UDC: 004.78 Info M str. 34-38

REINŽENJERING WEB-A: HTTP/2 I QUIC
REENGINEERING THE WEB: HTTP/2 AND QUIC

Marjan Milošević

REZIME: Omasovljenje korišćenja mobilnih platformi, razvoj Interneta stvari (Internet of Things) i povećanje kompleksnosti web-aplikacija iziskuju unapređene performanse mreže. TCP i HTTP već decenijama uspešno vrše svoje funkcije uz odgovarajuća poboljšanja, međutim današnjem Internet-iskustvu neophodne su inovacije. U radu je dat prikaz modernih web-protokola, kojima se unapređuju performanse WWW-a i Interneta uopšte. Dat je osvrt na protokole HTTP/2 i Google-ov eksperimentalni QUIC, na njihove dimenzije poboljšanja, aspekte implementacije i potencijalne nedostatke. Uvođenje novih protokola pokazalo se kao zahtevan zadatak i dobici u performansama i bezbednosti su plaćeni kompleksnošću i dodatnim zahtevima za resursima.

KLJUČNE REČI: WWW, HTTP, QUIC, Internet

ABSTRACT: Heavy usage of mobile platforms, development of the Internet of Things and increasing of web-applications complexity require better network performance. TCP and HTTP have been playing their roles successfully for decades, with certain improvements. However, today's Internet-experience needs innovations. The paper shows a preview of modern web-protocols, which improve performance of the WWW and of the Internet in general. A look into the HTTP/2 technology is given, as well as into the new Google's experimental QUIC protocol, enhancement features, aspects of implementations and potential drawbacks. Involving new protocols showed up to be a highly demanding task and gain in performance and security is greatly payed with complexity and extra-resource requirements.

KEY WORDS: WWW, HTTP, QUIC, Internet

1. UVOD

Bliže se pune četiri decenije od publikovanja TCP protokola [1], a od prve objavljene verzije HTTP-a je prošlo više od 20 godina. U međuvremenu Internet je narastao u globalnu mrežu sa više milijardi krajnjih uređaja. Kategorija "pametni uređaji" nije više ograničena na računare i mobilne telefone, već se pod plaštom paradigme Internet of Things obuhvataju najraznovrsniji sklopovi koji iziskuju povezanost na mrežu: od kamere do automobila. Uređaji su povezani različitim tehnologijama, najčešće bežičnim, čime se omogućava visok stepen autonomije i mobilnosti.

Sa druge strane, Internet je postao globalno tržište, na kojem je konkurentnost direktno zavisna od kvaliteta prisustva na globalnoj mreži, što je dalje tesno povezano sa mogućnostima web-aplikacija i njihovim performansama. Izvorni Internet-protokoli ne mogu lako ići u korak sa zahtevima modernog web-a, već je potreban njihov redizajn ili potpuna zamena. Međutim, prelazak na sasvim nove verzije protokola predstavlja veliki izazov. Jedan od primera koji to kvalitetno ilustruje jeste tranzicija na IPv6[2]. S obzirom na to da su protokoli transportnog sloja realizovani prevashodno na krajnjim sistemima, na nivou operativnog sistema, korenite promene iziskivale bi modifikacije u velikom broju sistemskih programa i aplikacija. Međutim, na aplikativnom sloju je izmene moguće ostvariti značajno jednostavnije. Stoga su promene u ovim protokolima, na primer HTTP-u, izvodljive bez zadiranja u nivo operativnog sistema. Kod usluga transportnog sloja se pribegava drugačijem rešenju, a to je izuzimanje TCP-a kao mandatornog nosioca pouzdanog prenosa i prelazak na UDP. Upravo ova dva pristupa odlikuju osnovne pravce unapređenja web-protokola.

Ovaj rad se bavi evolucijom protokola koji čine potku modernog WWW-a, sa posebnim osvrtom na aktuelne tehnologije i trendove u razvoju budućnosti WWW-a.

U nastavku rada prvo je sažeto izložena geneza WWW-a i HTTP-a, kao ključnog protokola web-a, uz akcenat na elemente vezane za performanse i prevazilaženje uskih grla u njegovom funkcionisanju. Zatim su predstavljene moderne verzije HTTP-a i eksperimentalni Google-ov protokol QUIC, ponovo uz posebnu analizu prednosti i nedostataka u smislu performansi i bezbednosti.

