Matematički fakultet,
Univerzitet u Beogradu

MASTER RAD

Pouzdan prenos podataka zasnovan na UDP transfer protokolu

student: Vladimir Stojanović

indeks: 1018/2012

smer: Računarstvo i informatika

mentor: doc. dr Miroslav Marić

Beograd, 2013.

Sadržaj

1. Uvod

Osnovni transfer protokoli koji su danas u masovnoj primeni su TCP (*Transmission Control Protocol*) [1] i UDP (*User Datagram Protocol*) [18]. TCP se koristi kada je prioritet pouzdanost, dok je UDP zasnovan na drugim principima – većoj brzini i smanjenoj razmeni kontrolnih informacija.

TCP je *de facto* podrazumevani transfer protokol za svaku primenu u kojoj je neophodan pouzdan prenos podataka. Zaslužio je svoju dominantnost godinama u kojima je primenjivan bez otežavajućih okolnosti. Ali TCP je projektovan u drugačijim mrežnim uslovima od onih prisutnih danas. U vreme njegovog nastanka je bilo malo korisnika i veze su bile manjeg kapaciteta. U današnjim uslovima sa velikim brojem korisnika koji žele da visokim brzinama razmenjuju ogromne količine podataka sa udaljenih lokacija sve slabosti TCP-a dolaze do izražaja. Činjenica je da TCP teško koristi dostupan propusni opseg veze. Vremenom, sa daljim porastom propusnog opsega veza i sa još više korisnika, slabosti TCP-a će još više umanjiti njegove performanse.

UDP ne pruža praktično nikakve garancije prenosu podataka, već samo osnovnu funkcionalnost prenosa podataka. Sa druge strane, TCP je isuviše konzervativan - vodi računa o mnogo detalja prilikom transfera kako bi obezbedio pouzdanost. Obrađivanje ovih detalja u TCP-u povećava procenat kontrolnih podataka i samim tim smanjuje procenat koji se koristi na podatke aplikacije, uz povećano trošenje sistemskih resursa. Ustanovljeno je da su garancije TCP-a nepotrebno velike i da je moguće postići pouzdan prenos podataka sa manje garancija. Tako da je moguće smanjiti konzervativnost TCP-a i samim tim smanjiti količinu kontrolnih podataka, vreme obrade tih podataka u sistemu i što je najbitnije – povećati prilagodljivost različitim mrežnim uslovima i poboljšati performanse.

Tako se pojavio prostor za nove transfer protokole koji se po količini garancija koje pružaju mogu svrstati između TCP-a i UDP-a. Ti transfer protokoli se mogu dobiti nadogradnjom UDP-a i povećanjem njegove funkcionalnosti. Jedan od takvih protokola je UDT (*UDP-based Data Transfer Protocol*) [21], [22], [23].

Drugačiji pristup rešavanju problema iskorišćenosti dostupnog propusnog opsega, prisutan kod UDT transfer protokola, ima svojih prednosti nad TCP-om. Pri analizi rezultata ovog rada primećena je veća efikasnost i stabilnost UDT protokola.

Sam rad je organizovan u 5 poglavlja. U poglavlju 2 se analiziraju osnovni transfer protokoli – TCP i UDP, njihova arhitektura i funkcionalnost. Poglavlje 3 nam predstavlja UDT transfer protokol. Detaljno se analizira sam UDT protokol i prikazuju se sličnosti i razlike u odnosu na TCP transfer protokol. U poglavlju 4 se opisuje metodologija uporednih testiranja TCP i UDT transfer protokola. Takođe, detaljnije se prikazuje interfejs UDT protokola kao i korišćena UDT test aplikacija. Na kraju 4. poglavlja su prikazani analizirani rezultati izvršenih testova. Poglavlje 5 iznosi zaključak.

Autor ovog rada bi želeo da se zahvali doc. dr Miroslavu Mariću, kao svom mentoru, zbog uvođenja u oblast računarskih mreža i razmatranja problematike ispitane u ovom radu. Kao i na iskazanom strpljenju, podršci i savetima pri izradi ovog rada. Autor takođe duguje zahvalnost prof. dr Aleksandru Jovanoviću zbog beskrajnih diskusija na temu ovog rada pri saradnji u Grupi za inteligentne sisteme Matematičkog fakulteta, Univerziteta u Beogradu. Kao i zbog velike pomoći pri stručnom usavršavanju. Naravno, zahvalnost zaslužuje i prof. dr Miodrag Živković zbog svih korisnih sugestija koje su svakako pomogle poboljšanju ovog rada.

2. Osnovni transfer protokoli

2.1. TCP transfer protokol

TCP pruža pouzdani, uređeni, proveren prenos podataka između programa na mrežno povezanim računarima [16], [17], [18].

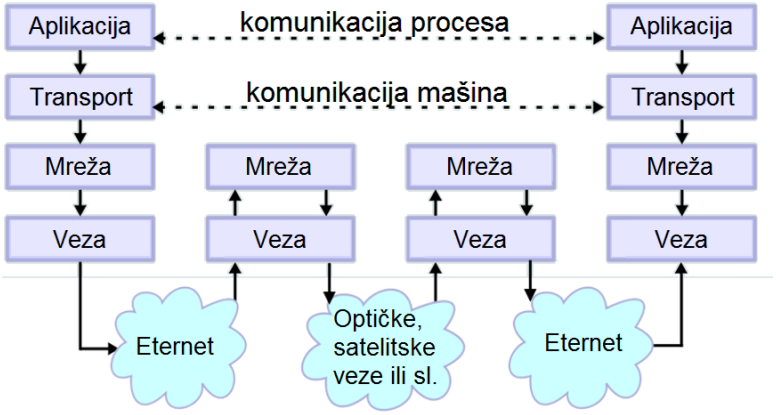
2.1.1. Istorija TCP protokola

U oktobru 1986. godine desio se prvi u nizu kolapsa na Internetu izazvanih zagušenjem. Tada je bilo jasno da se mora uvesti transfer protokol koji bi sprečio kolapse u budućnosti, kada se očekivao dalji porast broja korisnika [5].

Koreni TCP-a su predstavljeni u radu [2], gde je opisan mrežni transfer protokol za razmenu podataka korišćenjem paketa između mrežnih čvorova. Tada je TCP bio zamišljen kao jedna celina (*Transmission Control Program*) ali je kasnije uvedena modularnost podelom na transfer protokol (najčešće TCP) i IP (*Internet Protocol*), pa je ovaj model dobio ime TCP/IP.

2.1.2. Mrežna funkcija TCP protokola

TCP se nalazi u transportnom delu TCP/IP sloja. On pruža usluge komunikacije između aplikacije i IP sloja. Kada aplikacija želi da pošalje veću količinu podataka preko mreže ona pošalje samo jedan zahtev TCP sloju koji se dalje brine o pravljenju i distribuciji paketa.



Slika 2.1. Četiri sloja TCP/IP modela [3]

Nakon prijema podataka za slanje od aplikacije TCP deli podatke u segmente kako bi se lakše preneli preko mreže. Segment je uređena sekvenca bajtova sastavljena od zaglavlja i tela. U zaglavlju su kontrolni podaci dok su u telu podaci dobijeni od aplikacije. Segmenti se dalje prenose do IP sloja.

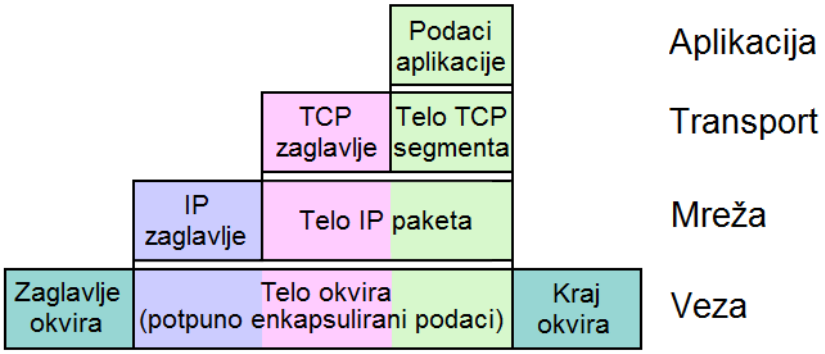
Pozicija Bajt		0								1								2								3							
Bajt	bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Izvorišni priključak																Odredišni priključak														
4	32	Redni broj bajta																															
8	64	Broj potvrde (ako je ACK bit postavljen)																															
12	96	Pozicija podataka				Rezervisano 0 0 0				N S	C W R	E C E	U R G	A C K	P S H	R S T	S Y N	F I N	Veličina prozora														
16	128	Kontrolni zbir																Pokazivač na urgentne podatke (ako je URG bit postavljen)														
20 ...	160 ...	Opcije (ako je *Pozicija podataka* > 5. Po potrebi dopunjeno "0" bajtovima.) ...																															

Slika 2.2. TCP zaglavlje

Zaglavlje TCP segmenta se sastoji od sledećih polja:

- Izvorišni i odredišni priključak (*port*) (16 bita) određuju priključke aplikacije na strani pošiljaoca i primaoca, respektivno.
- Redni broj bajta (32 bita) određuje početni broj bajta za slanje ili najveći redni broj bajta koji se trenutno šalje u zavisnosti od toga da li je SYN bit postavljen ili ne.
- Broj potvrde (32 bita) čuva kumulativni broj bajtova koji su primljeni.
- Pozicija podataka (4 bita) kazuje na kojoj poziciji (mereno u 32-bitnim rečima) TCP segmenta počinju podaci i mora biti u opsegu od 5 do 15 reči.
- Rezervisana 3 bita se čuvaju za eventualnu buduću upotrebu, ako se ne koriste moraju se postaviti na nulu.
- Flagovi (9 bita):
  - NS flag za korišćenje eksplicitne objave zagušenja.
  - CWR flag označava poruku bez bitnog sadržaja (kada je uključen ECE flag)
  - ECE flag signalizira zagušenje na mreži.
  - URG flag označava segment sa urgentnom porukom.
  - ACK flag se koristi za povratne informacije od strane primaoca.
  - PSH flag označava da se podaci segmenta moraju odmah proslediti aplikaciji.
  - RST flag zahteva ponovno uspostavljanje veze.
  - SYN flag se koristi pri uspostavljanju veze.
  - FIN flag označava kraj transfera.
- Veličina prozora (16 bita) za objavljivanje veličine prozora primaoca (kontrola toka).
- Kontrolni zbir (16 bita) proverava greške u TCP segmentu (zaglavlju i telu).
- Pokazivač na urgentne podatke određuje poziciju urgentnih podataka u TCP segmentu, ako je URG flag uključen.
- Opcije (između 0 i 10 32-bitnih reči) su razni dodaci osnovnoj funkcionalnosti TCP-a. Između ostalih, mogu se koristiti SACK (*Selective Acknowledgement*), FACK (*Forward Acknowledgement*) i vremenske oznake. Korišćenje opcija nije obavezno.

IP sloj funkcioniše tako što razmenjuje komade informacija – pakete. Paket je, kao i segment, uređena sekvenca bajtova sastavljena od zaglavlja i tela. Zaglavlje opisuje izvor i odredište paketa uz potrebne kontrolne informacije. Telo sačinjava TCP segment ili njegov deo, ako je veličina segmenta veća od veličine IP paketa.



Slika 2.3. Enkapsulacija podataka u TCP/IP modelu [3]

Mrežno zagušenje, balansiranje opterećenja veze ili drugih nepredvidivih okolnosti se mogu javiti na javnim mrežama. Usled ovih problema IP paketi mogu da se izgube, dupliraju ili da se dostave u pogrešnom redosledu. TCP sloj otkriva ove probleme i po potrebi zahteva ponovno slanje, sortira neuređene segmente i pomaže da se smanji nastalo zagušenje na mreži.

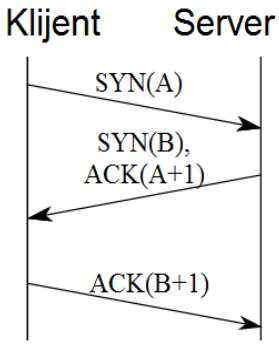
Kada TCP sloj sastavi tražene podatke od primljenih segmenata tek tada ih prosleđuje aplikaciji na strani primaoca i time apstrahuje komunikaciju između aplikacija.

2.1.3. Funkcionalnost TCP protokola

2.1.3.1. Uspostavljanje veze

Za uspostavljanje veze TCP koristi metod nazvan “trostepeno rukovanje”. Server mora prvi da se veže za određeni priključak (*port*) i da osluškuje nadolazeće zahteve za uspostavljanjem veze. Ovo se naziva pasivno otvaranje. Kada server završi sa pasivnim otvaranjem klijent može da započne aktivno otvaranje. Samo uspostavljanje veze se odvija u tri koraka:

1. **SYN:** Klijent šalje paket sa SYN flagom uključenim. Paket sadrži nasumice izabran broj A kao početni u nizu rednih brojeva bajtova za slanje.
2. **SYN-ACK:** Server, po prijemu SYN paketa od strane klijenta, šalje paket sa uključenim SYN i ACK flagovima. