ELEKTROTEHNIŠKI VESTNIK 91(5): 290-295, 2024
PREGLEDNI ZNANSTVENI ČLANEK

Kaj prinaša transportni protokol QUIC?

Grega Jakus

Univerza v Ljubljani, Fakulteta za elektrotehniko, Tržaška c. 25, 1000 Ljubljana, Slovenija
E-pošta: grega.jakus@fe.uni-lj.si

Povzetek. Za varen prenos spletnih vsebin najpogosteje uporabljamo kombinacijo protokolov HTTP (HyperText Transfer Protocol) za komunikacijo s spletnim strežnikom, TLS (Transport Layer Security) za šifriranje uporabniške vsebine in TCP (Transmission Control Protocol) za zanesljiv prenos sporočil. Posledica uporabe kombinacije protokolov je slabša učinkovitost, kot če bi uporabili en sam poseben protokol, saj vsak protokol vsebuje lastno režijo, kar se odraža predvsem v povečanih zakasnitvah pri prenosu. Z razvojem svetovnega spleta in drugih internetnih aplikacij so se poleg tega pojavile potrebe po učinkovitejših transportnih mehanizmih, ki pa jih je v praksi težko uvesti zaradi okostenelosti transportnega sloja. Nadgradnje transportnih mehanizmov so bile zato udejanjene v okviru aplikacijskih protokolov, še posebej protokola HTTP. Protokol QUIC (Quick UDP Internet Connections) prevzame pod svoje okrilje transportne mehanizme, udejanjene v protokolih HTTP, TLS in TCP, in s tem izboljša učinkovitost prenosa spletnih vsebin. QUIC ni popolnoma samostojen transportni protokol, saj za prenos svojih sporočil uporablja protokol UDP (User Datagram Protocol), s čimer zagotovi, da ga omrežne naprave ne zavračajo. Glavne prednosti protokola QUIC v primerjavi s kombinacijo protokolov HTTP, TLS in TCP so hitrejša vzpostavitev povezave, odprava problema blokade vodilnega sporočila in izvedba v okviru uporabniškega prostora operacijskih sistemov, kar omogoča hitrejše nadgradnje.

Ključne besede: QUIC, TCP, transportni mehanizmi, zakasnitve, okostenelost

What does the QUIC transport protocol bring?

For a secure transfer of the web content, we generally use a combination of the HyperText Transfer Protocol (HTTP) for communication with a web server, Transport Layer Security (TLS) for the encryption of the content and Transmission Control Protocol (TCP) for a reliable transfer. Using a combination of the protocols results in a poorer performance than using a single protocol, as each protocol has its own overhead, which is mainly reflected in an increased latency of the transfer. Moreover, the development of the World Wide Web and other Internet applications has created a need for more efficient transport mechanisms, which are difficult to implement in practice due to the ossification of the transport layer. Improvements to transport mechanisms have therefore been implemented as part of application protocols, in particular HTTP. The Quick UDP Internet Connections (QUIC) protocol combines the transport mechanisms implemented in HTTP, TLS and TCP under one roof and thus improves the efficiency of the web content transfer. QUIC is not a completely independent transport protocol, as it uses the User Datagram Protocol (UDP) to transport its messages and thus ensures that it is not rejected by the network devices. The main advantages of QUIC over the combination of the TCP, TLS and HTTP protocols are the faster connection establishment, the solution to the head-of-line blocking problem and its implementation in the user space of operating systems, which enables faster upgrades.

Keywords: QUIC, TCP, transport mechanisms, delay, ossification

1 UVOD

Svetovni splet je ena najbolj razširjenih internetnih aplikacij. Temelji na konceptu spletnega odjemalca, navadno spletnega brskalnika, ki od strežnika zahteva neko dejanje (npr. prenos spletne strani), strežnik pa se na te zahteve odziva. Komunikacija med odjemalcem in strežnikom poteka s pomočjo aplikacijskega protokola HTTP (HyperText Transfer Protocol). Če želita odjemalec in strežnik prenešeno vsebino šifrirati, za to uporabita poseben aplikacijski protokol, danes je to predvsem protokol TLS (Transport Layer Security). Kombinacijo protokolov HTTP in TLS označujemo kot varni HTTP oziroma HTTPS (HTTP Secure). Šifrirana (ali nešifrirana) sporočila HTTP prevzame protokol TCP (Transmission Control Protocol), ki zagotavlja zanesljiv prenos uporabniških sporočil med partnerskima gostiteljema.

