/http2

Implementációk (4)

- Egyéb kliensek:
  - *curl* (programozási nyelv: C; licenc: *X11 License*) https://curl.se/ https://github.com/curl/curl
  - *hyper-h2* (programozási nyelv: Python; licenc: *MIT License*) http://python-hyper.org/projects/h2/en/stable/ https://github.com/python-hyper/hyper-h2
  - *h2i* (programozási nyelv: Go; licenc: *New BSD License*) https://github.com/golang/net/tree/master/http2/h2i
  - *Netty* (programozási nyelv: Java; licenc: *Apache License 2.0*) https://netty.io/ https://github.com/netty/netty
  - *nghttp2* (programozási nyelv: C; licenc: *MIT License*) https://nghttp2.org/ https://github.com/nghttp2/nghttp2
  - *OkHttp* (programozási nyelv: Java; licenc: *Apache License 2.0*) https://square.github.io/okhttp/ https://github.com/square/okhttp

Hivatalos Java támogatás

- Kliens oldal:
  - A JDK 11-ben megjelent `java.net.http` csomag egy olyan HTTP kliens API-t biztosít, mely támogatja a HTTP/2-t és helyettesítheti a `java.net.HttpURLConnection` osztályt.
  - Lásd:
    - *JEP 110: HTTP 2 Client* http://openjdk.java.net/jeps/110
    - https://docs.oracle.com/en/java/javase/17/docs/api/java.net.http/java/net/http/package-summary.html
- Szerver oldal:
  - A Servlet 4.0 API támogatja, mely a Java EE 8-ban jelent meg.
    - Lásd:
      - *JSR-369: Java Servlet 4.0 Specification (July 2017)* https://jcp.org/en/jsr/detail?id=369
      - http://javadoc.io/doc/javax.servlet/javax.servlet-api/4.0.0
- Lásd még:
  - Edward Burns. *HTTP/2 comes to Java – What Servlet 4.0 means to you.* DevNexus 2015, March 10–12, 2015. https://www.slideshare.net/edburns/http2-comes-to-java-what-servlet-40-means-to-you-devnexus-2015
  - Alex Theedom. *Get started with Servlet 4.0*. May 10, 2018. https://developer.ibm.com/tutorials/j-javaee8-servlet4/

Böngésző kiegészítők

- A használt HTTP verziót a címsorban jelző böngésző kiegészítők:
  - **Firefox**: *HTTP Version Indicator*
    https://addons.mozilla.org/hu/firefox/addon/http2-indicator/
  - **Chromium**, **Google Chrome**: *HTTP Indicator*
    https://chrome.google.com/webstore/detail/http-indicator/hgcomhbcacfkpffiphlmnlhpppcjgmbl

curl

- Az *nghttp2* programkönyvtárat használja a HTTP/2 támogatás megvalósításához.
  - Lásd: *HTTP/2 with curl* https://curl.se/docs/http2.html
- Példa a használatra:
  - `curl --http2 --head https://www.w3.org/`

nghttp2

- Példa a használatra:
  - `nghttp https://nghttp2.org/ -nv`

Fogalmak

- **Kliens**: egy HTTP/2 kapcsolatot létesítő végpont, a kliensek HTTP kéréseket küldenek és HTTP válaszokat kapnak.
- **Szerver**: egy HTTP/2 kapcsolatot elfogadó végpont, a szerverek HTTP kéréseket fogadnak és HTTP válaszokat küldenek.
- **Végpont** (***endpoint***): a klienst vagy a szervert jelenti.
- **Kapcsolat**: két végpont közötti átviteli rétegbeli kapcsolat.
- **Keret** (***frame***): a legkisebb kommunikációs egység egy HTTP/2 kapcsolaton belül.
- **Adatfolyam** (***stream***): egy HTTP/2 kapcsolaton belül a kliens és a szerver között váltott keretek egy független, kétirányú sorozata.

HTTP/2 kapcsolat létrehozása (1)

- A HTTP/2 ugyanazt a `http` és `https` URI sémát használja, melyeket a HTTP/1.1 is.

HTTP/2 kapcsolat létrehozása (2)

- A TCP kapcsolatot a kliens kezdeményezi.
- A kliensnek először ki kell derítenie, hogy a szerver támogatja-e a HTTP/2-t, ez eltérően történik `http` és `https` URI-knál.
  - Egy `http` URI-nál a kliens egy, az `Upgrade` fejlécmezőt tartalmazó HTTP/1.1 kérést hajt végre.
  - Egy `https` URI-nál az ALPN révén történik a protokoll egyeztetése.

HTTP/2 kapcsolat létrehozása (3)

- Példa:
  - `curl --http2 -v http://nghttp2.org/`

```
> GET / HTTP/1.1
> Host: nghttp2.org
> User-Agent: curl/7.83.1
> Accept: */*
> Connection: Upgrade, HTTP2-Settings
> Upgrade: h2c
> HTTP2-Settings: AAMAAABkAAQCAAAAAAIAAAAA
>
< HTTP/1.1 101 Switching Protocols
< Connection: Upgrade
< Upgrade: h2c
<
< HTTP/2.0 200
...
```

HTTP/2 kapcsolat létrehozása (4)

- Példa:
  - `curl --http2 -v https://www.facebook.com/`

HTTP/2 kapcsolat létrehozása (5)

- Mindkét végpontnak egy **kapcsolat bevezetőt** (***connection preface***) kell küldenie a használt protokoll végső megerősítéseként és a kapcsolat kezdeti beállításainak megállapításához.
  - A kliens bevezetője a `"PRI * HTTP/2.0\r\n\r\nSM\r\n\r\n"` oktettsorozattal kezdődik, melyet egy `SETTINGS` keret kell, hogy kövessen, mely lehet üres.
    - A bevezető választásáról lásd: Matthew Kerwin, *Painting Sheds*, 2013-12-09. https://matthew.kerwin.net.au/blog/20131209_painting_sheds
  - A szerver bevezetője egy potenciálisan üres `SETTINGS` keretből áll.
- A bevezetőt a kliens közvetlenül a `101` (`Switching Protocols`) válasz fogadása után küldi el vagy a TLS kapcsolat első alkalmazási adatoktettjeiként.
- Ha a kliens tudja, hogy a szerver támogatja a protokollt, akkor a kapcsolat létrehozásakor küldi el a bevezetőt.

ALPN (1)

- A Transport Layer Security (TLS) egy kiterjesztése alkalmazási-réteg protokoll egyeztetéséhez:
  - Stephan Friedl, Andrei Popov, Adam Langley, Emile Stephan. *RFC 7301: Transport Layer Security (TLS) – Application-Layer Protocol Negotiation Extension.* July 2014. https://www.rfc-editor.org/rfc/rfc7301
- A protokoll azonosítók regisztrálását az IANA végzi.
  - Lásd: *Application-Layer Protocol Negotiation (ALPN) Protocol Ids* https://www.iana.org/assignments/tls-extensiontype-values/tls-extensiontype-values.xhtml#alpn-protocol-ids

ALPN (2)

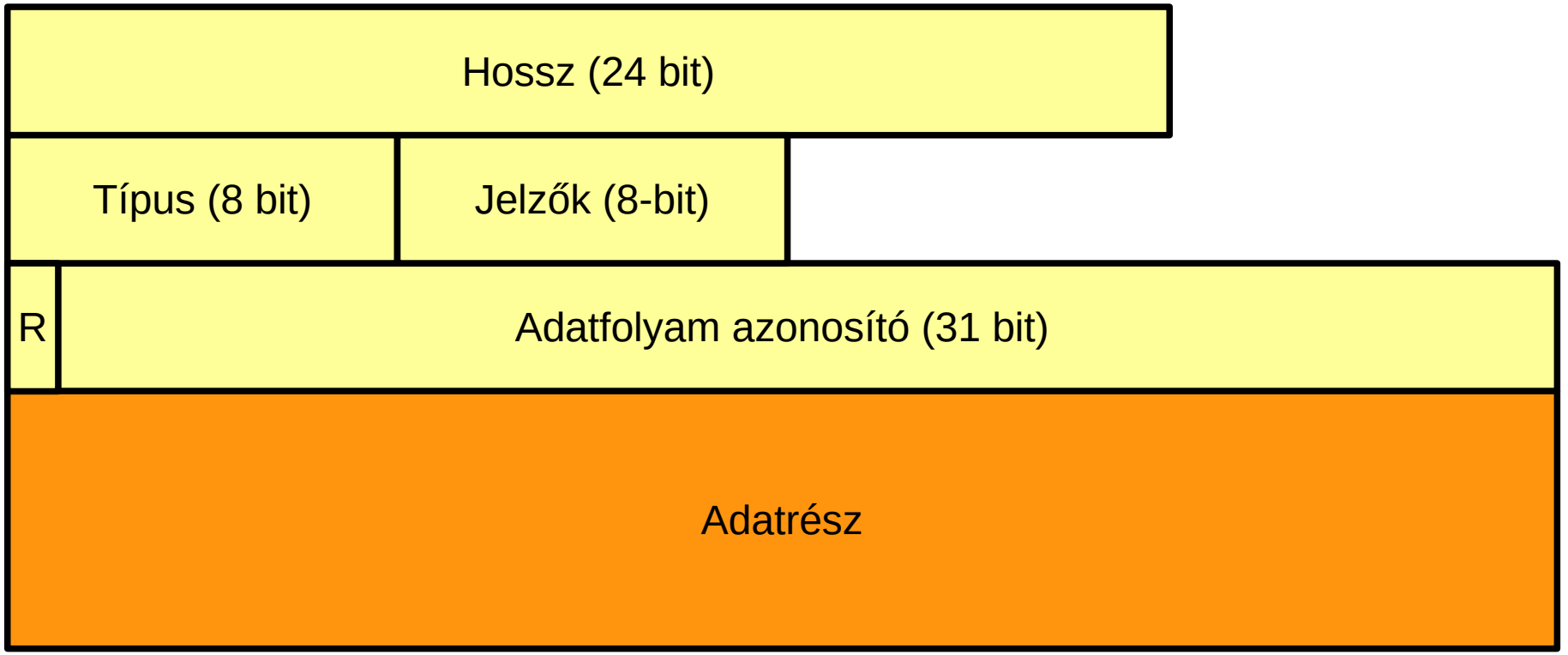
- Protokoll egyeztetés:
  - A kliens a TLS `ClientHello` üzenet részeként elküldi a szervernek az általa támogatott protokollok listáját.
  - A szerver kiválaszt egy protokollt, melyet a TLS `ServerHello` üzenet részeként küld vissza a kliensnek.
- Az alkalmazási protokoll egyeztetése így a TLS kézfogás során történik.

ALPN (3)

- A HTTP/2 által használt protokoll azonosítók:
  - `"h2"`: HTTP/2 TLS-en keresztül
  - `"h2c"`: HTTP/2 TCP-n keresztül (ahol `'c'` a *cleartext* kifejezést jelenti)

Keretek felépítése (1)

- Minden keret egy 9 oktett méretű fejléccel kezdődik, melyet egy változó hosszú **adatrész** (***payload***) követ.



Keretek felépítése (2)

- A keret fejléc mezői:
  - **Hossz**: az adatrész hosszát szolgáltató 24-bites előjel nélküli egész.
  - **Típus**: a keret 8 biten ábrázolt típusa, mely meghatározza a keret felépítését és szemantikáját.
  - **Jelzők**: 8 biten ábrázolt logikai jelzők, melyek jelentése a keret típusától függ.
  - **R**: fenntartott célú 0 értékű bit, melynek nincs definiált jelentése.
  - **Adatfolyam azonosító**: egy adatfolyamot azonosító 31-bites előjel nélküli egész.
- Az adatrész felépítése és tartalma a keret típusától függ.

Keretek fajtái (1)

Kód	Típus	Funkció
0x0	DATA	Üzenet *payload* átvitele.
0x1	HEADERS	Adatfolyam megnyitása és egy fejléc blokk töredék továbbítása.
0x2	PRIORITY	Adatfolyam prioritásának előírása.
0x3	RST_STREAM	Adatfolyam azonnali megszüntetése.
0x4	SETTINGS	Konfigurációs paraméterek továbbítása, vétel nyugtázása.

Keretek fajtái (2)

Kód	Típus	Funkció
0x5	PUSH_PROMISE	Szerver *push* megvalósítása: a másik végpont előzetes értesítése a küldő által létrehozni kívánt adatfolyamokról.
0x6	PING	Minimális körbefordulási idő (RRT) mérése, tétlen (*idle*) állapotú adatfolyam működőképességének megállapítása.
0x7	GOAWAY	Kapcsolat lezárásának kezdeményezése vagy kapcsolati hiba jelzése.
0x8	WINDOW_UPDATE	Forgalomvezérlés megvalósítása.
0x9	CONTINUATION	Fejléc blokk töredékek sorozatának folytatása.

Adatfolyamok jellemzői

- Egyetlen HTTP/2 kapcsolat több egyidejűleg nyitott adatfolyamot tartalmazhat.
- Az adatfolyamokat egyoldalúan (a másik féllel való egyeztetés nélkül) hozhatja létre a kliens vagy a szerver, de megosztva használhatják őket.
- Az adatfolyamokat mindkét végpont lezárhatja.
- Lényeges, hogy milyen sorrendben kerülnek elküldésre a keretek egy adatfolyamon.
  - A fogadó abban a sorrendben dolgozza fel a kereteket, melyben megkapja őket.
- Az adatfolyamokat egy előjel nélküli egész szám azonosítja.

Adatfolyamok azonosítása (1)

- Az adatfolyam azonosító egy 31-bites előjel nélküli egész, melyet az adatfolyamot létrehozó végpont rendel az adatfolyamhoz.
  - A kliens által megnyitott adatfolyamokat páratlan számok azonosítják.
  - A szerver által megnyitott adatfolyamokat páros számok azonosítják.
  - A `0x00` adatfolyam azonosítót a kapcsolatot vezérlő üzenetekhez használják.
- Egy újonnan létrehozott adatfolyam azonosítója nagyobb kell, hogy legyen a kezdeményező végpont által létrehozott vagy fenntartott adatfolyam azonosítók mindegyikénél.

Adatfolyamok azonosítása (2)

- Az adatfolyam azonosítók nem újrafelhasználhatóak.
- Hosszú élettartamú kapcsolatoknál kimerülhet a rendelkezésre álló adatfolyam azonosítók tartománya.
  - Ilyenkor egy új kapcsolatot kell létrehozni.

HTTP kérés/válasz váltás

- Egy kliens minden egyes HTTP kérést egy új adatfolyamon küld el a szervernek, melyhez egy előzőleg nem használt adatfolyam azonosítót használ.
- A szerver ugyanezen az adatfolyamon küldi vissza a választ.
- A kérés/válasz váltás során az adatfolyam teljesen „elhasználódik”.
- A választ záró keret lezárja az adatfolyamot.

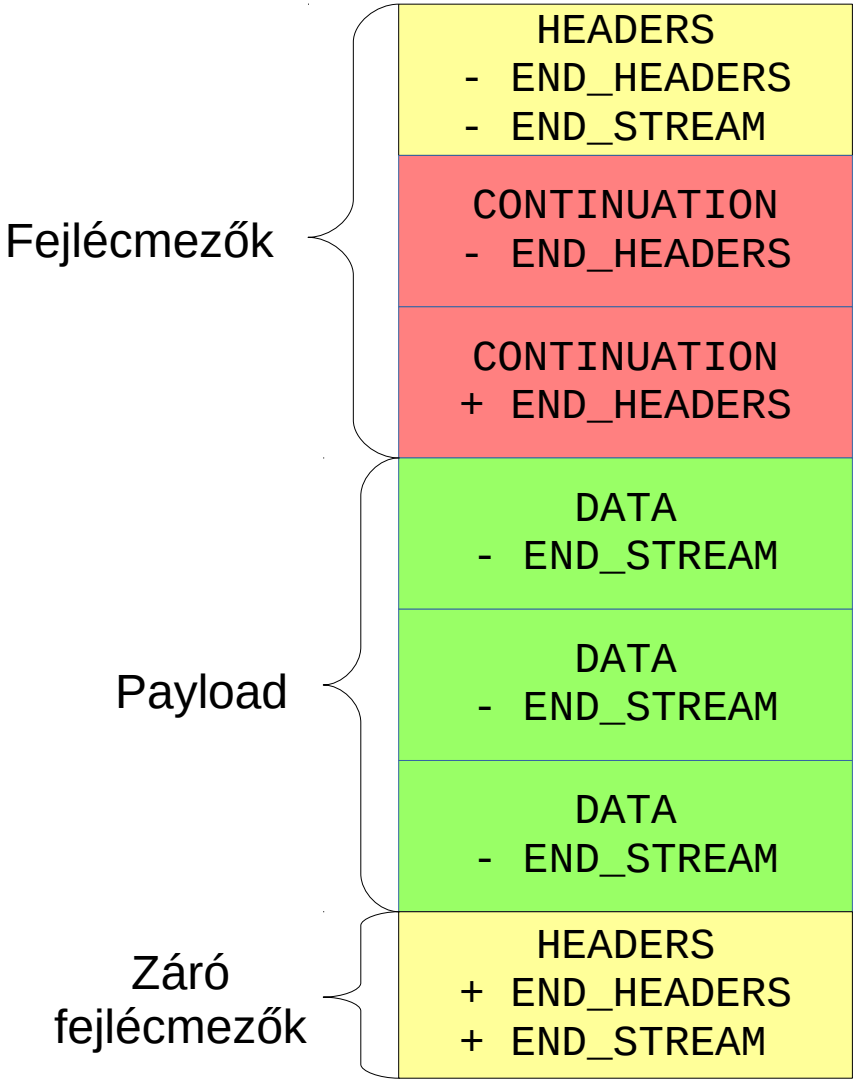
HTTP/2 üzenetek felépítése (1)

- Egy HTTP üzenet a következőkből áll:
  - Csak válaszoknál 0 vagy több `HEADERS` keretből, melyek mindegyikét 0 vagy több `CONTINUATION` keret követi, és melyek 1*xx* állapotkódú válaszok üzenet fejléceit tartalmazzák.
  - Egy `HEADERS` keretből és azt követő 0 vagy több `CONTINUATION` keretből, melyek az üzenet fejléceket tartalmazzák.
  - 0 vagy több, a *payload* törzset tartalmazó `DATA` keretből.
  - Opcionálisan egy `HEADERS` keretből és azt követő 0 vagy több `CONTINUATION` keretből.

HTTP/2 üzenetek felépítése (2)

- A sorozat utolsó keretében be van kapcsolva az `END_STREAM` jelző.
- A `HEADERS` és a `CONTINUATION` keretek között nem fordulhatnak elő más keretek (egyetlen adatfolyamból sem).
- Tilos a `chunked` átviteli kódolás használata a HTTP/2-ben.
  - A HTTP/2 `DATA` kereteket használ az üzenet *payload* továbbításához, az átvitel így valójában történhet darabokban.

HTTP/2 üzenetek felépítése (3)



Fejlécmezők

- A fejlécmezők nevét kisbetűssé kell alakítani a HTTP/2 szerinti kódolásuk előtt.

Pszeudo-fejlécmezők (1)

- A HTTP/1.*x* üzenetek kezdősorában adott információk ábrázolásához a HTTP/2 `':'` karakterrel kezdődő speciális pszeudo-fejlécmezőket használ.
- A pszeudo-fejlécmezők nem HTTP fejlécmezők.

Pszeudo-fejlécmezők (2)

- Kérés pszeudo-fejlécmezők:
  - **`:method`**: a HTTP metódust tartalmazza.
  - **`:scheme`**: a cél URI séma részét tartalmazza.
  - **`:authority`**: a cél URI autoritás részét tartalmazza, a `Host` fejlécmező megfelelője.
    - HTTP/2 kéréseket generáló kliensek számára az `:authority` pszeudo-fejlécmező használata ajánlott a `Host` fejlécmező helyett.
  - **`:path`**: a cél URI útvonal és lekérdezés részét tartalmazza.
    - Szerver-szintű `OPTIONS` kéréseknél `'*'` az értéke.
- Válasz pszeudo-fejlécmezők:
  - **`:status`**: a HTTP állapotkódot hordozza, minden válasznak tartalmaznia kell.

Pszeudo-fejlécmezők (3)

- A pszeudo-fejlécmezők meg kell, hogy előzzék a rendes fejlécmezőket.
- `CONNECT` kérések kivételével minden HTTP/2 kérés pontosan egy érvényes értéket kell, hogy tartalmazzon a `:method`, `:scheme` és `:path` pszeudo-fejlécmezőkhöz.
- A HTTP/2 nem teszi lehetővé:
  - Kérésben a protokoll verzió továbbítását.
  - Válaszban a protokoll verzió és az indok frázis továbbítását.

Példa (1)

```
> GET /index.html HTTP/1.1
> User-Agent: curl/7.83.1
> Host: eg.com
> Accept: */*
>
```

```
HEADERS (stream_id=1)
  + END_STREAM
  + END_HEADERS
  :method = GET
  :path = /index.html
  :scheme = https
  :authority = eg.com
  user-agent = curl/7.83.1
  accept = */*
```

```
< HTTP/1.1 200 OK
< ETag: "54ef4a17"
< Content-Length: 8192
< Content-Type: text/html
<
< ⟨adatok⟩
```

```
HEADERS (stream_id=1)
  - END_STREAM
  + END_HEADERS
  :status = 200
  etag = "54ef4a17"
  content-length = 8192
  content-type = text/html

DATA (stream_id=1)
  + END_STREAM
  ⟨adatok⟩
```

Példa (2)

```
> GET /index.html HTTP/1.1
> User-Agent: curl/7.83.1
> Host: eg.com
> Accept: */*
> If-None-Match: "54ef4a17"
>
```

```
HEADERS (stream_id=1)
  + END_STREAM
  + END_HEADERS
  :method = GET
  :path = /index.html
  :scheme = https
  :authority = eg.com
  user-agent = curl/7.83.1
  accept = */*
  if-none-match = "54ef4a17"
```

```
< HTTP/1.1 304 Not Modified
< ETag: "54ef4a17"
<
```

```
HEADERS (stream_id=1)
  + END_STREAM
  + END_HEADERS
  :status = 304
  etag = "54ef4a17"
```

Példa (3)

```
> PUT /image HTTP/1.1
> User-Agent: curl/7.83.1
> Host: eg.com
> Accept: */*
> Content-Length: 8192
> Content-Type: image/png
>
> ⟨adatok⟩
```

```
HEADERS (stream_id=1)
  - END_STREAM
  + END_HEADERS
  :method = PUT
  :path = /image
  :scheme = https
  :authority = eg.com
  user-agent = curl/7.83.1
  accept = */*
  content-length = 8192
  content-type = image/png

DATA (stream_id=1)
  + END_STREAM
  ⟨adatok⟩
```

Szerver push (1)

- Lehetővé teszi a szervernek a kliens egy korábbi kérésével kapcsolatban kéretlen válaszok küldését.
  - Ez hasznos lehet akkor, amikor a szerver tudja, hogy kliensnek szüksége lesz ezekre a válaszokra az eredeti kérésére adott válasz teljes feldolgozásához.
- A szerver ehhez egy kérést szintetizál, melyet egy `PUSH_PROMISE` keretként küld el. A szerver ezután egy külön adatfolyamon képes egy választ küldeni a szintetikus kérésre.

Szerver push (2)

- A küldő egy `PUSH_PROMISE` keretet használ a fogadó arról való értesítéséhez, hogy egy adatfolyamot szándékozik létrehozni egy kéretlen válasz küldéséhez.
- A `PUSH_PROMISE` keretet egy vagy több `CONTINUATION` keret követheti.
- A `PUSH_PROMISE` és a rá következő `CONTINUATION` keretek együtt kérés fejlécmezők egy olyan teljes sorozatát tartalmazzák, melyet a szerver a szintetikus kérésnek tulajdonít.
- A `PUSH_PROMISE` keret tartalmazza annak az adatfolyamnak az azonosítóját is, melyet a végpont létrehozni szándékozik.

Szerver push (3)

- Az ígért válaszok mindig a kliens egy explicit kéréséhez tartoznak, a szerver az eredeti kéréshez tartozó adatfolyamon keresztül küldi a `PUSH_PROMISE` kereteket.
- A szerver számára az ígért válaszokra hivatkozó keretek küldése előtt ajánlott PUSH_PROMISE keretek küldése.
- A `PUSH_PROMISE` keret küldése után kezdheti meg a szerver az ígért válasz küldését azon az általa létrehozott adatfolyamon, melyhez az ígért adatfolyam azonosítót tartozik.

Szerver push (4)

- A kliens kérheti a letiltását.
  - A `SETTINGS_ENABLE_PUSH` beállítás 0 értéke jelzi a szerver *push* letiltást.
    - A kezdőértéke 0, mely azt jelzi, hogy engedélyezett a szerver *push*.
  - Tilos `PUSH_PROMISE` keret küldése egy végpont számára, ha a paramétert 0 értékre állítva kapja meg.
- A `PUSH_PROMISE` keretek fogadói egy `RST_STREAM` keret küldésével utasíthatnak el ígért adatfolyamokat.

Szerver push (5)

- Példa:

```
HEADERS (stream_id=1)
  + END_STREAM
  + END_HEADERS
  :method = GET
  :path = /index.html
  :scheme = https
  :authority = example.com
  accept = */*
```

```
PUSH_PROMISE (stream_id=1,
  promised stream id=2)
  + END_HEADERS
  :method = GET
  :path = /style.css
  :scheme = https
  :authority = example.com
  accept = */*

HEADERS (stream_id=1)
  - END_STREAM
  + END_HEADERS
  :status = 200
  content-length = 8192
  content-type = text/html

DATA (stream_id=1)
  + END_STREAM
  ⟨adatok⟩
```

Szerver push (6)

- Példa (folytatás):

```
HEADERS (stream_id=2)
  - END_STREAM
  + END_HEADERS
  :status = 200
  content-length = 1024
  content-type = text/css

DATA (stream_id=2)
  + END_STREAM
  ⟨adatok⟩
```

Szerver push (7)

- Szerver oldali támogatás:
  - *Apache HTTP Server*: igen
    - Lásd: *Apache Module mod_http2 – H2Push Directive* https://httpd.apache.org/docs/current/mod/mod_http2.html#h2push
  - *Apache Tomcat*: igen
    - Lásd:
      - *Apache Tomcat 8 – Changelog* https://tomcat.apache.org/tomcat-8.5-doc/changelog.html
      - *Apache Tomcat 9 – Changelog* https://tomcat.apache.org/tomcat-9.0-doc/changelog.html
  - *Jetty*: igen
    - Lásd: *HTTP/2 Push of Resources* https://www.eclipse.org/jetty/documentation/jetty-11/programming-guide/index.html#pg-server-http2-push
  - *nghttp2*: igen
    - Lásd: https://nghttp2.org/blog/2015/02/10/nghttp2-dot-org-enabled-http2-server-push/
  - *nginx*: igen
    - Lásd: *Introducing HTTP/2 Server Push with NGINX 1.13.9* https://www.nginx.com/blog/nginx-1-13-9-http2-server-push/
  - *Undertow*: igen
    - Lásd: https://undertow.io/javadoc/2.1.x/io/undertow/UndertowOptions.html#HTTP2_SETTINGS_ENABLE_PUSH

Szerver push (8)

- Kliens oldali támogatás:
  - *curl*: nem
    - Lásd: https://curl.se/docs/http2.html
  - *Hyper*: igen
  - *Netty*: igen
  - *nghttp2*: igen
  - *OkHttp*: nem
    - Lásd: https://github.com/square/okhttp/issues/4156

Szerver push (9)

- Böngésző támogatás:
  - **Firefox**:
    - Lásd a `network.http.spdy.allow-push` opciót (`about:config`).
  - **Chromium/Google Chrome/Opera**:
    - Lásd: *Chrome DevTools – Network features reference* https://developer.chrome.com/docs/devtools/network/reference/
  - **Microsoft Edge**:
    - Lásd: https://developer.microsoft.com/en-us/microsoft-edge/status/http2serverpush/

Szerver push (10)

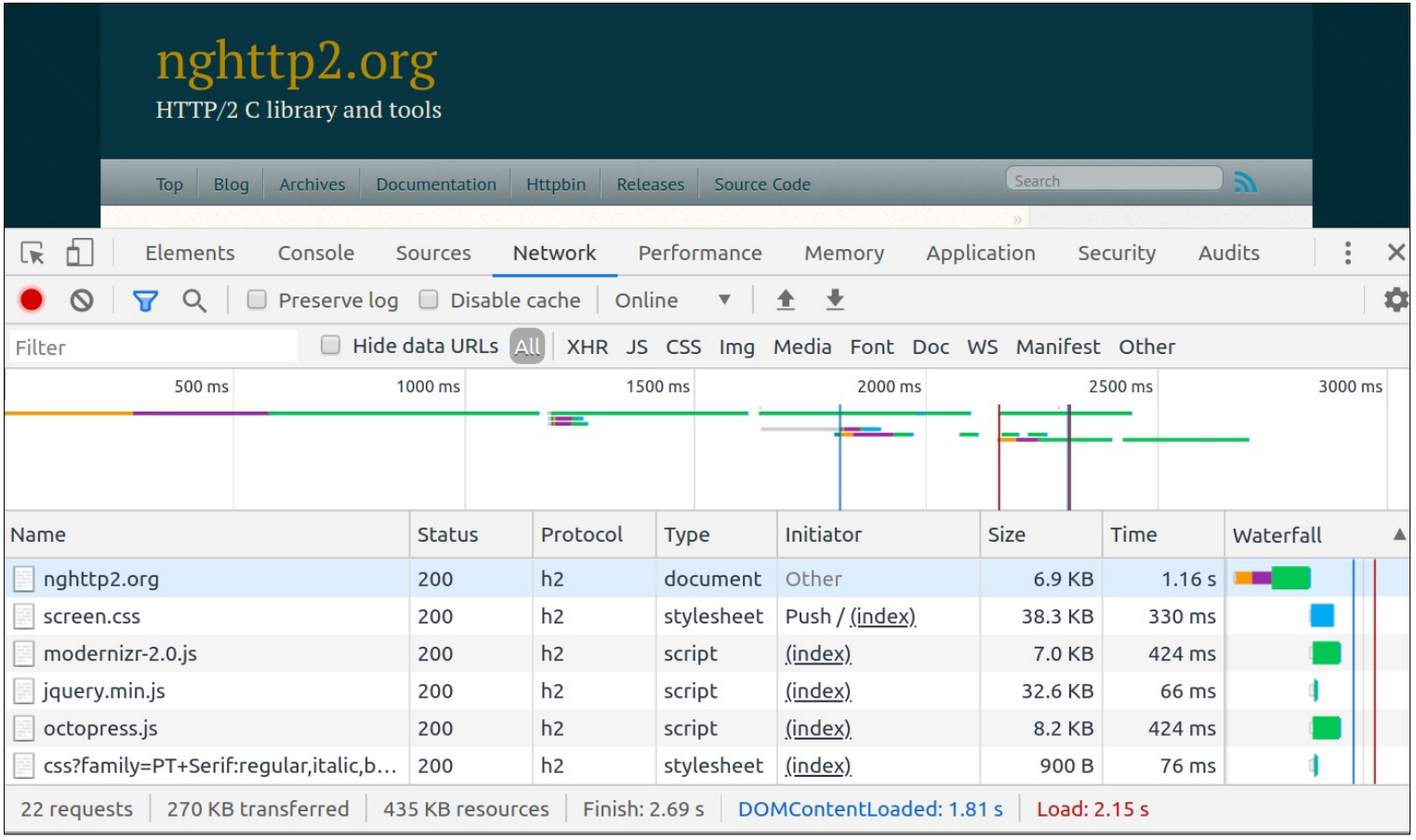
- Gyakorlati megvalósítás:
  - ***Apache HTTP Server***:
    - *Apache HTTP Server Version 2.4 – HTTP/2 guide – Server Push* https://httpd.apache.org/docs/current/howto/http2.html#push
  - ***nginx***:
    - Owen Garrett, *Introducing HTTP/2 Server Push with NGINX 1.13.9*, February 20, 2018. https://www.nginx.com/blog/nginx-1-13-9-http2-server-push/

Szerver push (11)

- Példa:
  - `nghttp https://nghttp2.org/ -nv`

Szerver push (12)

- Példa: https://nghttp2.org/ (Google Chrome DevTools)



Kapcsolatkezelés

- A HTTP/2 kapcsolatok perzisztensek.
- A kliensek számára nem ajánlott egy adott hoszt egy adott portjához egynél több HTTP/2 kapcsolatot nyitni.

HPACK

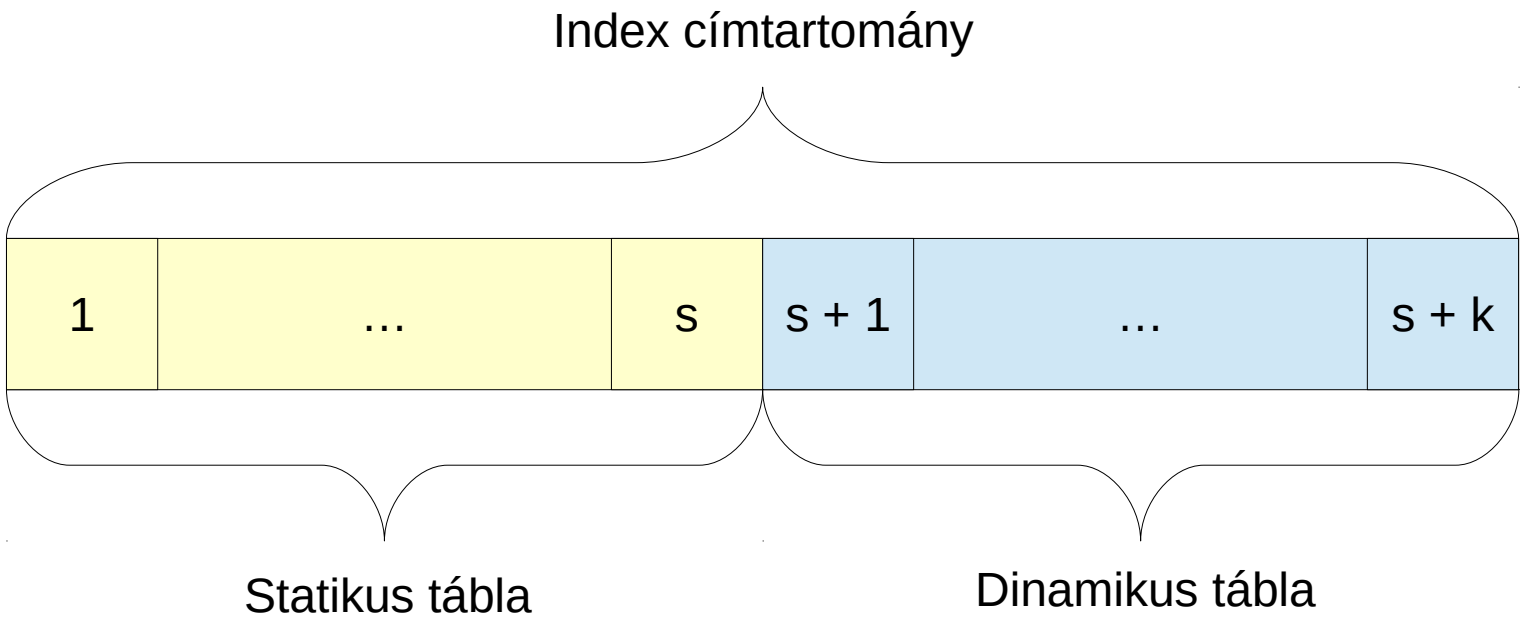
- HTTP fejlécmezők hatékony ábrázolására szolgáló tömörítési formátum a HTTP/2-höz.

Indextáblák (1)

- A fejlécmezők kódolása két indextábla használatával történik, melyek a fejlécmezőknek indexeket feleltetnek meg.
  - **Statikus tábla**: a gyakran előforduló fejlécmezőkhöz statikusan indexeket hozzárendelő, a specifikáció által előre meghatározott csak olvasható tábla.
  - **Dinamikus tábla**: a kódoló/dekódoló által kezelt kezdetben üres tábla a statikus táblában nem szereplő ismétlődő fejlécmezők indexeléséhez.
    - FIFO módon kezelt.
    - Maximális mérete korlátozható.
    - Egy végpont által kezelt kódoló és dekódoló dinamikus táblák teljesen függetlenek, azaz külön dinamikus táblákat használnak a kérésekhez és válaszokhoz.

Indextáblák (2)

- A statikus és a dinamikus tábla egyetlen index címtartományt alkot.



A statikus indextábla

Index	Név	Érték
1	:authority	
2	:method	GET
3	:method	POST
4	:path	/
5	:path	/index.html
6	:scheme	http
7	:scheme	https
8	:status	200
9	:status	204
…	…	…
58	user-agent	
59	vary	
60	via	
61	www-authenticate	

Fejlécmezők ábrázolása

- Egy fejlécmező kétféle módon ábrázolható a kódolásban:
  - **Indexként**: a statikus vagy dinamikus indextábla egy bejegyzésére hivatkozó indexként.
  - **Literálisan**: a fejlécmező nevének és értékének megadásával.
    - A fejlécmező neve ábrázolható literálisan vagy valamelyik indextábla egy bejegyzésére hivatkozó indexként.
    - A fejlécmező értékének ábrázolása literálisan történik.
- Fejlécmező nevének és értékének literális ábrázolása történhet közvetlenül vagy egy statikus Huffman-kóddal.

Huffman-kódolás

- Az RFC 7541 egy statikus Huffman-kódot határoz meg.

Decimális érték	Karakter	Kód	Hossz (bit)
…	…	…	…
32	' '	|010100	6
33	'!'	|11111110|00	10
34	'"'	|11111110|01	10
35	'#'	|11111111|1010	12
36	'$'	|11111111|11001	13
37	'%'	|010101	6
38	'&'	|11111000	8
39	'''	|11111111|010	11
…	…	…	…

Példa (1)

Index	Név	Érték
1	:authority	
2	:method	GET
3	:method	POST
4	:path	/
5	:path	/index.html
6	:scheme	http
7	:scheme	https
8	:status	200
9	:status	204
…	…	…
58	user-agent	
59	vary	
60	via	
61	www-authenticate	

:method	GET
:scheme	https
:path	/index.html
:authority	www.example.com
user-agent	my-user-agent
accept	*/*

2	
7	
5	
1	Huffman("www.example.com")
58	Huffman("my-user-agent")
19	Huffman("*/*")

Példa (2)

- HTTP/1.1 kérés mérete: 91 oktett

```
> GET /index.html HTTP/1.1
> Host: www.example.com
> User-Agent: my-user-agent
> Accept: */*
>
```

- Ekvivalens HTTP/2 kérés mérete (Huffman-kódolás): 9 + 6 + 34 = 49 oktett

```
HEADERS
  + END_STREAM
  + END_HEADERS
  :method = GET
  :scheme = http
  :path = /index.html
  :authority = www.example.com
  user-agent: my-user-agent
  accept = */*
```

Példa (3)

```
< HTTP/1.1 200 OK
< Date: Wed, 03 Oct 2018 11:45:09 GMT
< Last-Modified: Wed, 03 Oct 2018 11:45:09 GMT
< Content-Length: 4096
< Cache-Control: public, max-age=300
< Expires: Wed, 03 Oct 2018 11:50:09 GMT
< Content-Type: text/html
<
```

```
HEADERS
  - END_STREAM
  + END_HEADERS
  :status = 200
  date: Wed, 03 Oct 2018 11:45:09 GMT
  last-modified: Wed, 03 Oct 2018 11:45:09 GMT
  content-length: 4096
  cache-control: public, max-age=300
  expires: Wed, 03 Oct 2018 11:50:09 GMT
  content-type: text/html
```

- Az állapotsor és fejlécmezők mérete: 225 oktett
- Ekvivalens HEADERS keret mérete (Huffman-kódolás): 9 + 6 + 103 = 118 oktett

Teljesítmény

- Példa:
  - *HTTP/2 Technology Demo* http://www.http2demo.io/

További ajánlott irodalom

- Daniel Stenberg. *http2 explained*. https://daniel.haxx.se/http2/ https://github.com/bagder/http2-explained
- Ilya Grigorik. *High Performance Browser Networking*. O'Reilly, 2013. https://hpbn.co/
- Ilya Grigorik. *Introduction to HTTP/2*. https://web.dev/performance-http2/
- Jeremy Wagner. *A Comprehensive Guide To HTTP/2 Server Push.* April 10, 2017. https://www.smashingmagazine.com/2017/04/guide-http2-server-push/