Napredni razvoj programske potpore za web

- predavanja -

2021./2022.

HTTP2

Creative Commons

- slobodno smijete:
  - **dijeliti** — umnožavati, distribuirati i javnosti priopćavati djelo
  - **prerađivati** djelo
- pod sljedećim uvjetima:
  - **imenovanje:** morate priznati i označiti autorstvo djela na način kako je specificirao autor ili davatelj licence (ali ne način koji bi sugerirao da Vi ili Vaše korištenje njegova djela imate njegovu izravnu podršku).
  - **nekomercijalno:** ovo djelo ne smijete koristiti u komercijalne svrhe.
  - **dijeli pod istim uvjetima:** ako ovo djelo izmijenite, preoblikujete ili stvarate koristeći ga, preradu možete distribuirati samo pod licencom koja je ista ili slična ovoj.

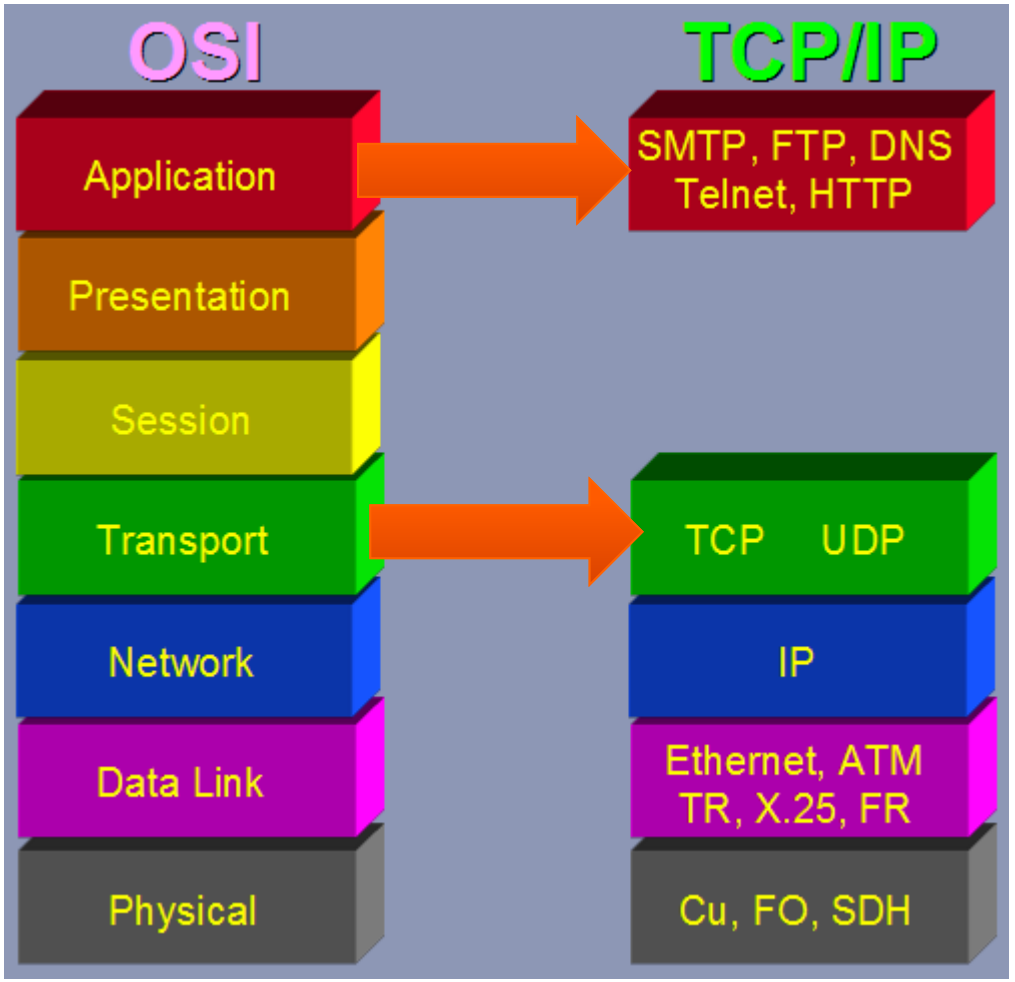
U slučaju daljnjeg korištenja ili distribuiranja morate drugima jasno dati do znanja licencne uvjete ovog djela.
Od svakog od gornjih uvjeta moguće je odstupiti, ako dobijete dopuštenje nositelja autorskog prava.
Ništa u ovoj licenci ne narušava ili ograničava autorova moralna prava. *Tekst licence preuzet je s http://creativecommons.org/*

Uvod (1)

- HyperText Transfer Protocol (HTTP) je internetski protokol aplikacijskog sloja koji definira format i način razmjene poruka između klijenta i poslužitelja. Protokol HTTP može biti izveden na bilo kojoj mreži temeljenoj na internetskim protokolima i može prenositi različite vrste podataka, a proširiv je i prema novim formatima podataka.
- HTTP definira dvije vrste poruka – zahtjev i odgovor.
  - Zahtjev definira metodu, resurs i protokol, dok odgovor sadrži ishod zahtjeva opisan statusnim kodom, i ovisno o vrsti zahtjeva i ishodu, sadržaj traženog resursa.
- Od 1999. godine je važeća verzija protokola HTTP verzija HTTP/1.1, specificirana u RFC-u 2616.

Uvod (2)


APLIKACIJSKI SLOJ
PREZENTACIJSKI SLOJ
SJEDNIČKI SLOJ
PRIJENOSNI SLOJ
MREŽNI SLOJ
PODATKOVNI SLOJ
FIZIČKI SLOJ


OSI
TCP/IP
Application
Presentation
Session
Transport
Network
Data Link
Physical
SMTP, FTP, DNS
Telnet, HTTP
TCP UDP
IP
Ethernet, ATM
TR, X.25, FR
Cu, FO, SDH

HTTP/2 u odnosu na HTTP/1.1 (1)

- Cilj pri izradi HTTP/2 protokola bio je otkloniti mnoge nedostatke HTTP/1.1 te povećati učinkovitost i sigurnost.
- Glavna razlika između HTTP/2 i HTTP/1.1 verzija HTTP protokola je u načinu na koji se podaci grupiraju u okvire i prenose između klijenta i poslužitelja.

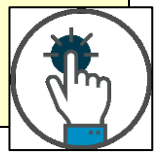
- Značajne razlike – uvodno (1):
  - HTTP/2 je binarni protokol, dok HTTP/1.1 podatke prenosi kao niz znakova
  - omogućeno je multipleksiranje više tokova podataka („streamova”) preko jedne veze. To znači da klijent može poslati više zahtjeva na istoj vezi, a poslužitelj može odgovarati u bilo kojem redoslijedu, odnosno kako odgovori postanu spremni.
  - Uvedena je HPACK kompresija zaglavlja koja uvelike sprječava redundantnost zaglavlja koja je sveprisutna kod ranijih verzija HTTP protokola

HTTP/2 u odnosu na HTTP/1.1 (2)

- Značajne razlike – uvodno (2):
  - *server push* – poslužitelji mogu unaprijed poslati resurse koji će se uskladištiti iako ih klijent nije eksplicitno zatražio. To omogućuje poslužitelju da „predvidi“ resurse koje će klijent zatražiti sljedeće te tako uštedi jedan obilazak (RTT – Round Trip Time)
  - prioritizacija zahtjeva
  - kontrola toka na razini veze i na razini *streama*
  - HTTP/2 je unio različite vrste okvira (DATA, HEADERS, PRIORITY, RST_STREAM, SETTINGS, PUSH_PROMISE, PING, GOAWAY, WINDOW_UPDATE, CONTINUATION).

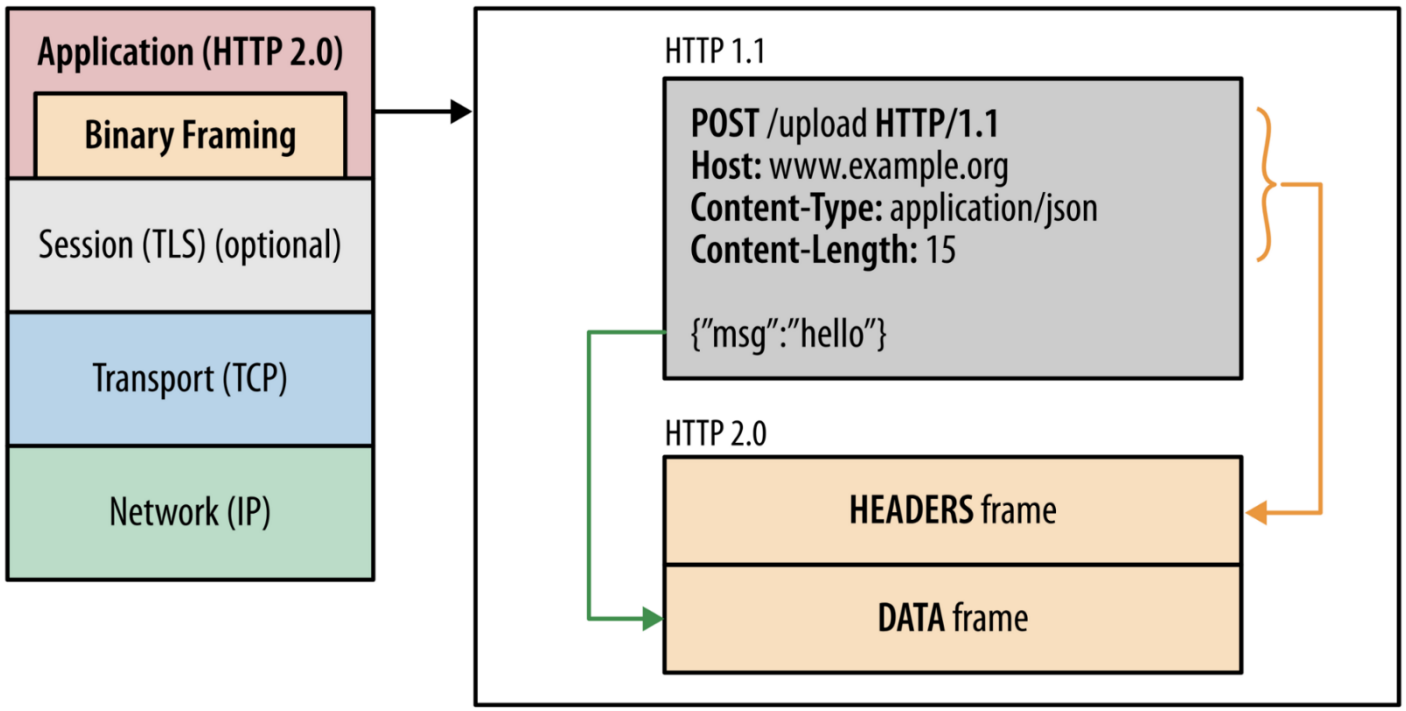
Jednostavna usporedba performansi HTTP/1.x i HTTP/2

http://www.http2demo.io/
https://http2.akamai.com/demo



Binarni prijenos (1)

- U središtu svih poboljšanja performansi HTTP/2 je novi binarni sloj uokvirivanja (*binary framing layer*), koji određuje kako se HTTP poruke inkapsuliraju i prenose između klijenta i poslužitelja.

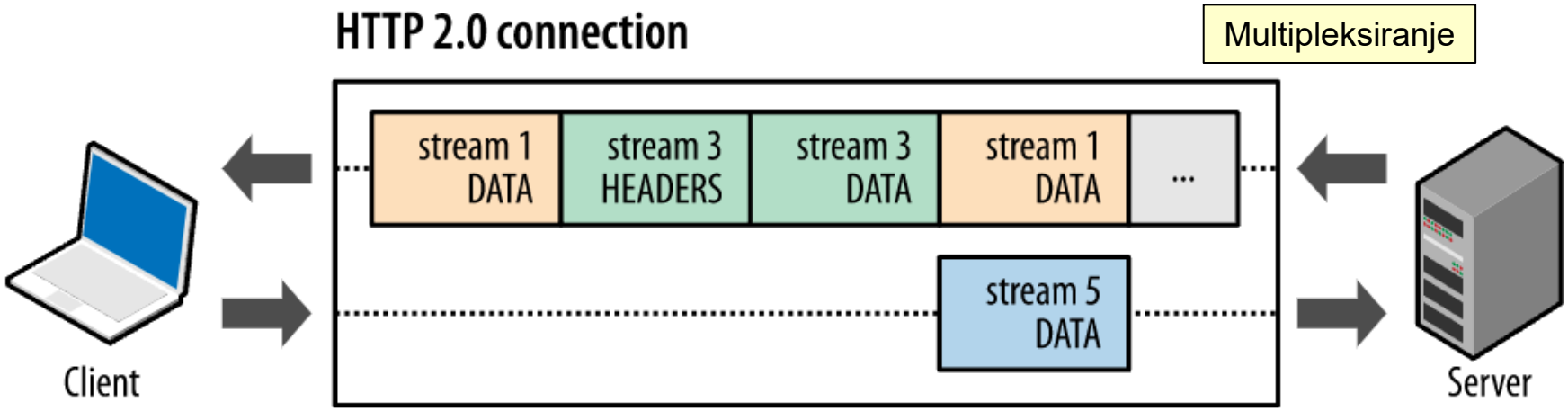


Binarni prijenos (2)

- Naziv „sloj" se odnosi na odabir dizajna za uvođenje novog optimiziranog mehanizma kodiranja poruka između utičnice i aplikacije.
  - HTTP semantika je ostala nepromijenjena.
  - Ali za razliku od protokola HTTP/1.x s otvorenim tekstom razgraničenog novim retkom, sva je HTTP/2 komunikacija podijeljena na manje poruke i okvire, od kojih je svaki kodiran u binarnom formatu.
- Klijent i poslužitelj moraju koristiti novi mehanizam binarnog kodiranja kako bi ispravno razmijenili poruke.
  - HTTP/1.x klijent nije kompatibilan sa serverom koji isključivo podržava HTTP/2 protokol, i obrnuto.

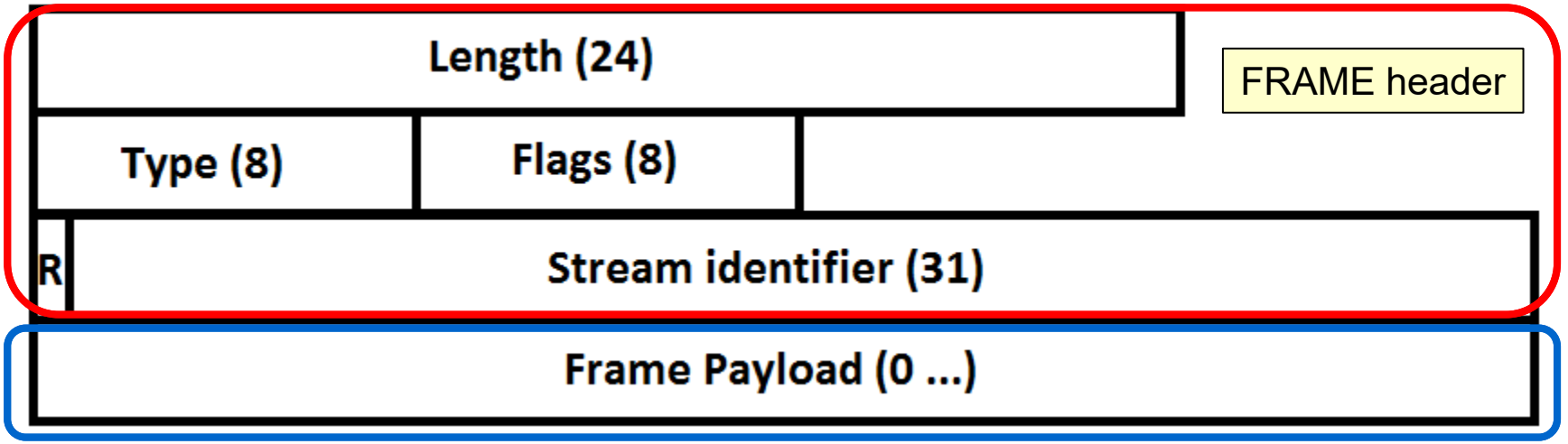
HTTP/2 konekcija

- Uvođenjem novog mehanizma binarnog uokvirivanja (*framing*) mijenja se način razmjene podataka između klijenta i poslužitelja.
- HTTP/2 rječnik:
  - Tok (*stream*): Dvosmjerni (bidirekcionalni) tok podataka kroz uspostavljenu vezu koji sadržava barem jednu poruku.
  - Poruka (*message*): Cjeloviti niz okvira koji mapiraju logički zahtjev i poruku odgovor.
  - Okvir (*frame*): Najmanji element komunikacije HTTP/2 protokolom. Sadržava zaglavlje okvira (*frame header*) i identifikator toka (ID) kojemu okvir pripada. Sadržava binarne podatke.



Binarni okviri

- Nakon uspostave konekcije, klijent i poslužitelj komuniciraju razmjenom okvira.
  - *Binary framing* je važno unapređenje u odnosu na HTTP1/1.x
- Svi okviri imaju zaglavlje (FRAME header) dužine 9 bajta:
  - Length: dužina okvira = 24 bit
    - Jedan okvir može prenijeti do 2^{24} bitova (~16MB) podataka, ipak zbog optimizacije veličina je do 2^{14} , a veći okvir klijent i poslužitelj moraju dogovoriti (SETTINGS_MAX_FRAME_SIZE)
  - Type: tip i semantika okvira, nepoznati će biti odbačeni = 8 bit
  - Flags: boolean zastavice = 8 bit
  - R: Rezervirano = 1 bit (uvijek vrijednost 0)
  - Stream identifer: jedinstveni identifikator HTTP/2 toka = 31 bit



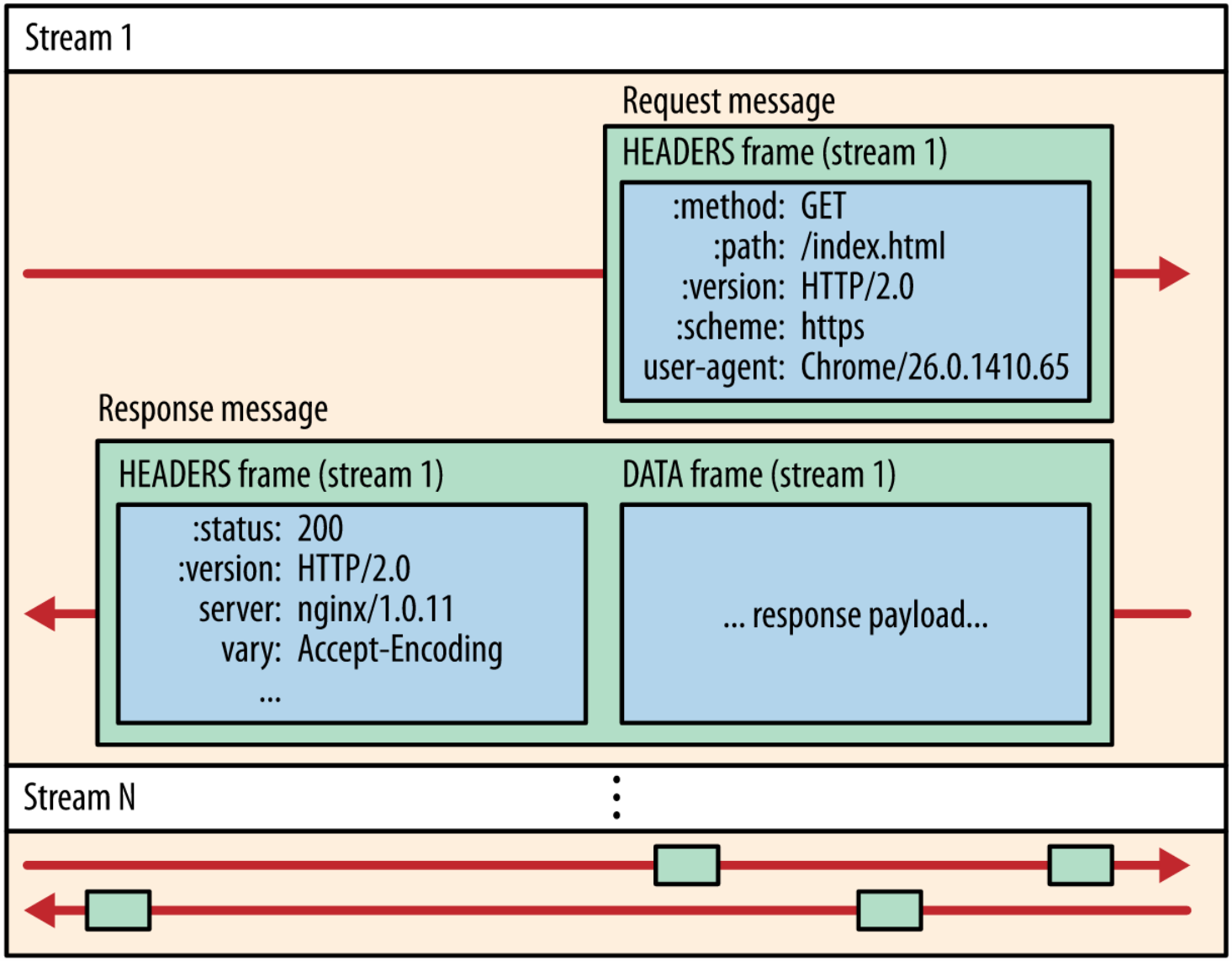
Tokovi, poruke i okviri (1)

- Sva komunikacija se odvija putem jedne TCP veze koja može nositi bilo koji broj dvosmjernih tokova.
- Svaki tok ima jedinstveni identifikator i dodatne informacije o prioritetu koje se koriste za prijenos dvosmjernih poruka.
- Svaka poruka je HTTP poruka, poput zahtjeva ili odgovora, koji se sastoji od jednog ili više okvira.
- Okvir je najmanja komunikacijska jedinica koja nosi određenu vrstu podataka - npr. HTTP zaglavlja, tekst, poruke itd.
- Okviri iz različitih tokova mogu se multipleksirati (*interleaving*), a zatim ponovno rastaviti (demultipleksirati) putem identifikatora toka u zaglavlju svakog okvira.
- Multipleksiranje je metoda u HTTP/2 pomoću koje se može poslati više HTTP zahtjeva i odgovori mogu primati asinkrono putem jedne TCP veze.
- Mehanizam razmjene binarno kodiranih okvira, koji se preslikavaju u poruke koje pripadaju određenom toku, a sve se multipleksira unutar jedne TCP veze zajedno čini temelj koji omogućuje sve ostale značajke i optimizaciju performansi koje pruža HTTP/2 protokol.

Tokovi, poruke i okviri (2)

- HTTP/2 je binarni protokol. Svaki HTTP/2 zahtjev i odgovor dobivaju jedinstveni ID koji se naziva *stream* ID, a HTTP zahtjev i odgovor podijeljeni su u okvire. Okviri su binarni podaci.
  - ID *streama* koristi se za identifikaciju kojem zahtjevu ili odgovoru okvir pripada.
  - Tok podataka je zbirka okvira s istim ID-om *streama*.
- Za postavljanje HTTP zahtjeva, klijent prvo dijeli zahtjev na binarne okvire i okvirima dodjeljuje ID streama zahtjeva. Zatim započinje TCP vezu s poslužiteljem. I tada klijent počinje slati okvire poslužitelju. Nakon što poslužitelj ima spreman odgovor, on dijeli odgovor na okvire i daje okvirima odgovora isti ID toka odgovora. Poslužitelj šalje odgovor u okvirima.
- ID *streama* je neophodan jer se više zahtjeva izvoru šalje putem jedne TCP veze pa ID omogućuje identifikaciju kojem zahtjevu ili odgovoru okvir pripada.

Tokovi, poruke i okviri (3)



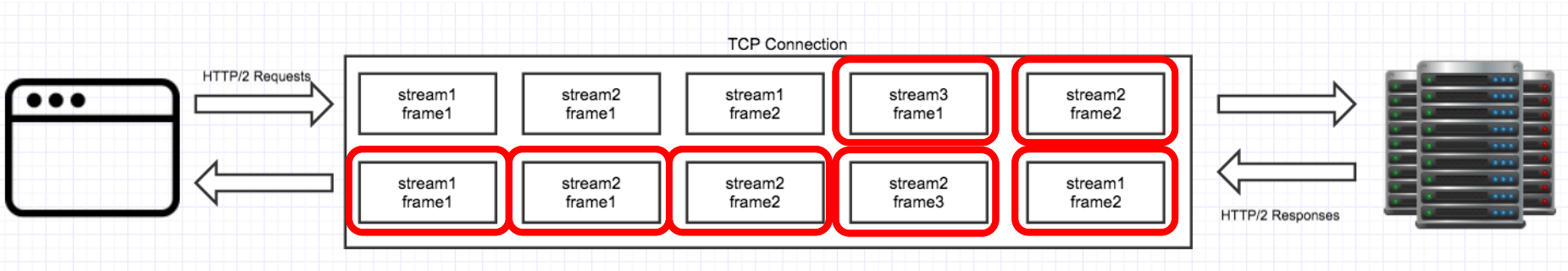
HTTP/2 multipleksiranje (1)

- Ako klijent koji podržava samo HTTP/1.x želi ostvariti višestruke paralelne zahtjeve kako bi povećao performanse, onda mora otvoriti višestruke TCP konekcije.
  - Ova neučinkovitost je posljedica HTTP/1.x *response queuing* svojstva gdje se u jednoj konekciji odgovori serijaliziraju (FIFO) samo jedan odgovor može biti isporučen odjednom, po vezi.
  - *Head-of-line blocking* zbog prekoračenja maksimalnog broja konekcija jednog klijenta i poslužitelja.

- Klijent koji podržava HTTP/2 ne podliježe ovim ograničenjima
  - *Binary framing layer*
  - Uvodi se multipleksiranje poruka zahtjeva i odgovora. Jedna HTTP/2 poruka podijeljena je na međusobno neovisne okvire koji se mogu asinkrono slati i primati. Demultipleksiranje kod klijenta.

HTTP/2 multipleksiranje (2)

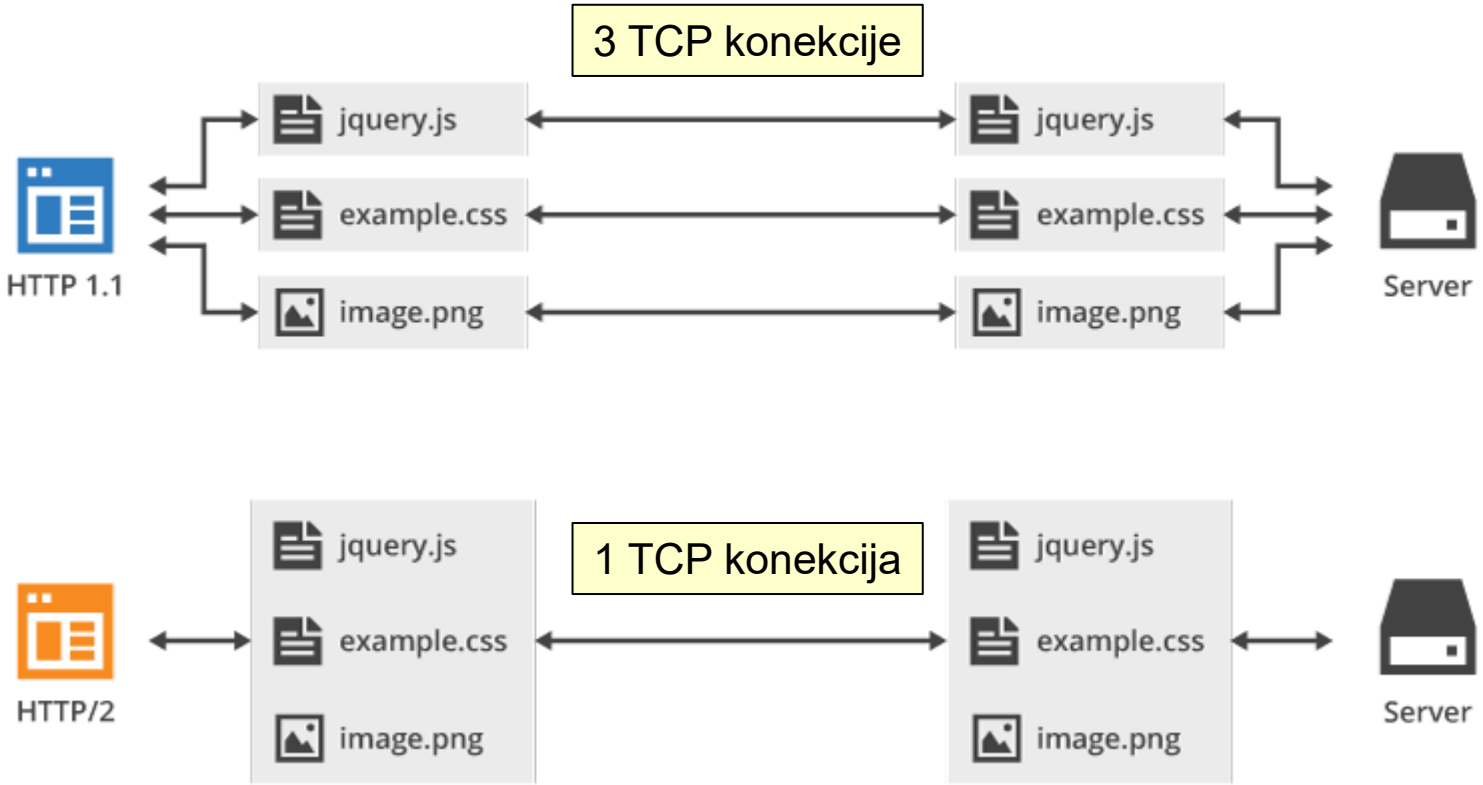
- Jedna TCP veza može se koristiti za slanje HTTP zahtjeva samo jednom poslužitelju.
  - Za povezivanje s više poslužitelja potrebno je više TCP veza.
- Svi HTTP/2 zahtjevi obavljaju se putem uspostavljene TCP veze.
  - Više HTTP/2 zahtjeva podijeljeno je u okvire i dodijeljeni su im odgovarajući ID-ovi toka.
  - Svi okviri iz više tokova šalju se asinkrono. Poslužitelj također šalje odgovore asinkrono. Ako jedan odgovor predugo traje, drugi ne moraju čekati da završi. Zahtjev i odgovor događaju se paralelno, dok klijent šalje okvire poslužitelj također šalje okvire natrag klijentu.
  - Klijent prima okvire sa poslužitelja i raspoređuje ih prema ID *streama*.