Posledica uporabe kombinacije protokolov je slabša učinkovitost, kot če bi lahko uporabili en sam poseben protokol, saj vsak protokol vsebuje lastno režijo, kar se odraža predvsem v povečanih zakasnitvah. Z razvojem svetovnega spleta in drugih internetnih aplikacij se je pojavila tudi potreba po razvoju učinkovitejših transportnih mehanizmov. A uvajanje novih in celo nadgradnja obstoječih transportnih protokolov sta se izkazala za izredno težavna, predvsem zaradi vmešavanja vmesnih omrežnih naprav (angl. *middlebox*), ki

Prejet 8. oktober, 2024
Odobren 18. november, 2024

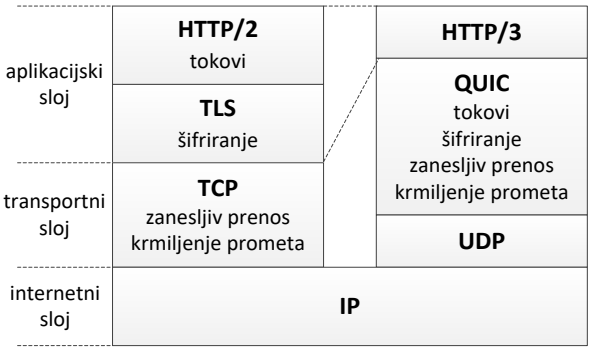
praviloma blokirajo vse njim neznane protokole ali njihove nadgradnje. Protokola TCP in UDP (User Datagram Protocol), ki izhajata iz sedemdesetih let prejšnjega stoletja, sta tako še danes edina protokola transportnega sloja v internetu, ki sta v praksi na voljo aplikacijskim protokolom. Marsikatera nadgradnja transportnih mehanizmov, predvsem protokola TCP, je bila zato udejanjena v okviru aplikacijskih protokolov, še posebej protokola HTTP. Ta se je zato uveljavil kot neke vrste transportni protokol tudi pri številnih aplikacijah zunaj svetovnega spleta.

Za premostitev opisanih problemov je bil zasnovan transportni protokol QUIC (Quick UDP Internet Connections), ki je na kratko predstavljen v naslednjem poglavju. Najpomembnejša prednost protokola QUIC v primerjavi s TCP so zmanjšane zakasnitve pri prenosu. Načini, kako jih protokol QUIC doseže, so predstavljeni v tretjem poglavju, preostale pomembnejše nadgradnje pa v četrtem. V petem poglavju se dotaknemo mehanizmov, ki QUIC-u omogočajo nadaljnji razvoj. Članek se zaključuje s kratkim pregledom razširjenosti protokola v praksi.

2 PROTOKOL QUIC

Protokol QUIC je začelo razvijati podjetje Google in ga javno objavilo leta 2013 [1]. Proces njegove standardizacije je nato prevzela organizacija Internet Engineering Task Force (IETF) in protokol standardizirala maja 2021 [2].

Slika 1 prikazuje umestitev protokola QUIC v internetni protokolni sklad. S slike je razvidno, da QUIC pod svoje okrilje prevzame nekatere mehanizme protokolov TCP, TLS in deloma HTTP. Ker zaradi zavračanja vmesnih omrežnih naprav ne more biti popolnoma samostojen transportni protokol, za pošiljanje svojih sporočil uporablja protokol UDP. Ta tako rekoč ne vsebuje nobenih transportnih mehanizmov, zato jih lahko QUIC udejanji na svoj način.



Slika 1: Protokolna sklada pri prenosu spletnih vsebin in uporabi protokolov TCP (levo) in QUIC (desno) ter pomembnejši transportni mehanizmi posameznih protokolov.

QUIC vsebuje podobne mehanizme zanesljivega prenosa in krmiljenja zamašitev kot TCP, vendar so ti izboljšani in prilagojeni značilnostim modernih omrežij. V nasprotju s protokolom TCP je uporabniška vsebina pri uporabi protokola QUIC šifrirana, zato ta vključuje tudi osnovne mehanizme izmenjave ključev protokola TLS različice 1.3.

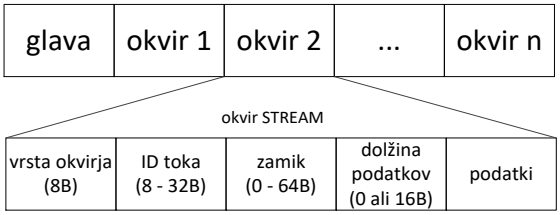
QUIC vsebuje tudi nekatere transportne mehanizme, ki so udejanjeni v okviru protokola HTTP/2, saj jih v praksi ni bilo mogoče vključiti v transportne protokole. Ob uporabi protokola QUIC ti mehanizmi v HTTP niso več potrebni, zato je bila razvita okrnjena različica tega protokola, HTTP/3. Ta je v osnovi podobna starejšima različicama 1.1 in 2, saj uporablja enake metode v zahtevah, statusne kode v odgovorih ter preostala polja v glavah sporočil.