2. RAZVOJ WEB-A

Jedan od najpopularnijih servisa Interneta - WWW nastao je, iz današnje perspektive, iz skromnih motiva, kao način poboljšane organizacije dokumenata i njihovog deljenja u CERN-u[3]. Kako je autor WWW-a, Tim Berners Li rekao jednom prilikom: „Potrebno je samo da uzmem ideju hiperteksta i povežem je sa idejom TCP-a i DNS-a i eto ga[1]" [4]. HTTP (HyperText Transfer Protocol), „kičma" WWW-a, je u svojoj prvoj verziji bio relativno jednostavan klijent-server *pull*-protokol, koji je koristio jednu TCP konekciju za prenos jednog objekta veb-strane, od servera do klijenta, na osnovu eksplicitnog zahteva [5]. Ovakav pristup odgovarao je skromnom sadržaju strana tog vremena - reč je o sredini 90-ih godina prošlog veka. Međutim, sa širenjem globalne mreže, obogaćuje se sadržaj WWW-a i složenost web-sajtova i traga se za poboljšanjima.

Ono što je u samom početku bilo izvesno jeste da HTTP zahteva pouzdan prenos, jer sâme web-strane moraju biti prenete u potpunosti i bez grešaka. Kao posledica razmatran je isključivo TCP kao protokol transportnog sloja. Uprkos tome što sâm WWW po svojoj prirodi nije servis u realnom vremenu i mogu se tolerisati određena kašnjenja, cena zasnivanja TCP konekcije, naročito kod neperzistentnog HTTP 1.0, pokazala se kao značajna. Kod verzije 1.0 svaki objekat (na primer

[1] U originalu: "I just had to take the hypertext idea and connect it to the TCP and DNS ideas and — ta-da!"

slika, JavaScript kôd, tekst...) prenosi se posebnom vezom. Širenjem Interneta, povećavaju se i fizička rastojanja između klijenta i servera, pa RTT (round time trip, vreme potrebno da paket pređe put do servera i nazad), postaje bitan faktor, naročito kada se za svaki deo strane (objekat) kreira posebna veza. U tom smislu HTTP 1.0 nije efikasan i kreirana je unapređena verzija 1.1, sa idejama ubrzavanja prenosa strane [6]. Za samo 3 godine između izdavanja specifikacija verzija 1 i 1.1, dokument je narastao sa 60 na 176 strana, što dovoljno govori o tome da, iako je suština HTTP-a relativno jednostavna, postoji pregršt detalja kojima je obogaćena implementacija. Veliki broj funkcija jeste opcione prirode i nije često implementiran.

Jedno od važnih unapređenja kod protokola HTTP 1.1 je upravo utilizacija TCP veze, gde se jednom TCP vezom može preneti više objekata. Na taj način se izbegavaju dodatna kašnjenja za uspostavljanje veze. Kod TCP-a se veza obavezno uspostavlja tzv. trodelnim rukovanjem, postupkom kod kojeg klijent inicira otvaranje veze postavljanjem SYN bita u TCP segmentu, server odgovara potvrdom i ujedno sa svoje strane inicira vezu i na kraju klijent potvrđuje ovaj zahtev i veza biva uspostavljena. U slučaju da se koristi ACK *piggybacking* (šlepanje), treći korak trodelnog rukovanja se koristi za slanje samog zahteva (na primer GET), tako da je faktički samo jedan RTT utrošen ekskluzivno za formiranje TCP veze. (Više informacija o trodelnoj sinhronizaciji može se naći u literaturi, npr. kod Stevens-a i Tanenbauma [7, 8].) Naredni korak u optimizaciji HTTP-a je realizacija tzv. *pipelining*-a, koji omogućava slanje više uzastopnih zahteva za različitim objektima, bez čekanja dopremanja samih objekata. Na primer, ukoliko je potrebno preuzeti JavaScript kôd i jpg sliku, potražuju se oba elementa zaredom, bez čekanja. Ova opcija je ipak tokom 2017. godine napuštena i u modernim verzijama popularnih web-čitača više nije prisutna. Razlozi su: problematičan rad sa proksi-serverima, nemogućnost prevazilaženja problema blokade čeonog segmenta (HOL - head of line blocking) i pojava novih protokola, koji su počeli da menjaju stare verzije. Problem blokade čeonog segmenta je generalno prisutan u različitim segmentima računarskih mreža kod kojih postoje redovi čekanja (na primer kod realizacije prosleđivanja kod rutera). Kod HTTP-a, ukoliko se pošalje nekoliko zahteva za različite objekte web-strane, ako pri prenosu jednog od objekata neki njegov segment nije prisutan, mora da se čeka dok ne bude upotpunjen i dok se ceo objekat ne dopremi, da bi otpočelo slanje drugog objekta. Dakle, uprkos slanju nekoliko zahteva i uprkos tome što se traženi objekti mogu dobiti različitim redosledom, na nivou samog objekta može doći do zastoja, koji onda može uticati na sve preostale odgovore i celokupnu komunikaciju.