Veza prema određenoj domeni ostane otvorena još neko vrijeme nakon prestanka zahtjeva

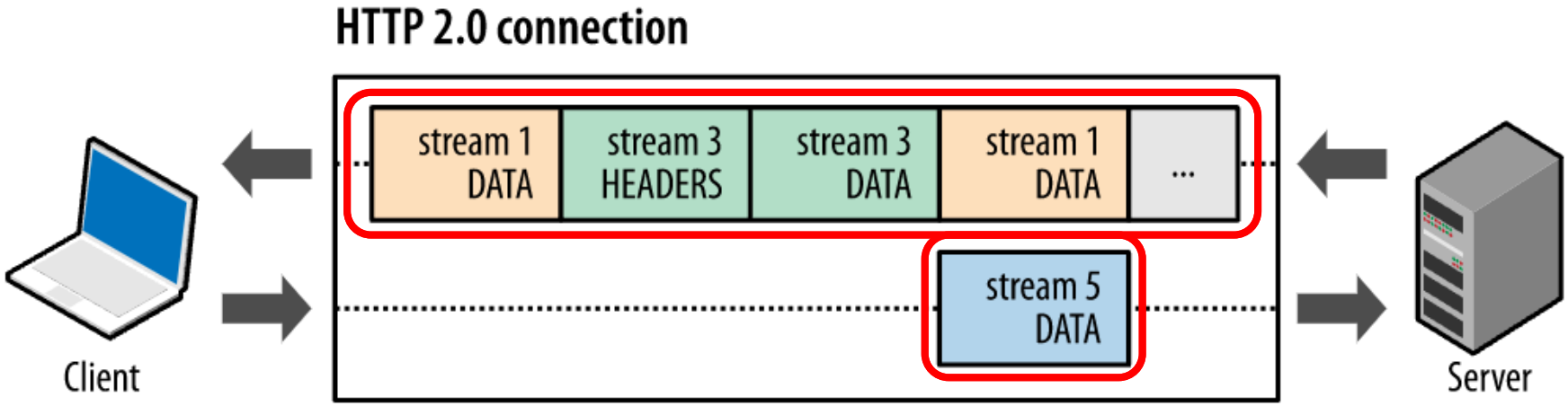
HTTP/2 multipleksiranje (3)

- Korištenje jedne TCP konekcije za više HTTP/2 poruka:

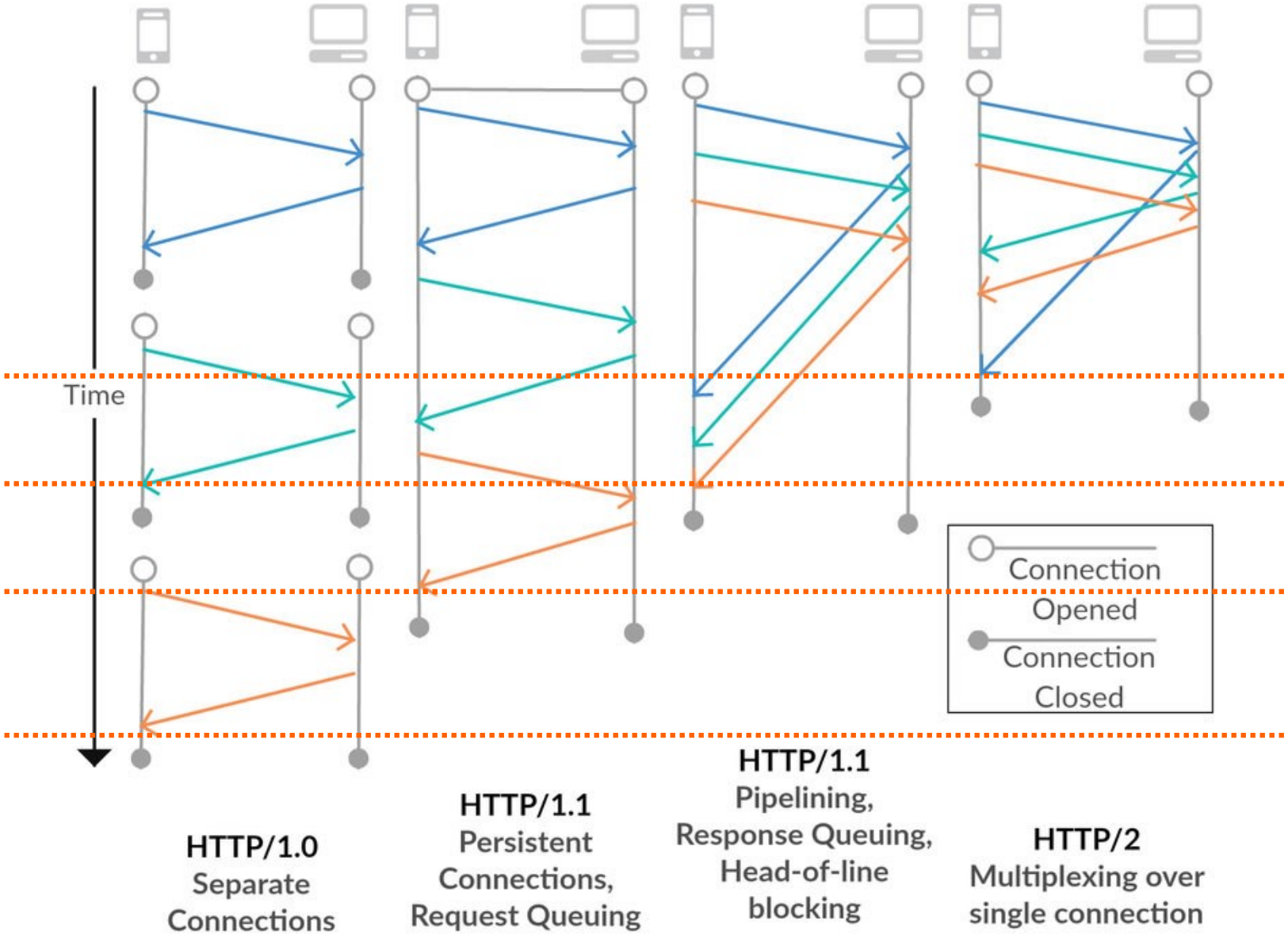


HTTP/2 multipleksiranje (4)

- U ovom primjeru klijent šalje DATA okvir (*stream* 5) na poslužitelja
- Istodobno poslužitelj odgovara sa multipleksiranim nizom okvira za *streamove* 1, 2 i 3
- Nema čekanja na zahtjev, čekanja poruke, blokiranja i obaveznog FIFO redoslijeda poruka



HTTP/2 multipleksiranje (5)

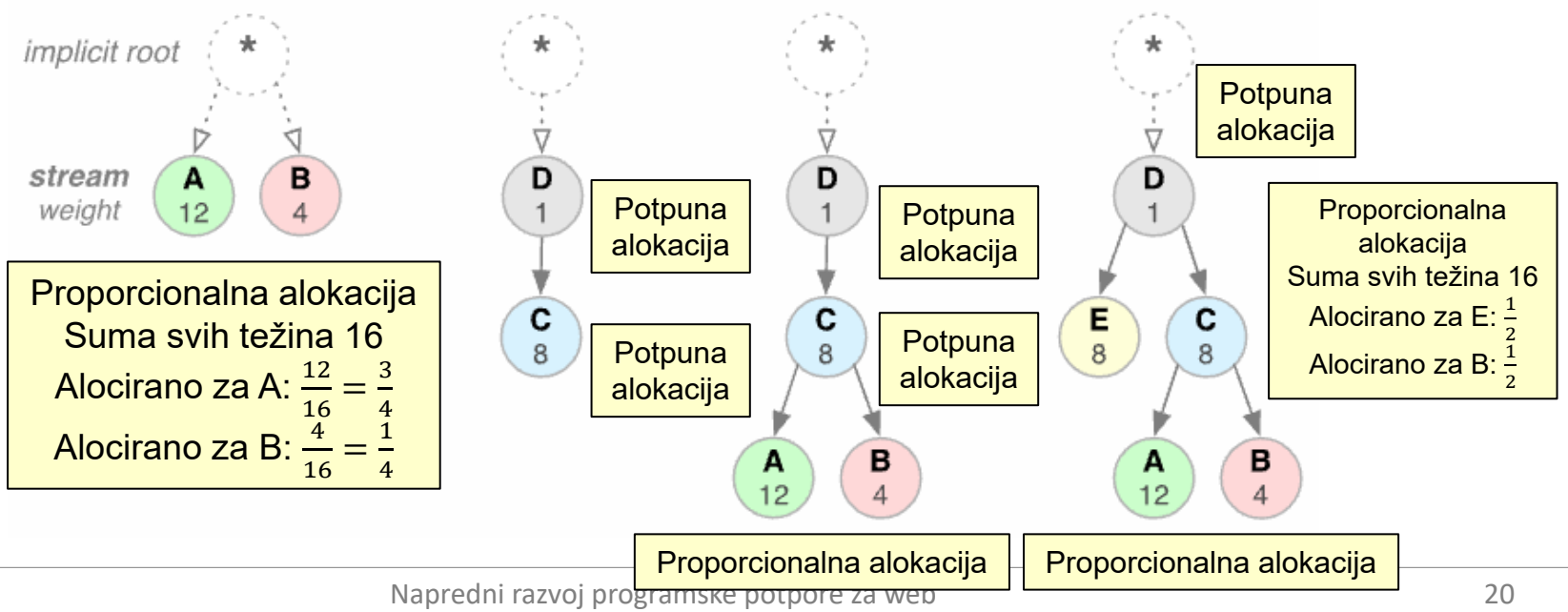


HTTP/2 multipleksiranje (6)

- Važna pozitivna svojstva mehanizma HTTP2 multipleksiranja:
  - Paralelno i isprepleteno slanje više zahtjeva bez blokiranja.
  - Paralelno i isprepleteno slanje više slanje odgovora bez blokiranja.
  - Uporaba jedne veze za paralelnu isporuku više zahtjeva i odgovora.
  - Kraće vrijeme učitavanja stranice uklanjanjem nepotrebnih kašnjenja i poboljšanjem korištenja raspoloživog mrežnog kapaciteta.