2.1 Format sporočil

Sporočilo QUIC vsebuje glavo, ki ji sledijo okvirji. QUIC pozna šest vrst sporočil. Pet vrst je namenjenih vzpostavitvi povezave in uporablja dolgo glavo. Sporočilo 1-RTT se uporablja po vzpostavitvi povezave (npr. za prenos uporabniških podatkov) in uporablja kratko glavo.

Obstaja 20 vrst okvirjev, ki so lahko vključeni v sporočila. Med njimi so okvirji za preverjanje dosegljivosti partnerja (PING), potrjevanje prejema sporočil (ACK), prenos podatkov v okviru transportnega toka (STREAM) in sprostitev povezave (CONNECTION_CLOSE).

Slika 2 prikazuje osnovni format sporočila QUIC in okvirja STREAM. V eno sporočilo UDP je lahko v okviru iste povezave vključenih več sporočil QUIC.



Slika 2: Format sporočila protokola QUIC in okvirja STREAM

3 ZMANJŠEVANJE ZAKASNITEV

Glavna prednost protokola QUIC v primerjavi s TCP so predvsem manjše zakasnitve pri prenosu podatkov. Te so posledica zlasti

- učinkovitejše vzpostavitve varne povezave v okviru enega samega protokola, in
- omilitve problema blokade vodilnega sporočila s posvojitvijo mehanizma podatkovnih tokov od protokola HTTP/2.

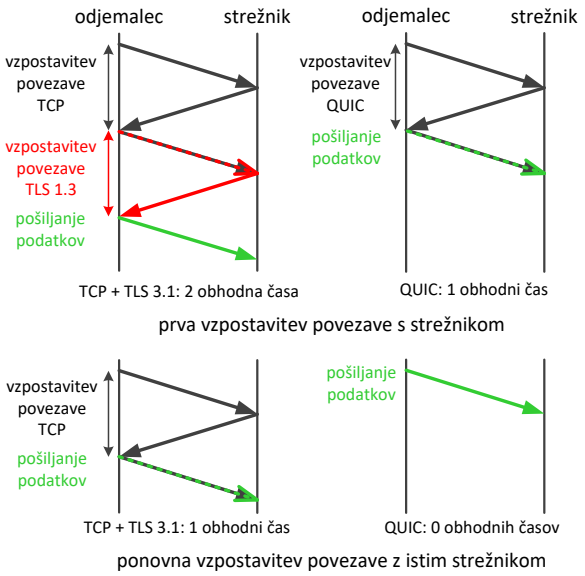
Izboljšavi sta podrobneje opisani v nadaljevanju.

3.1 Vzpostavitev povezave

Zanesljiv in varen prenos vsebin med spletnim odjemalcem in strežnikom je danes večinoma izveden z

uporabo kombinacije protokolov TCP in TLS[1]. Protokoli, ki zagotavljajo zanesljiv in varen prenos sporočil so povezavno naravnani, kar pomeni, da morata protokolna osebka pred prenosom sporočil vzpostaviti povezavo. Izraz *povezava* se nanaša na skupno stanje komunicirajočih partnerjev oziroma na informacijo, ki jo morata oba poznati, da lahko omogočita zanesljiv in varen prenos. V primeru protokola TCP ta informacija med drugim vključuje sekvenčne številke sporočil, ki sta si jih ali si jih še bosta partnerja izmenjala, in količino vsebine, ki jo je pripravljen sprejeti sprejemnik (velikost sprejemnega okna). V okviru protokola TLS pa se morata partnerja dogovoriti o šifrirnih postopkih in si izmenjati šifrirni ključ.

Ker sta TCP in TLS samostojna protokola, morata vzpostaviti vsak svojo povezavo, kar uporabniku prinaša neprijetno zakasnitev. S slike 3 je razvidno, da sta potrebna vsaj dva obhodna časa[2], da se med spletnim odjemalcem in strežnikom vzpostavi prva povezava za varen in zanesljiv prenos sporočil. Če se namesto TLS 1.3 uporablja starejša različica 1.2, pa so potrebni celo trije obhodni časi. Pri vsaki ponovni vzpostavitvi povezave med istima partnerjema se omenjeni parametri lahko sicer pošljejo skupaj s podatki. Ker v tem primeru izmenjava posebnih sporočil za vzpostavitev povezave TLS ni potrebna, je en obhodni čas prihranjen.



Slika 3: Vzpostavitev povezave s strežnikom. Levo: kombinacija protokolov TCP in TLS, desno: protokol QUIC. Zgoraj: prva vzpostavitev povezave s strežnikom, spodaj: ponovna vzpostavitev povezave z istim strežnikom.

