Зборник радова Факултета техничких наука, Нови Сад

UDK: 004.8
DOI: https://doi.org/10.24867/15BE05Beker

ЗАМЕНЕ ЗА TCP ПРОТОКОЛ

ALTERNATIVES FOR TCP PROTOCOL

Александар Бекер, *Факултет техничких наука, Нови Сад*

Област – ПРИМЕЊЕНЕ РАЧУНАРСКЕ НАУКЕ И ИНФОРМАТИКА

Кратак садржај – *Обрада особина TCP протокола, кратак осврт на алтернативе истог и начине на који они превазилазе недостатке TCP-а. Акценат је стављен на QUIC протокол, где су описане особине протокола, док је начин функционисања приказан кроз библиотеку LSQUIC и клијентску апликацију која је развијене за потребе задатка.*

Кључне речи: *TCP, UDP, QUIC, protokol, DCCP, SCTP, MPTCP, lsquic*

Abstract – *Process the features of the TCP protocol, look at alternatives to it and the ways in which they overcome the disadvantages of TCP. Emphasis will be placed on the QUIC protocol, where the properties of the protocol will be described, and the way it works will be shown through the LSQUIC library and the client application which is developed for the needs of the task.*

Keywords: *TCP, UDP, QUIC, protocol, DCCP, SCTP, MPTCP, lsquic*

1. УВОД

Задатак овог рада јесте да се обради протокол четвртог нивоа *OSI (Open Systems Interconnection)* модела и то конкретно *TCP* протокол, обраде недостаци и размотре друга могућа решења. Развијен 1974. године, а данас доминантан транспортни протокол, свакако да има својих добрих страна, али исто тако има и мана које ће бити побројане и анализиране. Временом су настали и други протоколи који превазилазе одређене недостатке *TCP* (*Transmission Control Protocol*) протокола, а енкапсулирају исте или сличне функционалности. Неки од тих протокола биће наведени и обрађени у овом раду.

У првом поглављу дат је кратак увод у рад. У другом поглављу је детаљније описан начин на који ради *TCP*, његове добре стране, као и мане овог протокола. Треће поглавље односи се на опис протокола четвртог нивоа *OSI* модела, који су одабрани као подобне замене, где су такође описане добре стране тих протокола и које мане *TCP* протокола надомешћују. У четвртом поглављу, описан је *QUIC* протокол, који је производ компаније *Google* и који је званично одобрен маја текуће године.

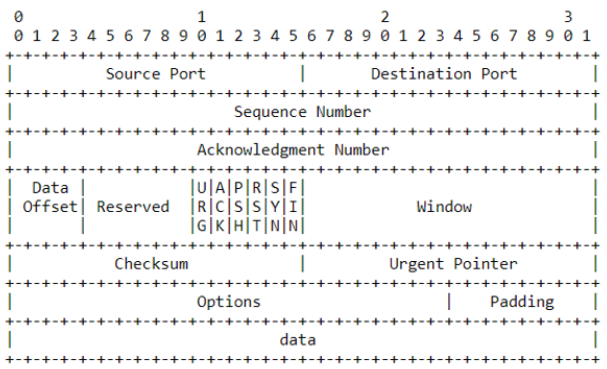
Након тога, у петом поглављу дат је закључак, којим смо сумирали претходно урађено и предложили даље кораке развоја рада.

2. *TCP* ПРОТОКОЛ

TCP протокол је транспортни протокол који омогућава комуникацију између два удаљена процеса, која су идентификована портовима. Предуслов за транспорт података је отварање конекције између две комуникационе стране.

TCP протокол групише одређене бајтове података у сегмент, а затим сегмент користи као основна јединицу преноса података. Сегменти се енкапсулирају у *IP* пакет, а затим се транспортују кроз мрежу.