Optimizacija rada protokola HTTP 1.1, pored *pipelining*-a vršena je i na druge načine. S obzirom na to da je protokol ograničavao broj TCP konekcija po izvorištu *(origin)* na 6, a može postojati potreba za više paralelnih veza, kako ne bi dolazilo do zastoja, pribegavalo se tzv. *domain sharding*-u, tj. pojedini elementi strane bi se smeštali na različite izvore (različite *origin*-e), na primer logoi i ikonice se smeštaju na x.domen.net, skripte na y.domen.net, a slike na z.domen.net, tako da se tri puta povećava ukupan broj veza koje je moguće ostvariti od strane web-čitača. Broj zahteva za resursima smanjuje se takođe umetanjem sadržaja u sam HTML, tzv. *resource inlining*. Umesto uobičajenog linkovanja slike ili pdf dokumenta sa osnovne strane, isti se nalazi u samom sadržaju strane, kodiran na odgovarajući način, tako da se u prvom odgovoru servera već može preneti. Na primer, za umetanje jednog piksela u sam HTML, može se koristiti sledeći kôd (preuzeto iz [9]):

```
<img src="
          AAAAAACH5BAAAAAAALAAAAAABAAEAAAICTAEAOw==”
     alt="1x1 transparent (GIF) pixel” />
```

Daljom ekspanzijom Interneta, još više ce povećavaju rastojanja između krajnjih tačaka koje komuniciraju, a u slučaju satelitskog prenosa, može biti reč o više desetina hiljada kilometara, što povlači značajno kašnjenje. Kašnjenje je naročito izraženo kod mobilnih mreža, prevashodno usled njihove prirode funkcionisanja na nižim slojevima, što izaziva pad performansi [9]. U takvoj postavci ušteda u koracima potrebnim za uspostavljanje veze, kao i uopšte dalja optimizacija zahteva i distribucije odgovora, mogu biti ključni za funkcionisanje komunikacije. Google je u tom smislu svojevremeno počeo sa razvojem novog protokola, poznatog pod imenom SPDY, koji je publikovan i u odgovarajućem RFC dokumentu 2012. godine [10]. Ovaj protokol je poslužio kao osnova za HTTP/2, tako da je sam po sebi kasnije postao suvišan, jer je HTTP/2 prihvaćen i podržan kod svih popularnih web-čitača. U nastavku će manja pažnja biti posvećena SPDY-u, a fokus će biti na aktuelnom HTTP/2.

3. HTTP/2

Nova verzija HTTP-a nastala je pod upravom Google-a i fondacije Mozilla, pod čijim okriljem se nalaze najpopularniji web-čitači današnjice. HTTP2 je direktan naslednik SPDY protokola. Jedna od razlika, oko koje nije postignut konačni konsenzus, jeste šifrovanje saobraćaja. Dok je SPDY obavezno koristio TLS (Transport Layer Security), što znači da svaka SPDY veza jeste bila podrazumevano šifrovana i https, kod HTTP2 je šifrovanje formalno ostavljeno kao opciono. U praksi ipak svi značajni web-čitači koriste HTTP/2 isključivo uz TLS, tako da je de fakto HTTP/2 ipak šifrovan.

Među značajnim inovacijama u odnosu na prethodnika, kod HTTP/2 se mogu izdvojiti sledeće:

- Binarni podaci, umesto tekstualnih. Binarni format olakšava razdvajanje jedinica podataka. Sa druge strane, otežan je pregled zahteva i odgovora neposrednim korišćenjem alata poput Wireshark-a. S obzirom na to da je faktički svaki deo komunikacije šifrovan i ne može se neposredno pregledati, svakako se mora pribeći drugim načinima za analizu.
- Multipleksiranje. Svaki podatak (jedinica kod HTTP/2 je HTTP-okvir) pripada određenom toku (strimu) i svi okviri se šalju i primaju bez zadrške, kao deo jednog toka na transportnom sloju.
- Prioriteti za tokove podataka. Tok (strim) može imati prioritet, čime se definiše eventualna prednost jednog toka u odnosu na druge tokove.