Protokol QUIC po drugi strani poskrbi za varen in zanesljiv prenos sam. Partnerja si zato lahko vse potrebne informacije izmenjata v okviru enotne vzpostavitve povezave. Za vzpostavitev prve povezave je tako pri protokolu QUIC potreben le en obhodni čas, za ponovno povezavo med istima partnerjema pa poseben postopek sploh ni potreben, saj se uporabijo kar parametri iz prejšnje povezave (angl. *0-RTT connection establishment*) [2].

3.2 Multipleksiranje podatkovnih tokov

Multipleksiranje je razvrščanje različnih podatkovnih tokov v enoten tok. Izvajamo ga, ko je na voljo en sam fizični ali navidezni vir, ki bi lahko te tokove sprejel v obravnavo: en sam procesor, fizični medij ali navidezna povezava. Pri prenosu spletnih vsebin ustvari podatkovne tokove spletna aplikacija (oz. brskalnik) in jih želi poslati prek omejenega števila transportnih povezav (večinoma ene same). Ko želi na primer uporabnik obiskati neko spletno stran, mora namreč brskalnik za njen prikaz navadno prenesti več različnih virov – poleg ogrodja spletne strani še več različnih slik in skript. Zahteva za posamezen vir in njegov prenos predstavlja samostojen podatkovni tok.

Ker TCP ne podpira razvrščanja aplikacijskih podatkovnih tokov v eno povezavo, je na spletu do zdaj za to, čeprav gre za transportni mehanizem, skrbel aplikacijski protokol HTTP. V nadaljevanju sta predstavljeni evolucija mehanizma multipleksiranja podatkovnih tokov in njegova izvedba v protokolu QUIC.

3.2.1 Uporaba več TCP-povezav v HTTP/1.1

Pri uporabi protokola HTTP/1.1 je moral brskalnik za sočasne prenose spletnih virov vzpostaviti več sočasnih TCP-povezav z istim strežnikom, saj je po eni povezavi mogoče hkrati zahtevati in dobiti le en vir. To je vneslo dodatne zakasnitve, večje število povezav pa je tudi težje upravljati in predstavljajo varnostno tveganje, še posebej za strežnik. Večina modernih brskalnikov zato omejuje število sočasnih povezav s strežnikom (v okviru istega domenskega imena) na šest.

Protokol HTTP/1.1 je sicer uvedel mehanizem *cevljenja* (angl. *pipelining*) [3], pri katerem lahko odjemalec strežniku po eni TCP-povezavi pošlje več zaporednih zahtev, ne da bi čakal odgovore nanje (slika 4). A ker je ta model multipleksiranja v praksi težko udejanjiti, ga moderni brskalniki skorajda ne uporabljajo. Cevljenje je poleg tega podvrženo še t. i. *blokadi vodilnega sporočila* (angl. *head-of-line blocking*). Ker morajo odgovori na posamezne zahteve slediti vrstnemu redu zahtev, povzroči zamuda pri obdelavi ene izmed zahtev zastoj tudi pri preostalih.

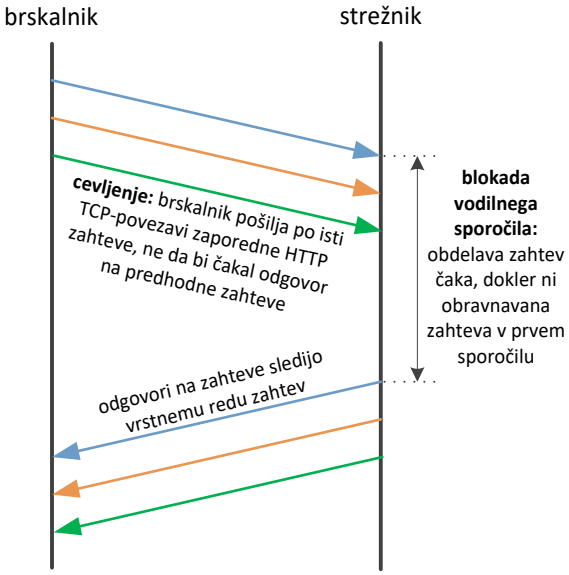
3.2.1 Razvrščanje podatkovnih tokov v eno TCP-povezavo v HTTP/2

Namesto neučinkovitih sočasnih TCP-povezav in cevljenja lahko pri HTTP/2 brskalnik uporabi le eno

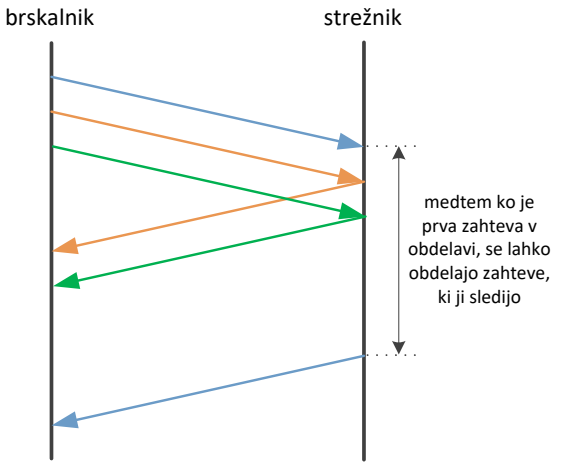
[1] HTTP lahko uporablja tudi transportni protokol UDP, ki ne zagotavlja zanesljivega prenosa, kar pa je v praksi zelo redko. Poleg tega ni nujno, da sporočila HTTP šifriramo, je pa to z varnostnega vidika vsekakor zaželeno.