Сваки сегмент садржи део са подацима и обавезан део који се назива заглавље сегмента. Дешава се и да сегменти не садрже податке, уколико се користе за потврђивање пријема других сегмената. Заглавље сегмента садржи обавезна и опциона поља, тако да сама дужина није фиксна. На слици 1 приказан је формат заглавља *TCP* сегмента



Слика 1. *Формат заглавља TCP сегмента [1]*

Пре слања података неопходно је да се успостави комуникациона веза, процесом који се назива *three-way handshaking*. Обе стране, пријемна и предајна, морају учествовати у овом процесу, с обзиром на могућност одвијања комуникације у оба смера, истовремено. За успостављање конекције било је неопходно усаглашавање обе стране. Међутим, када је реч о затварању исте довољно је само иницирање неког од учесника. Полузатварање је процес који је омогућава да једна страна затвори конекцију, док други смер остаје отворен.

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био доцент др Жељко Вуковић.

2.1. Предности *TCP* протокола

Циљ рада је да идентификујемо предности и мане *TCP* протокола, односно ситуације у којима је овај протокол прикладно решење за проблеме транспортног нивоа.

Једна од најзначајнијих особина јесте да *TCP* протокол има механизме за гаранцију испоруке података, као и детекције и корекције грешака приликом преноса. Један од начина је подржан на нивоу самог сегмента, коришћењем *TCP* checksum поља у оквиру заглавља. Други начин се односи на постављање *ACK flag*-а у заглављу сегмента и слање таквог сегмента, односно поруке потврде. Овде заправо користимо механизам потврђивања пријема података и то је још један од начина контроле грешака.

Наравно, неретко се дешавају и ситуације да се оваква порука потврде, енкапсулирана у *IP* пакет, загуби у мрежи. Како би се избегло потенцијално непотребно кашњење и чекање потврде, поставља се време у ком се очекује пристизање сегмента са постављеним *ACK flag*-ом. За ову сврху се користи *RTO* (*Retransmission Time-Out*) тајмер.

Дешавају се и ситуације да изгубљени пакет ипак стигне на пријемну страну, а да је у међувремено пакет опет послат. Тада на одредишту имамо дупликате, али се одбацује онај који је накнадно пристигао.

Још једна од битних особина протокола јесте да се гарантује испорука пакета у редоследу који дефинише пошиљалац. *TCP* протокол је оријентисан на ток података и води рачуна о реконструкцији редоследа сегмената на пријемној страни. Тако нам, као што смо претходно навели, омогућава да на одредишту прихватамо сегменте у редоследу у ком су и послати.

Током комуникације, неретко долазимо у ситуацију да страна која шаље и пријемна страна не могу да усагласе брзину слања, односно примања пакета. У циљу спречавања таквог загушења мреже, *TCP* протокол нуди механизам контроле протока који се назива *sliding window*. Идеја овог механизма јесте да се комуникационе стране договоре о величини сегмента, односно количини података коју ће у једној итерацији размењивати. То је омогућено постављањем поља, у оквиру сегмента, који се назива *Window Size*.

2.2. Мане *TCP* протокола

Сврха овог потпоглавља је да наведемо све особине које су мане *TCP* протокола, како бисмо касније могли анализирати те мане и решења која имплементирају други протоколи.

Сам почетак комуникације је приметно спорији у поређењу са брзином даље комуникације, услед неопходног успостављања конекције које претходи преносу података. Процес успостављања конекције успорава саму мрежу.

Континуално слање података и њихово преузимање у редоследу у ком су и послате, представља предност, али иману *TCP* протокола. У одређеним ситуацијама, губљење једног пакета неће бити уочено на пријемној страни, тачније неће утицати на жељени резултат.

Уколико се користи у оквиру *LAN* (*Local Area Network*) мреже може се десити да не искористимо погодности протокола јер је дизајниран и оптимизован пре свега за *WAN* (*Wide Area Networks*) мрежу. Такође треба напоменути да *TCP* користи *HTTP/2*.

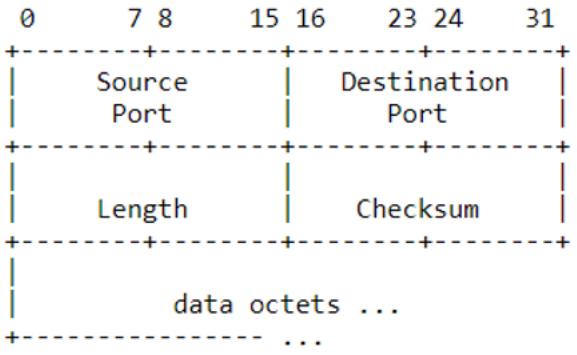
3. ДРУГИ ПРОТОКОЛИ ЧЕТВРТОГ НИВОА

Како смо у претходном поглављу побројали основне особине, предности и мане *TCP* протокола, сада ћемо више пажње посветити протоколима четвртог нивоа који се могу сматрати заменама за *TCP*.

3.1. *UDP* протокол

User Datagram Protocol (UDP) је настао 1980. године, а формално дефинисан и објављен у документу RFC 768 *[2]*. *TCP* и *UDP* су протоколи који се налазе на транспортном нивоу *OSI* модела. Протоколи нису дијаметрално различити, али одређене особине су од круцијалног значаја за један, односно други протокол чиме дефинишу његову сврху.

Пакети који се размењују *UDP* протоколом називамо *UDP datagram*. На слици 2 можемо погледати структуру једног таквог пакета. Посматрајући у односу на *TCP segment*, можемо приметити да је структура једноставнија.



Слика 2. *Формат заглавља UDP datagram-а [2]*

3.1.1. Разлике *TCP* и *UDP* протокола

TCP је *connection-oriented* протокол, који омогућава комуникацију између пошиљаоца и примаоца тек по успостави везе. Док је *UDP connectionless* протокол, што би значило да се не захтева успостава везе пре него што комуникација почне.

Битна особина коју треба нагласити јесте да *UDP* протокол не захтева успоставу везе и не гарантује испоруку свих пакета који су послати. Што се *TCP*-а тиче, он гарантује испоруку свих пакета, али захтева и успоставу везе пре почетка слања података. *UDP* не гарантује испоруку у тачно одређеном редоследу, нити омогућава превенцију од загушења тока података, што *TCP* омогућава.

Што се примене протокола тиче, *UDP* користимо у ситуацијама попут клијент-сервер конфигурације, где клијент шаље упит и очекује брз и кратак одговор. Поред тога, овај протокол интензивно се користи у *real-time* мултимедијалним апликацијама. Насупрот томе, *TCP* протокол је своју примену нашао у

ситуацијама у којима нам је неопходна поуздана комуникација, без губитка података.

3.2. Друге замене за *TCP* протокол

Осим набројана два најраспрострањенија протокола транспортног нивоа – TCP и UDP, временом су настали протоколи који покушавају да реше одређене проблеме специфичне за одређена поља примене и начине преноса, за које ни један од ова два постојећа протокола не представља адекватно решење.

Datagram Congestion Control Protocol (*DCCP*) је протокол транспортног нивоа *OSI* модела објављен 2006. године [3]. Имплементирано је поуздано успостављање, прекид везе, експлицитно обавештење о загушењу (*ECN*), контрола загушења и *feature nagotiation*.

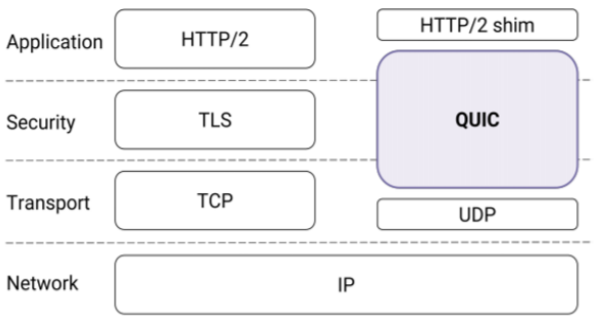
Поред *DCCP*-а, појавили су се и *Stream Control Transmission Protocol* (*SCTP*) 2007. године [4], *MultiPath TCP* (*MPTCP*) који се појавио у Јануару 2013. год [5].

QUIC протокол је комуникациони протокол развијен од компаније *Google*, који настоји да превазиђе мане данашњих комуникационих протокола и устали се као доминантан комуникациони протокол. Почетак развоја био је 2012. године, док је формално објављен тек маја 2021. године [6].

4. *QUIC* ПРОТОКОЛ

Посматрано из угла замене за *TCP* протокол, можемо констатовати да *QUIC* доста обећава. У прилог томе говори и чињеница да иза овог протокола стоји један гигант као што је *Google*. На серверској страни QUIC је имплементиран у светским гигантима попут *Google*, *Facebook*, *Microsoft*, *Apple*, итд. Текуће године водећи европски интернет провајдери објавили су информације да 20% њихових пакета иде преко QUIC-а.

Структура овог протокола нам омогућава да заменимо већину традиционалног HTTP стека, где се под тим подразумева HTTP/2, TLS и TCP приказан са леве стране на слици 3. Саобраћај је енкриптован, подаци су шифровани и обављена је превенција од измене истих. Комуникација подразумева енкриптовани *handshake* што ће нам омогућити да смањимо кашњење користећи унапред познате креденцијале сервера при поновној комуникацији. Тиме смо практично уклонили непотребно „представљање“ при свакој наредној комуникацији са истим сервером.



Слика 3. *Структура QUIC протокола [7]*

Како је QUIC протокол развијен скоро 40 година након TCP протокола, тако је у могућности да анализом претходних протокола и употребом савременије технологије развије напреднији вид комуникације.

Главне карактеристике протокола су:

- смањење кашњења,
- реализација комуникације кроз токове података,
- реакција на губитак података (пакета) и
- контрола загушења,
- безбедност и
- могућност одржања сесије и поред промене иницијалне адресе

Three-way handshake постао је оптерећавајући фактор који умногоме утиче на брзину успостављања конекције, а потом може и да проузрокује кашњење. Тако је QUIC нашао решење да се *handshake* не обавља у три етапе као раније, већ у два корака, док уколико је већ претходно успостављена конекција између клијента и сервера, да то буде само један корак.

Поред тога, *QUIC* обезбеђује комуникацију кроз випе токова података, чиме се повећава концептуална независност. Тиме је обезбеђено да семантички различити подаци путују кроз исту конекцију. Заправо, *QUIC* ће креирати неколико токова података преко исте *UDP* конекције. За разлику од *TCP*-а, који секвенцијално доставља пакете и нема могућност да избегне *head-of-line blocking*[1], QUIC осигурава да изгубљени *UDP* пакет може једино утицати на ток података који преноси тај пакет, чиме не може доћи до загушења. Веома важна особина токова је да се они идентификују путем *stream ID*. Оно што омогућава креирање више токова је то што су подаци енкапсулирани у један или више фрејмова, при чему један *QUIC* пакет може да садржи неколико фрејмова који су везани за токове података.

Како је реакција на губитак података код *TCP*-а доста поуздана, али спора, тако *QUIC* омогућава унапређење и овог сегмента. За разлику од *TCP*-а, *QUIC* ће започети опоравак од губитка пакета, одмах након што се исти изгуби, где се епоха опоравка завршава онда када се било који пакет, који је послат након тога, прихвати на другој страни [8]. Опоравак се може реализовати:

- поновним покушајем преношења истих података,
- слањем измењеног frame-а или
- одбацивањем frame-а

Како је *QUIC* базиран на *UDP*-у који нема имплементирану контролу загушења, стога он то ради на апликативном нивоу. Није везан ни за један конкретан алгоритам контроле загушења, већ има интрефејс који је *plug*-абилан где притом дозвољава различите имплементације. Подразумевани контролер загушења је *Cubic*. Као што смо у претходном потпоглављу дефинисали, *QUIC* је у стању да детектује губитак пакета, а ту информацију може да искористи да прилагоди контролер загушења. Сваки контролер задужен је за једну путању, где притом нема интерференције између контролера на различитим путањама.