- Kompresija zaglavlja. Zaglavlja se komprimuju posebnim algoritmom HPACK [11], čime se dobija na performansama. S obzirom na to da se u samim zaglavljima šalje relativno velika količina podataka koji se ponavljaju, dobici ostvareni kompresijom su značajni.
- *Push* sadržaja. HTTP je u prethodnim varijantama u potpunosti bio *pull* protokol, što znači da je svaki sadržaj sa servera dopreman na osnovu eksplicitnog zahteva klijenta. U verziji 2 server može biti konfigurisan tako da unapred samoinicijativno pošalje podatke, koji nisu direktno traženi, ali postoji velika verovatnoća da su potrebni klijentu.

Sa razvojem protokola SPDY, odnosno HTTP/2, nestala je potreba za različitim specifičnim načinima ubrzavanja performansi, navedenih pri kraju prethodne sekcije (*pipelining, domain sharding,* umetanje resursa).

Analiza HTTP/2 protokola korišćenjem softvera kao što je Wireshark je otežana, jer je HTTP2 binarne prirode, a takođe i u faktički svim slučajevima koristi se u kombinaciji sa TLS, što znači da je neophodno instalirati odgovarajući privatni ključ ili deljeni ključ, sačuvan u logu web-čitača i primeniti odgovarajući dodatak *(plugin)* za dekodiranje HTTP/2.

HTTP/2 podržava tokove *(streams)* podataka, kojima se ostvaruje najveća dobit u performansama. Kako se navodi u samom standardu: "Tok je nezavisni, dvosmerni niz ramova razmenjenih između klijenta i servera u HTTP/2 konekciji". Jedna TCP veza može podržati više tokova, a samim tokovima se prenose HTTP-ramovi. Dva osnova tipa HTTP-ramova su ramovi zaglavlja i ramovi podataka. HTTP/2 tokovi su definisani automatom stanja, sličnom onom kod TCP-a. Tako svaki tok može primati podatke, biti otvoren, polu-zatvoren itd. Strogo je definisano tačno koji ramovi mogu biti poslati u određenom stanju. Jedan od značajnih tipova HTTP-rama jeste SETTINGS-ram, u kojem se definišu različiti parametri HTTP/2 komunikacije. Na primer, broj konkurentnih tokova (u jednoj konekciji) je podrazumevano beskonačan, ali se kroz postavljanje vrednosti MAX_CONCURRENT_STREAMS u okviru SETTINGS-rama može ograničiti. Takođe, "PUSH" opcija (pri kojoj server šalje klijentu određene resurse samoinicijativno, bez eksplicitnog zahteva) može se uključiti ili isključiti kroz SETTINGs ram i njegov parametar ENABLE_PUSH.

HTTP/2 je podržan od strane svih značajnih web-čitača. Sa serverske strane, podrška je ugrađena i u mnoge značajne web-servere, kao što su Apache, nginx i Internet information Server (uz određena ograničenja [12]). Podrška za HTTP2 uglavnom nije podrazumevano prisutna, već je potrebno konfigurisati je posebno. U radu su date određene informacije vezane za Apache web-server.

Pri povezivanju klijenta, potrebno je izvršiti dogovor o korišćenom protokolu. Klijent podrazumevano šalje zahtev HTTP 1.1 i u njemu takođe nudi opciju da se komunikacija nastavi sa HTTP2, kroz direktivu "Upgrade". Ukoliko server podržava HTTP2, komunikacija se nastavlja statusom 101 ("Switching protocol") i u nastavku server šalje HTTP ram tipa "SETTINGS", a zatim i "MAGIC" ram i time okončava prelazak na HTTP2. Formalno, postoje dve opcije rada HTTP2: prva je bez TLS-a, a druga je uz TLS, odnosno faktički predstavlja HTTPS. U praksi, kako je već napomenuto, HTTP2 se uglavnom svodi na komunikaciju preko TLS-a, uz tzv. ALPN (Application-Layer Protocol Negotiation). ALPN predstavlja ekstenziju TLS rukovanja. Umesto da se kroz nekoliko koraka (uz potrošnju nekoliko RTT vremena) izvrši priprema TLS konekcije i posebno izvrši dogovor o parametrima HTTP2, ove dve aktivnosti se vrše istovremeno, čime se dobija na performansama. Ova ekstenzija je opisana u RFC dokumentu 7301 [13].

3.1. *Apache i HTTP/2*

Kod Apache-a HTTP/2 je implementiran kroz poseban modul: mod_http2 [14]. Prvi način angažovanja podrške za HTTP/2 jeste uključivanje samog modula u postojeći, prethodno iskompajlirani web-server. Druga, preporučljiva, jeste dodavanje tokom kompajliranja samog web-servera. Pri instalaciji treba voditi računa o verziji web-servera, ali i verzijama drugih pomoćnih biblioteka, koje HTTP2 zahteva, na primer minimalnoj verziji OpenSSL biblioteke.