[2] Obhodni čas je čas, ki mine od začetka pošiljanja sporočila do zaključenega prejema odgovora nanj (oziroma potrditve njegovega prejema).

povezavo, v katero razvrsti več podatkovnih tokov [4]. Najpomembnejša izboljšava v primerjavi s cevljenjem je, da odgovorom na zahteve ni treba slediti vrstnemu redu zahtev (slika 5).



Slika 4 – Cevljenje v HTTP/1.1



Slika 5: Multipleksiranje podatkovnih tokov v HTTP/2.

A multipleksiranje v okviru protokola HTTP/2 še ne odpravlja v celoti problema blokade vodilnega sporočila, saj se ta prenese en sloj nižje – v protokol TCP. Ta namreč zagotavlja ohranjanje vrstnega reda prenešenih sporočil v okviru povezave. Če se eno izmed sporočil izgubi ali pokvari, ga oddajnik TCP ponovno pošlje, vsa preostala sporočila, ki so medtem uspešno prispela do sprejemnika, pa morajo nanj počakati, saj mora sprejemnik vsa sporočila svojemu uporabniku (npr. protokolu TLS, ta pa naprej HTTP) izročiti v pravilnem vrstnem redu.

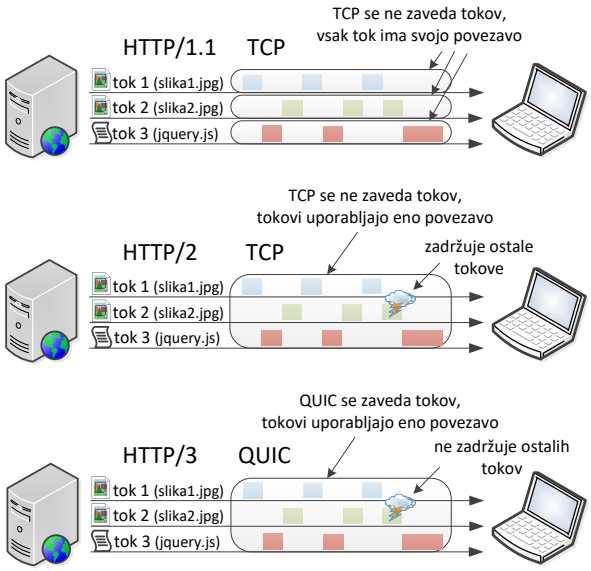
Če torej v okviru ene TCP-povezave prenašamo sporočila več neodvisnih podatkovnih tokov (npr. več slik) in se eno izmed sporočil izgubi ali pokvari, bodo morala vsa nadaljnja (uspešno prenešena) sporočila počakati z izročitvijo aplikacijskemu protokolu, čeprav niso vsebinsko povezana s prizadetim sporočilom (prenašajo na primer delčke druge slike, ki bi lahko bila medtem prikazana na spletni strani).

3.2.2 Uvedba tokov v protokolu QUIC

QUIC uvaja mehanizem tokov na transportnem sloju po zgledu protokola HTTP/2. Podobno kot HTTP/2 lahko tudi QUIC več tokov, ki pripadajo isti aplikaciji, umesti v eno transportno povezavo [2].

Problem blokade vodilnega sporočila, ki je eden glavnih problemov protokola TCP, odpravlja QUIC tako, da ohranitve vrstnega reda ne zagotavlja več v okviru povezave kot protokol TCP, temveč v okviru posameznega transportnega toka v povezavi. Manjkajoče ali pokvarjeno sporočilo v enem izmed tokov zato zadržuje le tista sporočila, ki mu sledijo v istem podatkovnem toku, ne pa sporočil v preostalih tokovih v povezavi.

Slika 6 ponazarja primerjavo med multipleksiranjem podatkovnih tokov pri prenosu spletne strani pri protokolih HTTP/1.1, HTTP/2 in QUIC.



Slika 6: Primerjava multipleksiranja podatkovnih tokov pri prenosu spletne strani pri protokolih HTTP/1.1, HTTP/2 in QUIC.