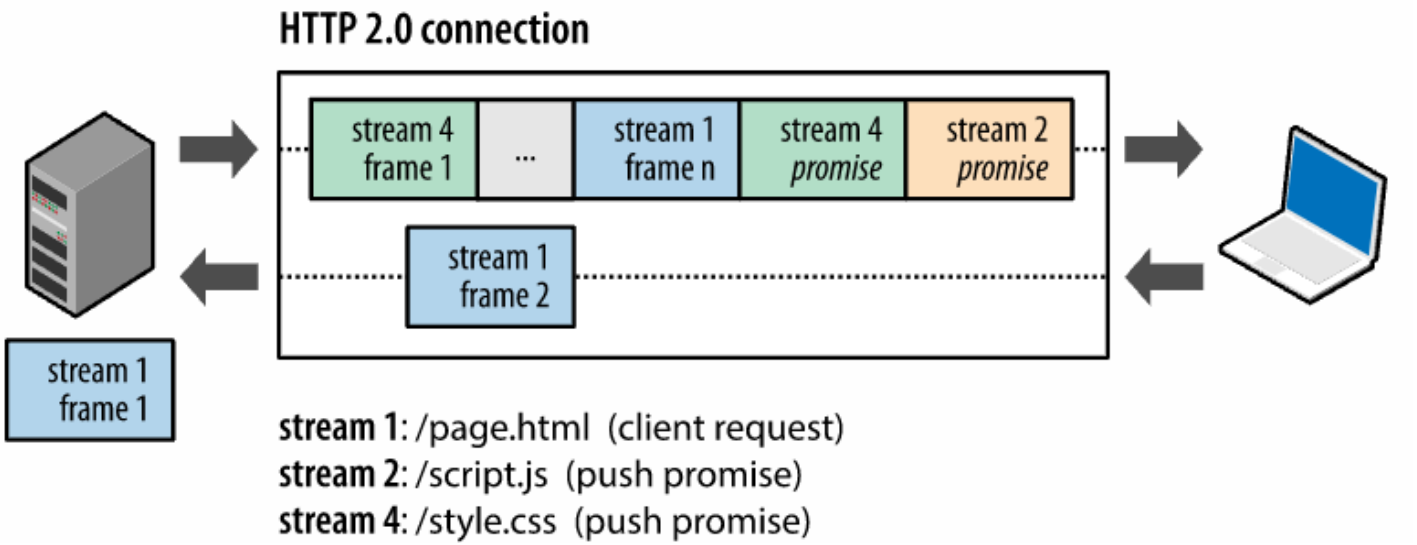
HTTP/2 prioritet toka

- HTTP/2 poruka dijeli se na proizvoljno mnogo pojedinačnih okvira, a okviri iz više tokova se multipleksiraju → redoslijed okvira postaje važan za performanse
- Stoga se definira razina prioriteta i međusobna ovisnost tokova (*stream prioritization*):
  - Svakom toku može se dodijeliti cjelobrojna težina prioriteta između 0 i 255
  - Svakom toku može se dati izričita ovisnost o drugom toku
- Izgrađuje se stablo prioriteta (*prioritization tree*)
  - Proporcionalno udaljenosti od čvora stabla i dodijeljenoj vrijednosti prioriteta poslužitelj alocira računalne resurse, memoriju, procesorsko vrijeme i propusnost mreže. Ovisnosti i težine izražavaju samo transportnu prednost, ne jamče određen redoslijed slanja.



HTTP/2 Server Push

- Poslužitelj može poslati više odgovora na jedan zahtjev klijenta.
  - Osim prvog odgovora, poslužitelj *gura* (push) dodatne odgovore bez novih zahtjeva klijenta.
  - Temeljem prvog zahtjeva poslužitelj anticipira sljedeće zahtjeve klijenta. Posljedica je ubrzavanje prijenosa resursa.
  - Poslužitelj prvo šalje PUSH_PROMISE okvir da bi „signalizirao" svoju namjeru slanja daljnjih tokova. PUSH_PROMISE okvir sadržava samo zaglavlje HTTP poruke. Podaci (DATA okviri) šalju se kasnije. PUSH_PROMISE mora biti zaprimljen da bi se izbjegli klijentski zahtjevi za istim resursima.
  - Ako je resurs već u privremenoj memoriji klijenta (*cache*) on može odbiti *server push* slanjem RST_STREAM okvira. Ovaj mehanizam nije moguć u HTTP/1.x.

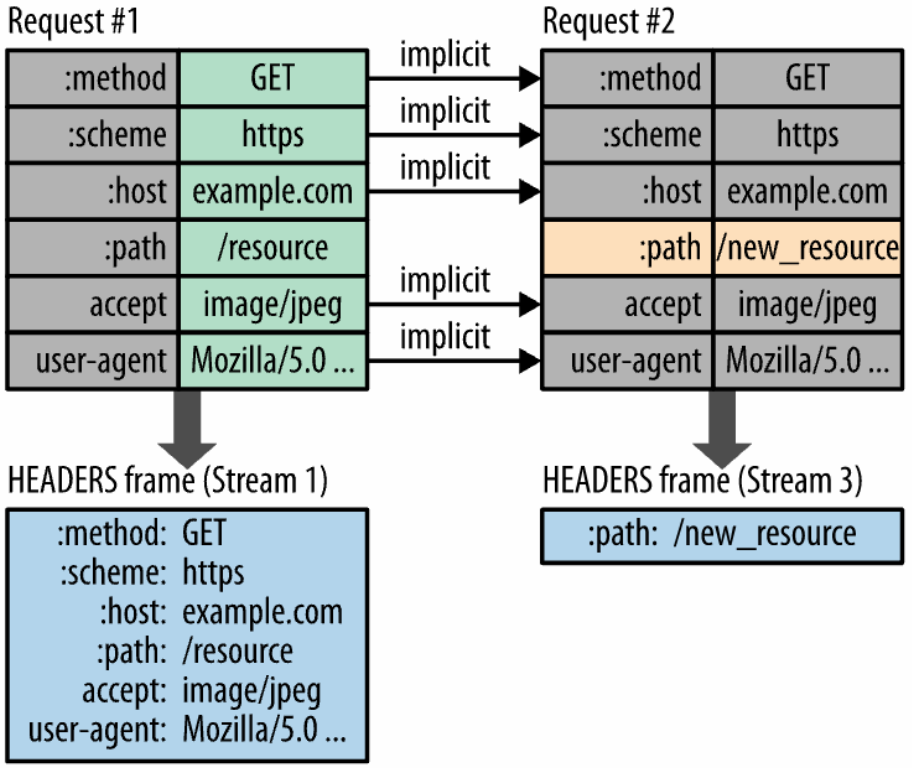


Komprimiranje zaglavlja (1)

- HTTP/1.x prenosi nekodiran *plain text*
  - Svaki HTTP prijenos sadrži skup zaglavlja koja opisuju preneseni resurs i njegova svojstva. U HTTP/1.x, ti se metapodaci uvijek šalju kao običan tekst i dodaju od 500-800 bajtova *overheada* po *requestu*, a ponekad i više ako se koriste HTTP kolačići.
- Kako bi se poboljšale performanse HTTP/2 uvodi komprimiranje zaglavlja zahtjeva (*HPACK compression, Header Compression*) korištenjem dvije metode koje zajedno smanjenju potrebu za količinom podataka koja se mora prenijeti:
  1. Kompresija podataka
     - Huffmanovo kodiranje zaglavlja.
  2. Izbjegavanje višestrukog prijenosa podataka
     - Klijent i poslužitelj uspostavljaju i kontinuirano aktualiziraju indeksiranu tablicu prenesenih polja zaglavlja. Ovime se uspostavlja zajednički kontekst kompresije i onemogućuje se ponovni prijenos podataka koji su već prethodno poslani.

Komprimiranje zaglavlja (2)

- Dodatna optimizacija:
  - Statičke i dinamičke tablice:
    - Statička tablica je unaprijed popunjena i sadržava popis uobičajenih polja HTTP/2 zaglavlja koja će vjerojatno koristiti veze i klijenti
    - Dinamička tablica je u početku prazna i ažurira se na temelju izmjenjenih vrijednosti unutar određene veze.

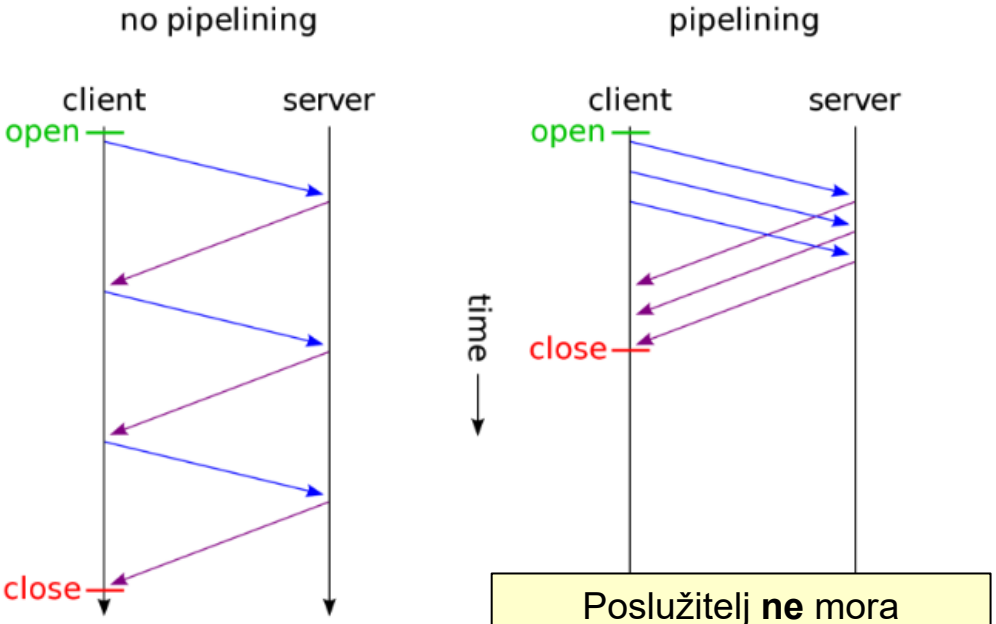


Http2 pipelining (1)

- Http pipelining je mehanizam u kojem se više HTTP zahtjeva šalje jednom TCP konekcijom bez čekanja na njihove odgovarajuće odgovore.
  - Kod HTTP/1.x poslužitelj mora održavati ispravan poredak reda odgovora (*response queuing*). Jednak redoslijedu zahtjeva klijenta.
  - To dovodi do *Head-of-line-blocking* ograničenja.