Jedna od bitnih stavki vezanih za korišćenje HTTP2 jeste odgovarajući *fallback* mehanizam, kojim se omogućava korišćenje HTTP1.1 ukoliko postoji neki problem u vezi sa HTTP/2. Kod Apache-a je potrebno navesti oba protokola u odgovarajućoj konfiguraciji, međutim, da bi HTTP/2 uopšte bio opcija, neophodno ga je navesti pre HTTP1.1, tako da je on prvi izbor klijenta. HTTP/2 se može definisati na nivou virtuelnih hostova ili na opštem nivou.

Kod popularnih Linux distribucija podrška za HTTP/2 je ugrađena od verzije jezgra 2.4.6, i može se naći kod faktički svih aktuelnih verzija distribucija (CentOS, Ubuntu i dr.), međutim da bi se pokrenuo režim HTTP/2, neophodno je aktivirati modul i podesiti sam server.

HTTP/2 pruža unapređene performanse za klijenta. Međutim, opterećenje servera se povećava i treba imati u vidu veće zahteve za resursima koje postavlja HTTP/2, te računati sa potencijalnom nadogradnjom kapaciteta servera. Povećano opterećenje procesora javlja se usled povećanog broja niti *(threads)*. Kontrola HTTP/2 niti može se vršiti posebnim instrukcijama za tu namenu (H2MinWorkers i H2MaxWorkers). Priroda HTTP/2 podrazumeva vođenje računa o velikom broju zahteva koji su poslati i koji se multipleksiraju, tako da je zauzeće memorije takođe povećano u odnosu na verziju 1.1. Ponovo, postoje posebne instrukcije vezane za ovaj aspekt: H2MaxSessionsStreams (broj paralelnih zahteva po jednoj konekciji) i H2WindowSize (količina podataka u telu zahteva koju server može da baferuje).

HTTP/2 faktički zahteva šifrovanu vezu, odnosno rad preko TLS-a. Apache-ova implementacija protokola HTTP/2 strogo prati formalnu definiciju, datu u odgovarajućem RFC dokumentu. Ponovo, formalno HTTP/2 ne definiše obavezne algoritme šifrovanja, ali klijenti (dakle pre svega web-čitači) uglavnom iziskuju bezbedne algoritme. S tim u vezi, potrebno je konfigurisati i server tako da podržava samo određene algoritme *(cypher suite)*. Kako se u samom standardu RFC 7540 navodi i naglašava: "implementacija MOŽE tretirati dogovaranje narednih

algoritama za šifrovanje sa TLS 1.2 kao grešku konekcije tipa 'INADEQUATE SECURITY'" i dalje se navodi "crna lista" algoritama. Preporuka je da se ovi algoritmi ne koriste. Ukoliko se ne vodi računa o tome, sledi *fallback* na HTTP1.1.

Pregledom mogućnosti protokola HTTP/2 i njegove implementacije, može se zaključiti sledeće:

- dobici u performansama su značajni: kompresijom zaglavlja i eliminisanjem blokade čeonog paketa prevazilaze se značajne mane HTTP1.1
- forsira se TLS i u praksi HTTP/2 radi gotovo isključivo kroz šifrovane veze
- protokol je podržan u svim web-čitačima i u svim popularnim web-serverima
- protokol je znatno složeniji od starog HTTP i iziskuje više sistemskih resursa
- ekspanzija HTTP/2 je u toku: iako svi veliki sajtovi podržavaju ovaj protokol, preostaje značajan deo web-a koji još uvek ne radi sa HTTP/2, a za jedan deo se ne može ni očekivati skori prelazak

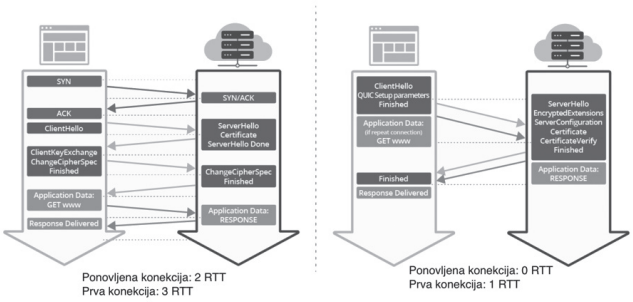
4. QUIC

SPDY je protokol dizajniran od strane Google-a i praktično predstavlja okosnicu aktuelnog HTTP/2. Međutim time se nije stalo sa radom na razvoju novih web-protokola. Tako je nastao i QUIC (Quick UDP Internet Connection), protokol aplikativnog sloja, koji je za sada u eksperimentalnoj fazi i podržan uglavnom na servisima Google-a. U planu je objavljivanje QUIC-a u vidu RFC standarda: za sada postoje nacrti koji se postepeno dorađuju. Za eksperimentalno korišćenje na raspolaganju je testno okruženje razvijeno u okviru projekta Chromium[2]. U eksperimentalnoj fazi QUIC je dostupan i van Google-a kod pojedinih web-servera: Caddy [15], LiteSpeed [16].

Dok HTTP/2 do neke mere održava sličnost sa svojim prethodnicima i koristi TCP za transport, QUIC predstavlja radikalniji pristup, u smislu prenosa većine transportnih funkcija na aplikativni sloj. Pošto kreiranje novih protokola transportnog sloja nije opcija, a TCP unosi previše kašnjenja (prevashodno zbog zasnivanja veze), rešeno je da se na transportnog sloju koristi UDP. Može se reći da se time narušava čitav koncept slojevitosti mrežnih modela i demantuje tradicionalna distinkcija TCP/UDP po kojoj je TCP sporiji, ali pouzdan i koristi se kod svih onih aplikacija koje zahtevaju prenos bez gubitaka, uz kontrolu toka i praćenje sekvence podataka. U slučaju QUIC-a, UDP obavlja praktično samo zadatak multipleksiranja na sloju transporta, dok ostale usluge, neophodne za funkcionisanje WWW servisa, obavlja aplikativni sloj.

Prvi element u smanjenju kašnjenja odnosi se na trodelno rukovanje, koje kod UDP-a, a samim tim i kod QUIC-a ne postoji. Drugi element odnosi se na TLS rukovanje, pri kojem se razmenjuju kriptografske informacije neophodne za šifrovanje veze. TLS je kod QUIC-a obavezan i oba rukovanja, koja kod klasičnog HTTP-a podrazumevaju nekoliko RTT-a, kod QUIC-a se svode na jedan RTT. Poređenje zasnivanja šifrovane konekcije dato je na slici 1 (preuzeto sa [16]).

[2] https://www.chromium.org/quic/playing-with-quic



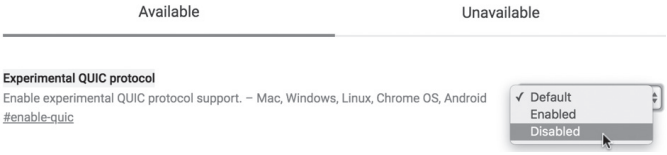
Slika 1 - *Zasnivanje šifrovane konekcije kod HTTP1 i kod HTTP/2*

S obzirom na to da se produkcionom okruženju koristi uglavnom u sklopu Google infrastrukture, informacije o performansama potiču prevashodno od Google-a, a procenti uštede o kojima se govori su 10% kod učitavanja strana i 30% manje baferovanja videa.[17]

QUIC podržava multipleksiranje zahteva/odgovora, slično kao i HTTP/2. Sa druge strane, budući da koristi UDP, server koristi poseban UDP port 443. Svi zadaci transportnog sloja, koje kod HTTP-a obavlja TCP, padaju na teret samog QUIC-a: kontrola toka, signalizacija zagušenja (inače zasnovana na izostanku potvrde koju nosi TCP) i sekvenciranje podataka. Na primer, konvencionalna kontrola zagušenja, kada se realizuje na krajevima (što je najčešći slučaj) zasniva se na potvrdi TCP segmenta. Ukoliko potvrda izostane, to je signal pošiljaocu da je došlo do zagušenja i smanjuje se prozor za slanje. Kod QUIC-a, pošto se koristi UDP, ovaj mehanizam je takođe realizovan u okviru aplikacije, a osnovni principi detekcije zagušenja i ponovnog slanja izgubljenih paketa ispraćeni su i u QUIC implementaciji (na primer "Fast Retransmit"). Ono što je bitna razlika u sekvenciranju podataka je što, za razliku od TCP-a, gde se kod ponovnog slanja (retransmisije) mogu pojaviti i dupli segmenti i duple potvrde - što dalje komplikuje i procenu RTT-a podataka, kod QUIC-a paketi imaju jedinstvene brojeve. To znači da ukoliko neki paket ne stigne na odredište, on ne biva poslat u izvornom obliku, nego se šalje sa novim brojem, tako da ne postoji mogućnost zabune. Detalji su opisani u poslednjem nacrtu standarda [18]. Za povećanje veličine prozora za slanje QUIC koristi CUBIC algoritam. Poenta je u formuli kojom se računa nova, povećana vrednost prozora za slanje, koja je kubni polinom. Odgovarajući dokument je RFC 8312 [19] i predviđa korišćenje ovog algoritma i kod TCP-a.

QUIC poseduje poseban format paketa, koji se sastoji iz zaglavlja i okvira. Postoje 4 tipa paketa i čak 16 tipova okvira [20].

QUIC podržavaju web-čitači Chrome i Opera, dok se za ostale popularne čitače očekuje podrška. Sajtovi koji podrazumevano koriste QUIC, u slučaju da čitač ne podržava ovaj protokol, prelaze *(fallback)* na HTTP. QUIC se može isključiti u samom web-čitaču kroz podešavanja (slika 2).



Slika 2 - *Upravljanje QUIC-om u web-čitaču Chrome*

Bezbednost je direktno inkorporirana u sam protokol u smislu interfejsa između QUIC-a i TLS-a. Gotovo čitav saobraćaj QUIC-a je šifrovan. Izuzetak su fleg i ID veze, koji se nalaze u zaglavlju QUIC paketa. Google je kreirao poseban protokol (QUIC crypto), kompatibilan sa postojećim TLS, zamišljen da traje do izdavanja TLS1.3, koji je planiran za trajnu upotrebu. (TLS1.3 je izdat u avgustu 2018. [21])

QUIC, s obzirom na to da koristi UDP, ima određene zahteve vezane za infrastrukturu, u smislu otvaranja pojedinih portova. QUIC-server koristi port UDP 443, i on mora biti otvoren u zaštitnoj barijeri (firewall) servera za dolazni, odnosno klijenta za odlazni saobraćaj.

5. ZAKLJUČCI

Reinženjering Interneta je kompleksan i dugotrajan posao, skopčan sa stalnim promenama tehnologija, koje nije moguće trenutno ispratiti. Evolucija web-a se nastavlja sa novim protokolima u dva pravca, a to su HTTP/2 i QUIC. HTTP/2 održava osnovne karakteristike svojih prethodnika uz obimna poboljšanja. Međutim, uprkos svojoj zrelosti, i dalje značajan broj sajtova radi sa starim HTTP-om i može se očekivati da će određen procenat do daljeg i ostati na tom protokolu. Moderni web-serveri u potpunosti podržavaju HTTP/2 i za novije, kompleksnije sajtove, sa ozbiljnijim zahtevima za bezbednošću, HTTP/2 je logičan izbor.

Za QUIC se sa druge strane može reći da u potpunosti raskida sa dosadašnjim konceptom WWW protokola. Zasnovan na UDP-u, QUIC gotovo u potpunosti preuzima uloge transportnog i aplikativnog sloja. Performanse i bezbednost su stubovi QUIC arhitekture. Džim Roskind, jedan od kreatora protokola QUIC,u jednom komentaru na Internetu [22] izrazio je svoje ubeđenje da QUIC zaista može da se nazove TCP2 i da će njegov mehanizam za kontrolu zagušenja uspešno ići u korak sa razvojem Interneta.

Sa druge strane, složenost QUIC protokola i njegova ograničenost mahom na Google infrastrukturu, trenutno umanjuju mogućnosti primene. Očekuje se da, nakon eksperimentalne faze, QUIC bude implementiran kod popularnih web-servera i na taj način postane dostupan većem broju aplikacija.

LITERATURA

[1] J. Postel, 'RFC 793 - Transmission Control Protocol', *Rfc 793*, 1981. [Online]. Dostupno na: http://www.ietf.org/rfc/rfc793.txt. [Posećeno: 1.6.2018]

[2] P. Dell, 'Two economic perspectives on the IPv6 transition', *info*, vol. 12, no. 4, pp. 3–14, Jun. 2010.

[3] CERN, 'The birth of the web'. [Online]. Dostupno na: https://home.cern/topics/birth-web. [Posećeno: 1.6.2018.].

[4] T. Berners-Lee, *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web*. HarperBussiness, 2000.

[5] T. Berners-Lee, R. Fielding i H. Frystyk, 'Hypertext Transfer Protocol–HTTP/1.0', *Network Working Group*, 1996. [Online]. Dostupno na: http://www.hjp.at/doc/rfc/rfc1945.html [Posećeno: 1.5.2018].

[6] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach i T. Berners-Lee, 'RFC2616 - Hypertext transfer protocol–HTTP/1.1', *Internet Engineering Task Force*, pp. 1–114, 1999. [Online]. Dostupno na: https://tools.ietf.org/html/rfc2616 [Posećeno 1.5.2018]

[7] W. R. Stevens, 'TCP/IP Illustrated, Volume 1: The Protocols (2nd Edition)', *Addison-Wesley Publishing Company*, 1994.

[8] A. S. Tanenbaum, 'Transportni sloj', u *Računarske mreže*, 4. izdanje, Beograd: Mikroknjiga, 2013, pp. 461–554.

[9] I. Grigorik, *High Performance Browser Networking: What every web developer should know about networking and web performance*. O'Reilly Media, 2013.

[10] M. Belshe, 'SPDY Protocol draft-mbelshe-httpbis-spdy-00', *IETF*, 2012. Dostupno na: https://tools.ietf.org/html/draft-mbelshe-httpbis-spdy-00 [Posećeno 1.6.2018]

[11] H. Ruellan and R. Peon, 'HPACK-Header Compression for HTTP/2.0', Dostupno na https://tools.ietf.org/html/rfc7541 [Posećeno 1.8.2018]

[12] D. So, 'HTTP/2 on IIS', *Microsoft Documentation*, 2016. [Online]. Dostupno na: https://docs.microsoft.com/en-us/iis/get-started/whats-new-in-iis-10/http2-on-iis [Posećeno: 1.6.2018].

[13] S. Friedl, A. Popov, and A. Langley, 'Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension, RFC 7301', 2014. [Online]. Dostupno na: https://tools.ietf.org/html/rfc7301. [Posećeno: 1.5.2018].

[14] Apache Foundation, 'HTTP/2 guide', 2018. [Online]. Dostupno na: https://httpd.apache.org/docs/2.4/howto/http2.html. [Posećeno: 1.6.2018].

[15] Caddy, 'Caddy Features'. [Online]. Dostupno na: https://caddyserver.com/features. [Posećeno: 28.6.2018].

[16] LiteSpeed, 'QUIC Support in LiteSpeed'. [Online]. Dostupno na: https://ma.ttias.be/googles-quic-protocol-moving-web-tcp-udp/#comment-36462. [Posećeno: 28.6.2018].

[17] 'Google Edge Network - QUIC'. [Online]. Dostupno na: https://tools.ietf.org/html/draft-ietf-quic-recovery-14.[Posećeno: 13.6.2018].

[18] J. Iyengar, I.Swett (Eds) 'QUIC Loss Detection and Congestion Control (draft standard ver. 12)'. [Online]. Dostupno na: https://tools.ietf.org/html/draft-ietf-quic-recovery-12. [Posećeno: 1.6.2018].

[19] I. Rhee, L. Xu, S. Ha, L. Eggert, and R. Scheffenegger, 'CUBIC for Fast Long-Distance Networks', 2018. [Online]. Dostupno na: https://tools.ietf.org/html/rfc8312. [Posećeno 1.8.2018]

[20] J. Iyegar and M. Thomson (eds.), 'QUIC: A UDP-Based Multiplexed and Secure Transport - standard draft', 2018. [Online]. Dostupno na: https://datatracker.ietf.org/doc/draft-ietf-quic-transport/. [Posećeno: 1.8.2018].

[21] E. Rescorla, 'The Transport Layer Security (TLS) Protocol Version 1.3', 2018. [Online]. Dostupno na: https://tools.ietf.org/html/rfc8446. [Posećeno: 1.8.2018]

[22] M. Geniar, 'Google's QUIC protocol: moving the web from TCP to UDP', 2016. [Online]. Dostupno na: https://ma.ttias.be/googles-quic-protocol-moving-web-tcp-udp/. [Posećeno: 28.6.2018].

Dr Marjan Milošević, Univerzitet u Kragujevcu, Fakultet tehničkih nauka u Čačku
Kontakt: marjan.milosevic@ftn.kg.ac.rs
Oblast interesovanja: e-obrazovanje, bezbednost informacija, računarske mreže, interakcija čovek-računar.

CIP – Каталогизација у публикацији Народна библиотека Србије, Београд 659.25

INFO M : časopis za informacione tehnologije i multimedijalne sisteme = journal of Information technology and multimedia systems / glavni i odgovorni urednik Miroslav Minović. - [Štampano izd.]. - God. 1, br. 1 (2002)- . - Beograd : Fakultet organizacionih nauka, 2002- (Novi Sad : Sajnos). - 30 cm

Dostupno i na: http://www.infom.org.rs. - Tromesečno. - Tekst ćir. i lat. - Je nastavak: Info Science = ISSN 1450-6254. - Drugo izdanje na drugom medijumu: Info M (CD-ROM izd.) = ISSN 1451-4435 ISSN 1451-4397 = Info M (Štampano izd.)

COBISS.SR-ID 105690636