4 DRUGE POMEMBNEJŠE IZBOLJŠAVE

4.1 Varnost

Omenili smo že, da so sporočila protokola QUIC vedno overjena, uporabniška vsebina pa šifrirana. Poleg nje je v nasprotju s TCP (šifriranim s TLS) šifriran tudi večji del glave sporočil [2]. To še dodatno povečuje varnost prenosa, saj je uspeh številnih vrst napadov odvisen od sposobnosti napadalcev prebrati vrednosti v glavah izvornih sporočil.

4.2 Selitev povezave

Zanimiva lastnost protokola QUIC je tudi možnost t. i. *selitve povezave* (angl. *connection migration*) [2]. Pri protokolu TCP določa povezavo par t.i. *vtičnic* (angl. *socket*), ki ju sestavljata IP-naslova in naslova vrat (angl. *port*) partnerskih aplikacijskih protokolnih osebkov[3]. Če se med vzpostavljeno povezavo spremeni IP-naslov ali vrata enega izmed gostiteljev[4], morata gostitelja vzpostaviti novo povezavo.

QUIC po drugi strani za identifikacijo povezave uporablja oznako, ki je neodvisna od IP-naslova in naslova vrat. Ta gostiteljema omogoča sklicevanje na obstoječo povezavo, čeprav se spremeni naslov vtičnice enega izmed gostiteljev.

4.3 Nadzor nad zamašitvami

QUIC prinaša tudi kopico izboljšav mehanizmov nadzora nad zamašitvami v omrežju [5]. V tem razdelku so omenjene nekatere izmed njih.

V nasprotju s protokolom TCP QUIC razlikuje med vrstnima redoma uporabniških sporočil (angl. *delivery order*) in protokolnih sporočil (angl. *transmission order*), ki prva prenašajo. Vrstni red uporabniških sporočil izraža z zamikom oziroma položajem sporočila v toku uporabniške informacije (polje *stream offset*), vrstni red protokolnih sporočil pa s številko sporočila (polje *packet number*). Do razlike med obema vrstnima redoma pride, ko je treba neko sporočilo ponovno poslati. Takrat se številka sporočila (*packet number*) poveča, saj se za ponovno pošiljanje uporabniškega sporočila uporabi novo protokolno sporočilo, položaj v toku uporabniških sporočil (*stream offset*) pa se ne spremeni, ker gre za ponovno pošiljanje iste vsebine.

Z razlikovanjem vrstnih redov uporabniških in protokolnih sporočil QUIC odpravlja dvoumnost pri ponovnem pošiljanju (angl. *retransmisson ambiguity*) [6], ki se pojavlja pri protokolu TCP. Ko oddajnik TCP od sprejemnika prejme potrditev, lahko namreč iz nje sicer ugotovi, na katero uporabniško sporočilo se potrditev nanaša, ne pa tudi, kdaj (v katerem poskusu) je bilo sporočilo poslano. Odprava dvoumnosti bistveno poenostavlja mehanizem ponovnega pošiljanja sporočil, preprečuje nepotrebna ponovna pošiljanja in omogoča bolj natančno določanje obhodnega časa [5].

Podobno kot TCP tudi QUIC uporablja koncept zamašitvenega okna, ki krmili oddajno okno, vendar ne predpisuje konkretnih mehanizmov za njegovo upravljanje. Namesto tega QUIC ponuja splošne mehanizme za indikacijo zamašitev, izvedbo mehanizmov za nadzor zamašitev pa prepušča oddajniku.

5 PREPREČEVANJE OKOSTENELOSTI

Komunikacijski protokol mora biti mogoče posodabljati in nadgrajevati, da lahko zadovoljuje spreminjajoče se potrebe svojih uporabnikov, ali pa ga nadomestiti z novim. V nasprotnem primeru postaneta protokol in omrežje, v katerem se ta uporablja, *okostenela*. Okostenelost protokolov (angl. *protocol ossification*) je postala precej pereča težava v modernih omrežjih, še zlasti na transportnem sloju v internetu. V zadnjih letih je bilo namreč kar nekaj neuspešnih poskusov nadgradnje obstoječih in vpeljave novih transportnih protokolov. Leta 2014 je bila na primer standardizirana razširitev protokola TCP (TCP Fast Open) [7], ki omogoča hitrejšo ponovno vzpostavitev povezave z istim strežnikom (na podoben način kot QUIC), a ta ni zaživela. Podobno usodo doživljata tudi razširitev MPTCP (Multipath TCP) [8], ki omogoča hkratno uporabo več omrežnih poti med dvema gostiteljema, in povsem nov transportni protokol SCTP (Stream Control Transmission Protocol) [9], [10]. Zaradi okostenelosti transportnega sloja imajo danes aplikacijski protokoli v praksi tako možnost izbire le med protokoloma TCP (z omejenim naborom razširitev[5]) in UDP.