QUIC примењује екрипцију на транспортни ниво. Цео *UDP* садржај који се преноси је аутентификован и

[1] head-of-line blocking - феномен је који ограничава перформансе и јавља се када су два независна захтева непотребно блокирана због ограничења протокола

обављена је превенција од потенцијалне измене и практично сви подаци који се преносе су енкриптовани. Делови пакета који нису енкриптовани, битни су нам за рутирање или декриптовање пакета. Ту спадају *Flags, Connection ID, Version Number, Diversification Nonce* и *Packet Number.*

QUIC протокол, као што је претходно поменуто, за сваку конекцију чува посебну вредност – *Connection ID.* Овај идентификатор јединствено идентификује сваку конекцију која је остварена овим протоколом. Чиме имамо могућност да наставимо отворену сесију са једне адресе, на другој адреси.

4.1. Демонстрација истраживања

За демонстрацију рада *QUIC* протокола искоришћена је библиотека *LSQUIC* [9], како бисмо на једноставнијем примеру показали на који начин се подаци размењују применом поменутог протокола. Ова библиотека представља *open-source* имплементацију *QUIC*-а.

Било је неопходно *setup*-овати окружење. Подигнута је виртуелна машина са *Linux* оперативним системом. Након постављања виртуелне машине, потребно је било креирати погодно окружење за компајлирање и стартовање библиотеке.

Након стартовања библиотеке, прешли смо на практични део. Подигнута је једноставна апликација за потребе задатка. Апликација је *deploy*-ована и такође покренута. Следећи задатак је био да размењујемо податке са апликацијом користећи *QUIC* протокол.

Ограничавајућа околност при изради задатка било је то што је протокол веома млад и стога нема довољно података и информација које би биле од користи при практичном делу. Тако смо се при имплементацији морали ослонити на *RFC* документ [6] и документацију за *LSQUIC* библиотеку [9].

5. ЗАКЉУЧАК

Циљ рада био је да се осврнемо на *TCP* протокол, да сагледамо његове карактеристике и да истражимо алтернативе за исти. Акценат рада стављен је на *QUIC* протокол, стога је сврха задатка упознавање и демонстрација рада са истим. Дефиниција и обрада начина функционисања тог протокола дају нам релевантне чињенице које га истичу у односу на *TCP* протокол.

Приказана библиотека *LSQUIC* омогућава једноставну имплементацију клијентске стране софтверских решења за које је потребно подржати комуникацију QUIC протоколом.

У даљем току истраживања неопходно је идентификовати и анализирати библиотеке које омогућавају имплементацију и серверске стране. Од значаја може бити и анализа и поређење других у раду побројаних транспортних протокола виђених као алтернатива за TCP и UDP.

6. ЛИТЕРАТУРА

[1] Fu-Hau Hsu, Yan-Ling Hwang, Cheng-Yu Tsai, Wei-Tai Cai, Chia-Hao Lee, KaiWei Chang, “TRAP: A Three-Way Handshake Server for TCP Connection Establishment”, Новембар 2016.

[2] J.Postel, “RFC 768”

[3] E. Kohler, M. Handley, S. Floyd, “RFC 4340”

[4] R. Stewart, “RFC 4960”

[5] A. Ford, C. Raiciu, M. Handley, O. Bonaventure, “RFC 6826”

[6] J. Iyengar, M. Thomson, “RFC 9000”

[7] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, Jeff Bailey, Jeremy Dorfman, Jim Roskind, Joanna Kulik, Patrik Westin, Raman Tenneti, Robbie Shade, Ryan Hamilton, Victor Vasiliev, Wan-Teh Chang, Zhongyi Shi, “The QUIC Transport Protocol: Design and Internet-Scale Deployment”, Август 2017.

[8] J. Iyengar, M. Thomson, “RFC 9002”

[9] https://lsquic.readthedocs.io/en/latest/ (приступљено у августу 2021.)

Кратка биографија:

Александар Бекер рођен је у Новом Саду 1997. године. Мастер рад на Факултету техничких наука из области Електротехнике и рачунарства – Примењене рачунарске науке и информатика одбранио је 2021. године.

контакт: acabeker@gmail.com