Head-of-line-blocking

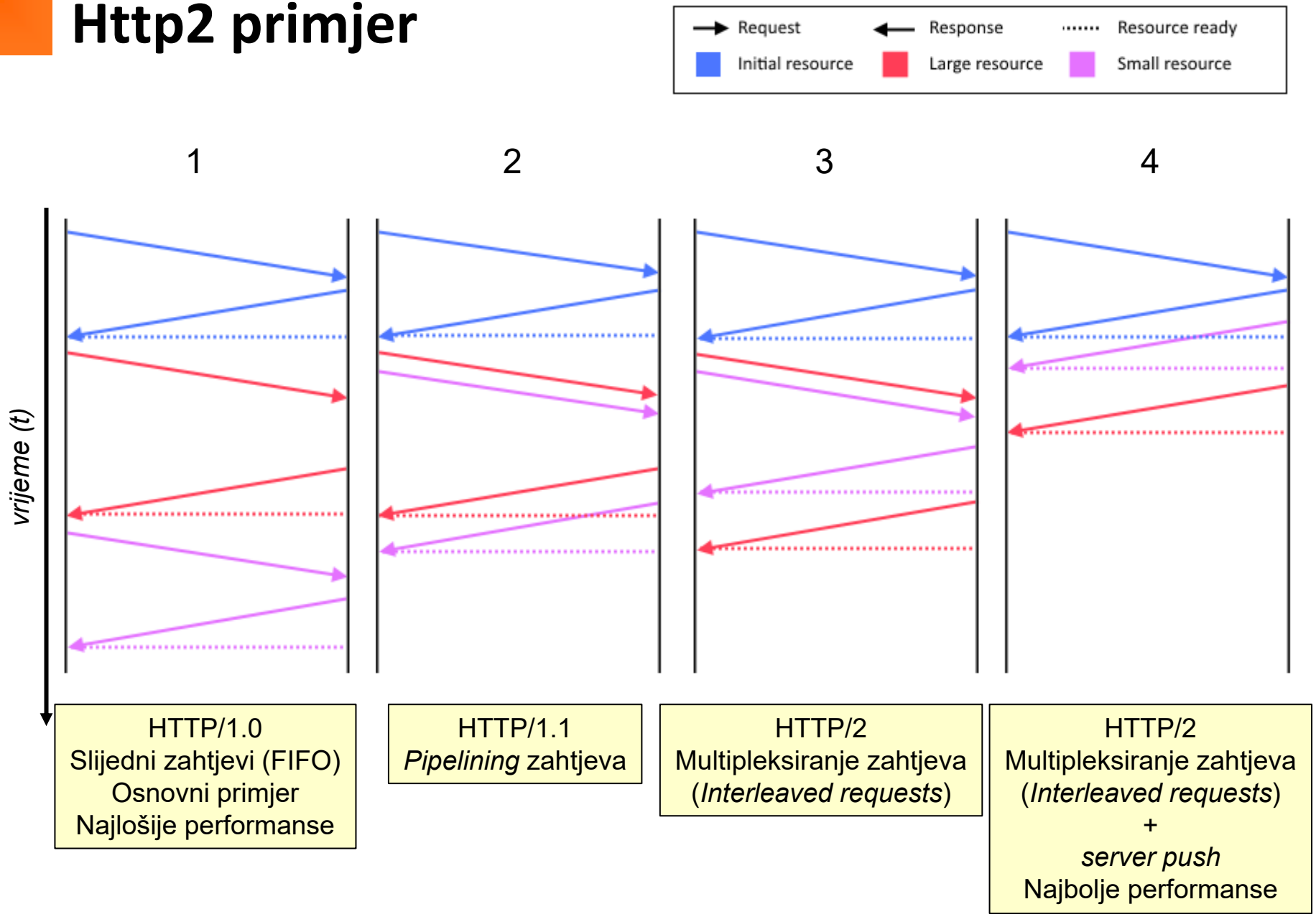
Neka klijent prvo šalje zahtjev A koji je složen i zahtijeva znatne resurse poslužitelja, i nakon toga zahtjev B koji je procesorski jednostavan i ne zahtijeva znatne računalne resurse. Poslužiteljsko računalo, koji je u stanju riješiti nekoliko zahtjeva odjednom, obraditi će zahtjev B vrlo brzo, ali ne može poslati odgovor jer čeka na kraj obrade zahtjeva A kako bi poštivao redoslijed zahtjeva. Na taj način spori zahtjevi postaju ograničenje svih sljedećih zahtjeva.



Poslužitelj u odgovoru mora poštivati redoslijed kojim su zaprimljeni zahtjevi

Poslužitelj **ne** mora poštivati redoslijed zahtjeva. Zahtjevi su označeni ID-om toka. *Head-of-line-blocking* nije moguć na OSI razini HTTP.

Http2 primjer



Zaključak – najvažnije razlike HTTP/2 i HTTP/1.x (1)

- Zašto? Ukratko – povećanje performansi i sigurnosti.
- Kompresija podataka
  - HTTP/2 nudi ugrađenu kompresiju zaglavlja zahtjeva (HPACK). Suvremene web aplikacije obično prihvaćaju niz različitih zaglavlja, poput autorizacije, podataka o klijentu Iako kompresija ovih podataka možda neće imati velike razlike za jedan zahtjev, postoji mnogo podataka poslanih preko mreže koji se spremaju pri komprimiranju u aplikacijama s velikim prometom. HTTP/1.1 prema zadanim postavkama ne komprimira zaglavlja.
- Binarni protokol
  - HTTP/2 je binarni, a HTTP/1.1 tekstualni. Pojednostavljena provedba naredbi, više se ne mogu pogrešno interpretirati zbog korištenja tekstualnog formata. Preglednici koji podržavaju HTTP/2 pretvorit će tekstualne naredbe u binarne prije slanja putem mreže.
- Sigurnost
  - Napadač više ne mogu manipulirati zaglavljima odgovora i ubaciti (*header injection*) znakove u HTTP *response* poslužitelja (*response splitting attack*). Tipično, napadač umeće novi odgovor i mijenja vrijednost *Location headera*.

Zaključak – najvažnije razlike HTTP/2 i HTTP/1.x (1)

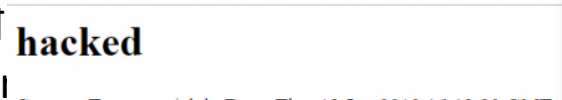
- Zašto? Ukratko – povećanje performansi i sigurnosti.
- Kompresija podataka
  - HTTP/2 nudi ugrađenu kompresiju zaglavlja zahtjeva (HPACK). Suvremene web aplikacije obično prihvaćaju niz različitih zaglavlja, poput autorizacije, podataka o klijentu Iako kompresija ovih podataka možda neće imati velike razlike za jedan zahtjev, postoji mnogo podataka poslanih preko mreže koji se spremaju pri komprimiranju u aplikacijama s velikim prometom. HTTP/1.1 prema zadanim postavkama ne komprimira zaglavlja.
- Binarni protokol
  - HTTP/2 je binarni, a HTTP/1.1 tekstualni. Pojednostavljena provedba naredbi, više se ne mogu pogrešno interpretirati zbog korištenja tekstualnog formata. Preglednici koji podržavaju HTTP/2 pretvorit će tekstualne naredbe u binarne prije slanja putem mreže.
- Sigurnost
  - Napadač više ne mogu manipulirati zaglavljima odgovora i ubaciti (*header injection*) znakove u HTTP *response* poslužitelja (*response splitting attack*). Tipično, napadač umeće novi odgovor i mijenja vrijednost *Location headera*.

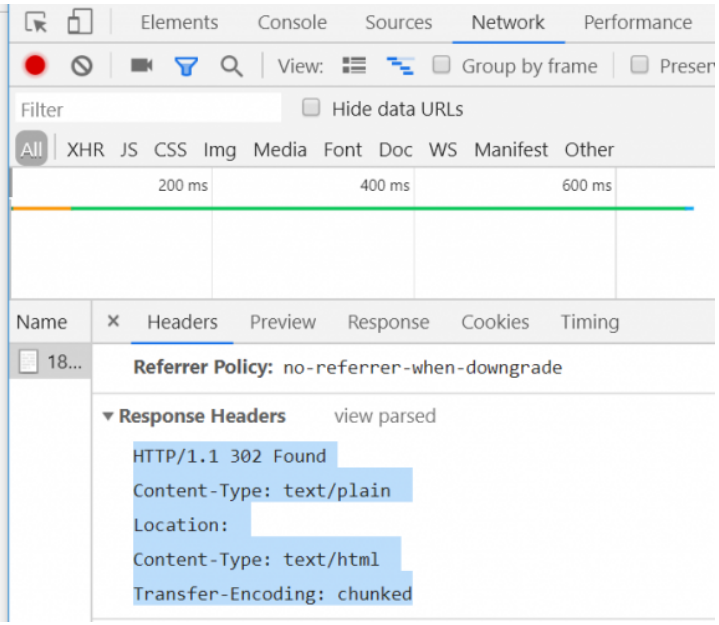
```
HTTP/1.1 302 Found
Content-Type: text/plain
Location: \r\n
Content-Type: text/html \r\n\r\n

<html><h1>hacked!</h1></html>
Content-Type: text/plain
Date: Thu, 13 Jun 2019 16:12:20 GMT
```

Zaključak – najvažnije razlike HTTP/2 i HTTP/1.x (1)

- Zašto? Ukratko – povećanje performansi i sigurnosti.
- Kompresija podataka
  - HTTP/2 nudi ugrađ
    aplikacije obično pı
    klijentu lako kompı
    zahtjev, postoji mn
    komprimiranju u ap
    postavkama ne kor





- Binarni protokol
  - HTTP/2 je binarni, a
    se ne mogu pogreš
    Preglednici koji poc
    slanja putem mreže.
- Sigurnost
  - Napadač više ne mogu manipulirati zaglavljima odgovora i ubaciti (*header injection*) znakove u HTTP *response* poslužitelja (*response splitting attack*). Tipično, napadač umeće novi odgovor i mijenja vrijednost *Location headera*.

```
HTTP/1.1 302 Found
Content-Type: text/plain
Location: \r\n
Content-Type: text/html \r\n\r\n

<html><h1>hacked!</h1></html>
Content-Type: text/plain
Date: Thu, 13 Jun 2019 16:12:20 GMT
```

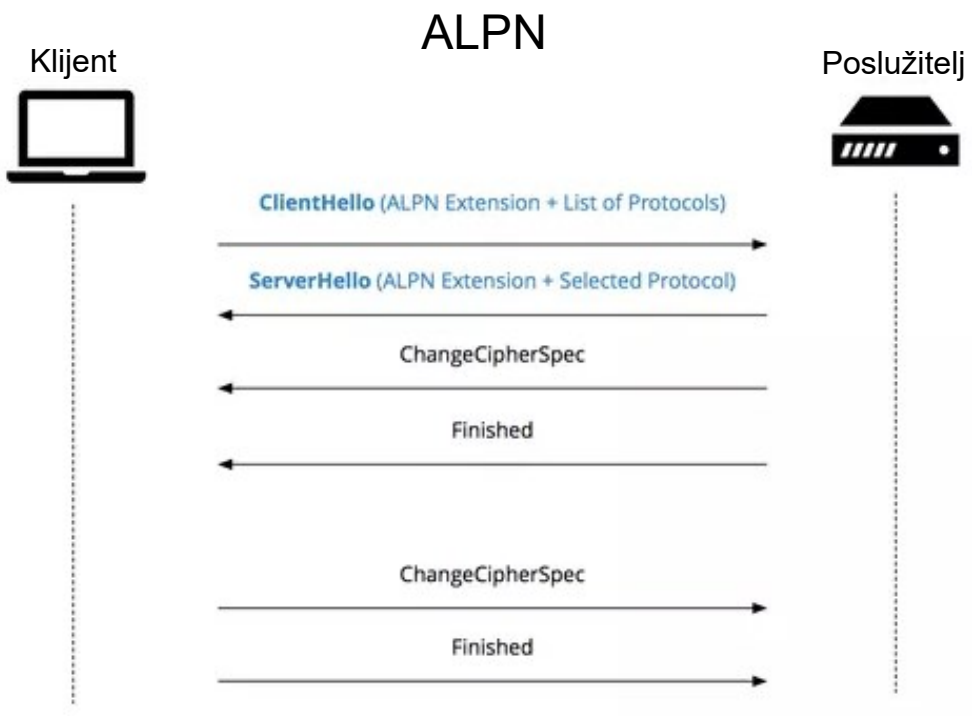
Zaključak – najvažnije razlike HTTP/2 i HTTP/1.x (2)

- Modeli isporuke
  - Multipleksiranje zahtjeva i odgovora te *server push* odgovora.
  - Nije više potrebno koristiti *ad hoc* optimizacije kao kod HTTP/1.x:
    - *Domain sharding* – fragmentiranje domene, datoteke se istodobno dohvaćaju sa više poddomena zaobilazeći ograničenja sekvencijalnog preuzimanja i ograničenog broja istodobnih TCP konekcija klijenta sa istom domenom.
    - *Content concatenation* – povezivanje sadržaja koristi se za smanjenje broja zahtjeva za različite resurse. Kako bi to postigli web programeri često kombiniraju sve CSS i JavaScript datoteke u jednu datoteku.
- Prekoračenje kapaciteta međuspremnika *(buffer overflow)*
  - Međuspremnik je prostor koji koriste klijent i poslužitelj za pohranu zahtjeva koji još nisu obrađeni. U HTTP/1.1, kontrola protoka koja se koristi za upravljanje raspoloživim međuspremnikom implementirana je na transportnom sloju.
  - Kod HTTP/2 klijent i poslužitelj implementiraju vlastite kontrole tijeka komunikacije dostupnog prostora međuspremnika.

Zaključak – najvažnije razlike HTTP/2 i HTTP/1.x (3)

- Brže enkriptirane konekcije
  - HTTP/2 na transportnom sloju koristi *Application-Layer Protocol Negotiation* (ALPN) koje omogućuje brže kriptirane veze jer se aplikacijski protokol određuje tijekom rukovanja (*handshake*).
  - Korištenje HTTP/1.1 bez ALPN-a zahtijeva dodatne poruke za početnu izmjenu postavki i uspostavu kriptirane veze.

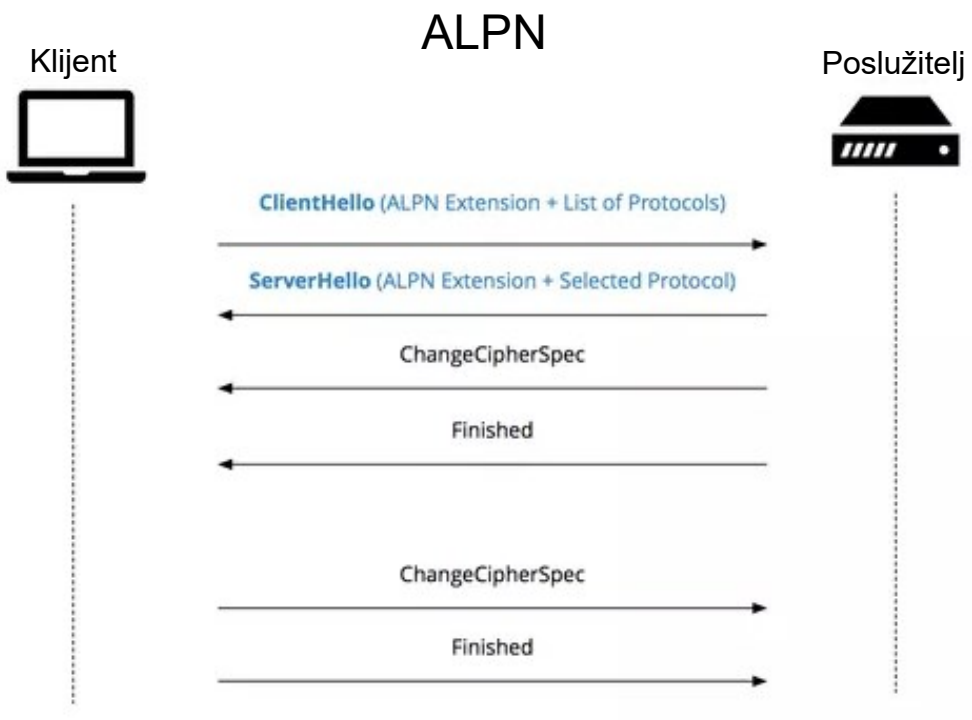
- Kompatibilnost poslužitelja i preglednika
  - HTTP/2 je postao standardna internetska tehnologija i aktualne verzije svih poznatih poslužitelja preglednika podržavaju HTTP/2 protokol.



Zaključak – najvažnije razlike HTTP/2 i HTTP/1.x (3)

- Brže enkriptirane konekcije
  - HTTP/2 na transportnom sloju koristi *Application-Layer Protocol Negotiation* (ALPN) koje omogućuje brže kriptirane veze jer se aplikacijski protokol određuje tijekom rukovanja (*handshake*).
  - Korištenje HTTP/1.1 bez ALPN-a zahtijeva dodatne poruke za početnu izmjenu postavki i uspostavu kriptirane veze.

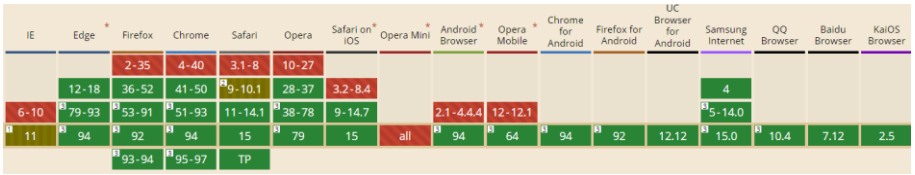
- Kompatibilnost poslužitelja i preglednika
  - HTTP/2 je postao standardna internetska tehnologija i aktualne verzije svih poznatih poslužitelja preglednika podržavaju HTTP/2 protokol.



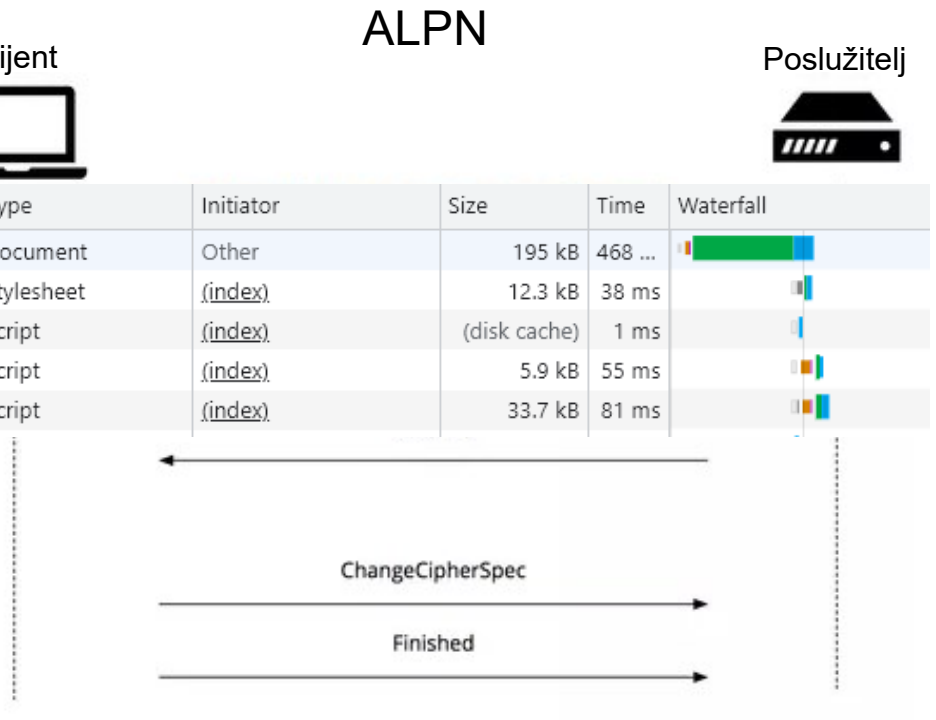
Zaključak – najvažnije razlike HTTP/2 i HTTP/1.x (3)

- Brže enkriptirane konekcije
  - HTTP/2 na transportnom sloju koristi *Application-Layer Protocol Negotiation* (ALPN) koje omogućuje brže kriptirane veze jer se aplikacijski protokol određuje tijekom rukovanja (*handshake*).
  - Korištenje HTTP/1.1 bez ALPN-a zahtijeva dodatne poruke za početnu izmjenu postavki i uspostavu kriptirane veze.

- Kompatibilnost poslužitelja i preglednika
  - HTTP/2 je postao standardna
  podržavaju HTTP/2 protokol.



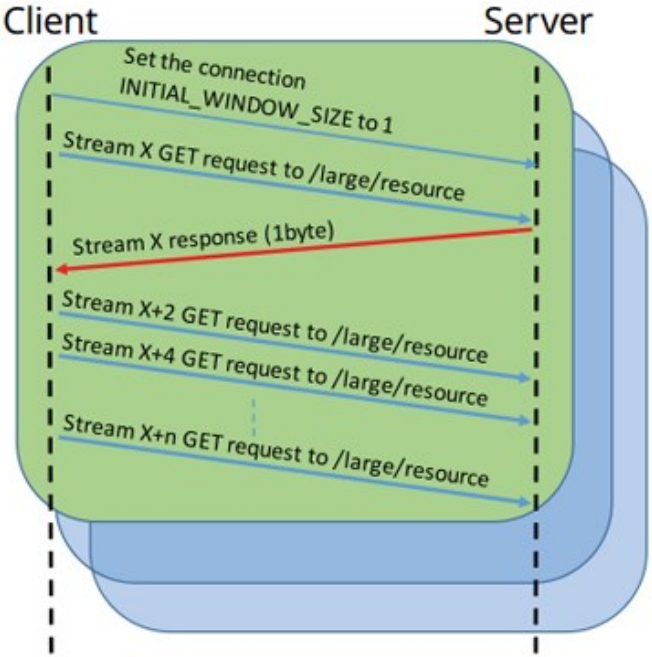
https://caniuse.com/http2

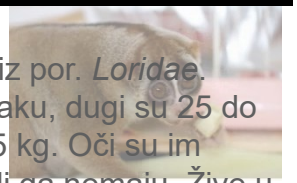


Nedostaci HTTP/2 (1)

1. Sporo čitanje (*Slow read*)
   - Sporo čitanje je napad na poslužitelja sličan Slowloris DDoS napadu, a poziva se na malicioznog klijenta koji čita odgovore pristigle od poslužitelja veoma sporo, čime nepotrebno zauzima vezu.

Sporo čitanje kao napad je proučavano i za vrijeme razvoja protokola HTTP/1.1, međutim niti u novom protokolu nisu poduzete nikakve mjere kako bi se ono spriječilo.





Loriji, polumajmuni iz por. *Loridae*. Imaju sivosmeđu dlaku, dugi su 25 do 40 cm, teški oko 1,5 kg. Oči su im krupne, rep kratak ili ga nemaju. Žive u šumama u Africi južno od Sahare, u jugoist. Aziji, Indiji, Šri Lanki i u Indoneziji. Sporo se kreću. Danju spavaju na granama, a noću traže hranu. Ne mogu hodati po ravnom tlu.

Slowloris is an application layer attack which operates by utilizing partial HTTP requests. The attack functions by opening connections to a targeted Web server and then keeping those connections open as long as it can.

Slowloris is not a category of attack but is instead a specific attack tool designed to allow a single machine to take down a server without using a lot of bandwidth. Unlike bandwidth-consuming reflection-based DDoS attacks such as NTP amplification, this type of attack uses a low amount of bandwidth, and instead aims to use up server resources with requests that seem slower than normal but otherwise mimic regular traffic. It falls in the category of attacks known as “low and slow” attacks. The targeted server will only have so many threads available to handle concurrent connections. Each server thread will attempt to stay alive while waiting for the slow request to complete, which never occurs. When the server’s maximum possible connections has been exceeded, each additional connection will not be answered and denial-of-service will occur.

A Slowloris attack occurs in 4 steps:
1. The attacker first opens multiple connections to the targeted server by sending multiple partial HTTP request headers.
2. The target opens a thread for each incoming request, with the intent of closing the thread once the connection is completed. In order to be efficient, if a connection takes too long, the server will timeout the exceedingly long connection, freeing the thread up for the next request.
3. To prevent the target from timing out the connections, the attacker periodically sends partial request headers to the target in order to keep the request alive. In essence saying, “I’m still here! I’m just slow, please wait for me.”
4. The targeted server is never able to release any of the open partial connections while waiting for the termination of the request. Once all available threads are in use, the server will be unable to respond to additional requests made from regular traffic, resulting in denial-of-service.

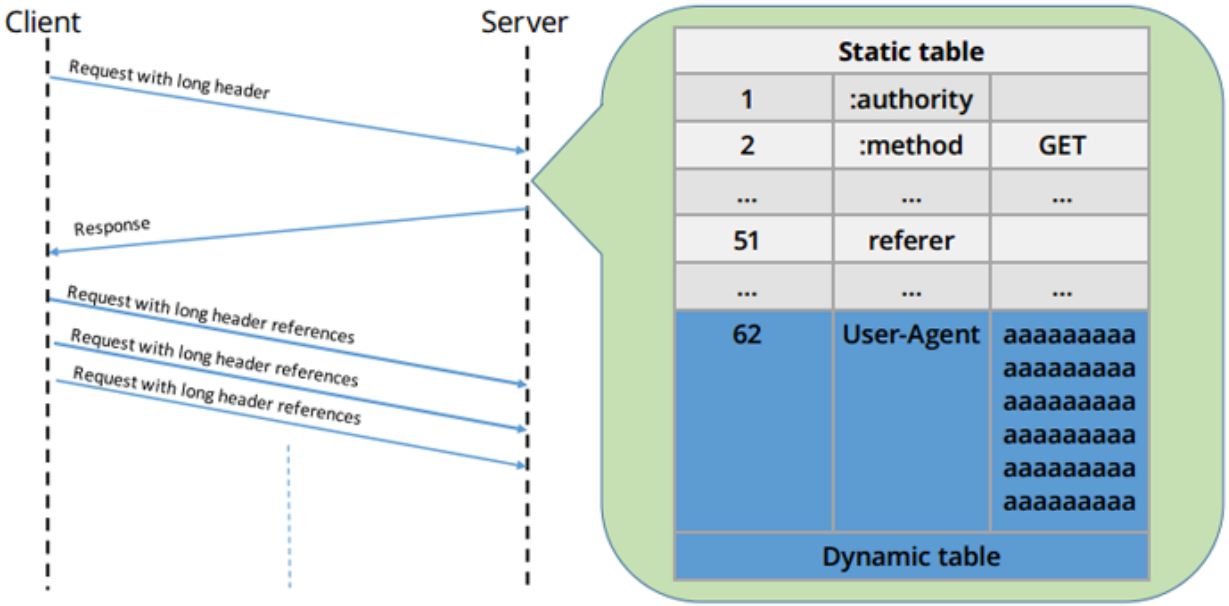
The key behind a Slowloris is its ability to cause a lot of trouble with very little bandwidth consumption.

Za one koje žele znati više Za one koje žele znati više o informacijskoj sigurnosti

https://www.cloudflare.com/fr-fr/learning/ddos/ddos-attack-tools/slowloris/

Nedostaci HTTP/2 (3)

2. HPACK bomba (*HPACK bomb, HPACK attack*)
   - Radi se o napadu na kompresijski sloj, slično napadu sa zip bombom, tj. "dekompresijskom" bombom.
   - Napadač izrađuje male i naizgled obične poruke, koje se mogu pretvoriti u gigabajte podataka na poslužitelju.
   - Na taj način troši se poslužiteljska memorija i tako efektivno usporava ili čak i ruši ciljane poslužitelje.



Nedostaci HTTP/2 (4)

A HTTP/2 implementation built using the priority library could be targetted for a denial of service attack based on HPACK, specifically a so-called “HPACK Bomb” attack.

HPACK is used to reduce the size of packet headers. Basically, the sender can tell the receiver the maximum size of the header compression table used to decode the headers.

This attack occurs when an attacker inserts a header field that is exactly the size of the HPACK dynamic header table into the dynamic header table. The attacker can then send a header block that is simply repeated requests to expand that field in the dynamic table.

Imperva created a header that was 4KB size -- the same size as the entire compression table. Then on the same connection, it opened up new streams with each stream that referred to the initial header as many times as possible (up to 16K of header references).

This can lead to a gigantic compression ratio of 4,096 or better, meaning that 16kB of data can decompress to 64MB of data on the target machine.

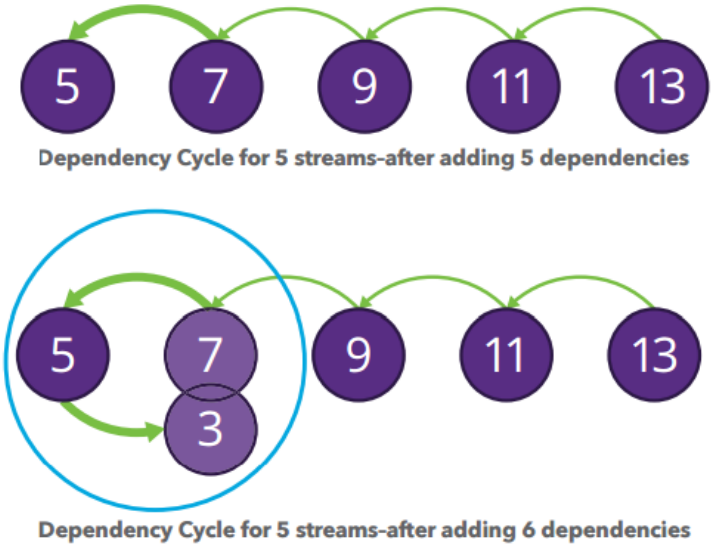
After sending 14 such streams, the connection consumed 896MB of server memory after decompression, which crashed the server, Imperva researchers explain.

It only takes a few such header blocks before the attacker has forced the target to allocate gigabytes of memory, which will take the process down. This requires relatively few resources on the part of the attacker.

Za one koje žele znati više o informacijskoj sigurnosti

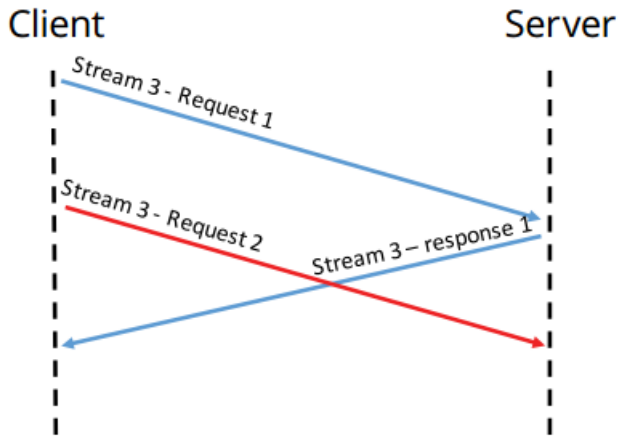
Nedostaci HTTP/2 (5)

3. Napad ciklusa ovisnosti (*Dependency Cycle Attack*)
   - Ovaj napad koristi mehanizme kontrole protoka koje HTTP/2 koristi za optimizaciju mreže.
   - HTTP/2, kao jedno od unaprjeđenja, uveo je i stvaranje prioritizacije i zavisnosti između paketa/podataka koji se šalju preko veze.
   - Međutim, uz informatičku sposobnost napadača, moguće je implementirati krug ovisnosti kao petlju, odnosno prisiljavajući poslužitelj u beskonačnu petlju, te tako izazvati poslužiteljski DoS (*Denial of Service*) napad ili pokrenuti zlonamjeran kôd.



Nedostaci HTTP/2 (6)

4. Zloupotreba multipleksiranja (*Stream Multiplexing Abuse*)
   - Napadač nastoji iskoristiti slabosti novog protokola te implementirati i promijeniti funkcionalnost multipleksiranja preko jedne veze te tako dovesti poslužitelja u stanje nemogućnosti odgovora za pojedine (prave) klijente ili pak potpunog rušenja poslužitelja.



Stream reuse attack diagram