Glavni razlog za pojav okostenelosti so vmesne omrežne naprave (angl. *middlebox*), kot so požarni zidovi, pretvorniki internetnih naslovov (Network Address Translation, NAT), izravnalniki obremenitev (ang. *load balancer*) in naprave DSI (Deep Packet Inspector). Te naprave pregledujejo, filtrirajo in posegajo v omrežni promet in praviloma blokirajo vse njim neznane protokole ali njihove nadgradnje. V praksi je tako pogosto blokiran ves promet, ki ne uporablja uveljavljenih transportnih protokolov TCP in UDP, včasih pa je blokiran celo slednji.

Dodatno težavo pri uvajanju novih transportnih protokolov povzroča dejstvo, da so ti udejanjeni v okviru jeder operacijskih sistemov (angl. *kernel space*), kjer jih je težje redno posodobljati, kar še posebej velja za vmesne omrežne naprave. Ker ne sledijo razvoju protokolov, izhaja predstava teh naprav o »normalnem« prometu in funkcionalnosti protokolov pogosto še iz časa, ko so bile nameščene. Čeprav sta nadgradnja obstoječih protokolov in uvedba novih na terminalih lažji, obstaja precejšnja verjetnost, da jih bodo vmesne naprave obravnavale kot slabonamerne in jih zato blokirale ali toliko zakasnile, da bo postala uporabniška izkušnja nesprejemljiva.

QUIC je eden od prvih protokolov, ki je zaradi slabih izkušenj iz preteklosti načrtno zasnovan z odpornostjo proti okostenelosti. Za najučinkovitejšo metodo preprečevanja okostenelosti velja šifriranje nadzorne protokolne informacije v glavah sporočil, saj se vmesne naprave težje odločijo o blokadi prometa, če o njem nimajo vseh informacij. Pri protokolu QUIC je tako

[3] Spletni strežnik navadno uporablja vnaprej znana vrata 80.

[4] Na primer, če mobilni telefon preklopi med lokalnim brezžičnim in mobilnim omrežjem ali poteče preslikava med javnim in zasebnim IP-naslovom v mehanizmu NAT (Network Address Translation).

[5] Omeniti je treba, da niso vsi mehanizmi protokolov okostenelosti podvrženi v enaki meri. Primer je mehanizem za nadzor zamašitev pri TCP, ki ga je mogoče razmeroma redno nadgrajevati.

šifrirano celotno sporočilo vključno z glavo, razen polja za različico protokola, in nekaj binarnimi vrednostmi, ki določajo vrsto sporočila.

K preprečevanju okostenelosti pripomore tudi dejstvo, da QUIC prenaša svoja sporočila s sporočili protokola UDP. To je v pomoč z dveh vidikov:

- UDP je uveljavljen transportni protokol, v katerega se vmesne naprave manj vpletajo;
- uporabniki protokola UDP so udejanjeni v okviru t. i. uporabniškega prostora operacijskih sistemov (angl. *user space*), kar omogoča hitrejše posodobitve, nadgradnje in evolucijo, kot če bi bili udejanjeni v jedru operacijskih sistemov.

QUIC ima še nekatere druge mehanizme in lastnosti, ki pomagajo preprečevati okostenelost. Eden takih je t. i. *mazanje* (angl. *greasing*), ki preprečuje »zapečenost« vrednosti v poljih za oznako različice protokola oziroma njegove razširitve [11]. Gostitelji redno spreminjajo vrednosti v teh poljih po predpisanem postopku, čeprav vedno uporabljajo isto različico protokola. S tem preprečujejo, da bi se vmesne naprave »navadile« na vedno iste vrednosti, ko bi se pojavile nove različice protokola in s tem nepričakovane vrednosti v teh poljih, pa bi te blokirale.

K odpornosti proti okostenelosti pripomorejo tudi mehanizem pogajanj o različici uporabljenega protokola in v standardih določene lastnosti, ki se v prihodnjih različicah ne morejo spremeniti. Oboje omogoča večjo predvidljivost glede protokola QUIC in s tem zmanjša verjetnost zavračanja njegovih novih različic v vmesnih omrežnih napravah.

Kljub številnim mehanizmom za preprečevanje okostenelosti pa specifikacija protokola zahteva, da morajo biti aplikacije, ki uporabljajo QUIC, pripravljene na uporabo drugih protokolov, če vmesne naprave blokirajo ta protokol ali protokol UDP, ki ga QUIC uporablja [12].

6 Zaključek

QUIC je danes podprt v vseh bolj razširjenih brskalnikih, vključno z Googlovim Chromom, Mozillinim Firefoxom, Applovim Safarijem in Microsoftovim Edgeem. Vsa omenjena podjetja oziroma organizacije imajo tudi lastne izvedbe strežnikov QUIC [13]. Med preostalimi bolj znanimi strežniki, ki podpirajo ta protokol, najdemo še Nginx in LiteSpeed. Po nekaterih podatkih uporablja QUIC že več kot 8 % spletnih mest [14]. Od večjih ponudnikov storitev ga na svojih strežnikih podpirajo Google, Facebook in Cloudflare.

Trenutno je v fazi standardizacije več razširitev protokola [15]. Med njimi naj omenimo možnost hkratne uporabe več omrežnih poti v okviru ene povezave (angl. *multipath*), podobno kot bi to omogočila TCP-jeva razširitev MPTCP.

Zahvala

To delo je podprla Javna agencija za znanstvenoraziskovalno in inovacijsko dejavnost Republike Slovenije (ARIS) v okviru raziskovalnega programa ICT4QoL – Informacijsko komunikacijske tehnologije za kakovost življenja (P2-0246).

Literatura

[1] Jim Roskind “QUIC: Design Document and Specification Rationale”, https://docs.google.com/document/d/1RNHkx_VvKWyWg6Lr8SZ-saqsQx7rFV-ev2jRFUoVD34/edit (13. 9. 2024)

[2] Iyengar Jana, Thomson Martin (ur.) “RFC 9000 – QUIC: A UDP-Based Multiplexed and Secure Transport”, *Internet Engineering Task Force (IETF)*, https://datatracker.ietf.org/doc/html/rfc9000 (13. 9. 2024).

[3] Henrik Nielsen et al. “RFC 2616 – Hypertext Transfer Protocol - HTTP/1.1”, *Internet Engineering Task Force (IETF)*, https://datatracker.ietf.org/doc/html/rfc2616 (13. 9. 2024).

[4] Mike Belshe, Roberto Peon, Martin Thomson “RFC 7540 – Hypertext Transfer Protocol Version 2 (HTTP/2)”, *Internet Engineering Task Force (IETF)*, https://datatracker.ietf.org/doc/html/rfc7540 (13. 9. 2024).

[5] Iyengar Jana, Ian Swett (ur.) “RFC 9002 – QUIC Loss Detection and Congestion Control”, *Internet Engineering Task Force (IETF)*, https://datatracker.ietf.org/doc/html/rfc9002 (13. 9. 2024).

[6] Phil Karn, Craig Partridge “Improving round-trip time estimates in reliable transport protocols”, *ACM Transaction on Computer Systems*, 9(4), str. 364–373, 1991.

[7] Yuchung Cheng, Jerry Chu, Sivasankar Radhakrishnan, Arvind Jain “RFC 7413 – TCP Fast Open”, *Internet Engineering Task Force (IETF)*, https://datatracker.ietf.org/doc/html/rfc7413 (13. 9. 2024).

[8] Alan Ford, Costin Raiciu, Mark J. Handley, Olivier Bonaventure “RFC 6824 – TCP Extensions for Multipath Operation with Multiple Addresses”, *Internet Engineering Task Force (IETF)*, https://datatracker.ietf.org/doc/html/rfc6824 (13. 9. 2024).

[9] Malleswar Kalla et al. “RFC 2960 - Stream Control Transmission Protocol”, *Internet Engineering Task Force (IETF)*, https://datatracker.ietf.org/doc/html/rfc2960 (13. 9. 2024).

[10] Randall R. Stewart, Michael Tüxen, Karen Nielsen “RFC 9260 - Stream Control Transmission Protocol”, *Internet Engineering Task Force (IETF)*, https://datatracker.ietf.org/doc/html/rfc9260 (13. 9. 2024).

[11] Martin Thomson “RFC 9287 – Greasing the QUIC Bit”, *Internet Engineering Task Force (IETF)*, https://datatracker.ietf.org/doc/rfc9287/ (13. 9. 2024).

[12] Mirja Kühlewind, Brian Trammell “RFC 9308 – Applicability of the QUIC Transport Protocol”, *Internet Engineering Task Force (IETF)*, https://datatracker.ietf.org/doc/html/rfc9308 (13.9.2024).

[13] Implementations · quicwg/base-drafts Wiki · GitHub, https://github.com/quicwg/base-drafts/wiki/Implementations (13. 9. 2024)

[14] Usage statistics of QUIC for websites https://w3techs.com/technologies/details/ce-quic (13. 9. 2024).

[15] QUIC Working Group, https://quicwg.org/ (13. 9. 2024)

Grega Jakus je leta 2007 diplomiral, leta 2012 pa doktoriral na Fakulteti za elektrotehniko Univerze v Ljubljani. Leta 2021 je bil izvoljen v naziv izredni profesor za področje elektrotehnike. Njegovo glavno raziskovalno področje je interakcija med napravami in njihovimi uporabniki, aktiven pa je tudi na področju raziskav in razvoja komunikacijskih protokolov.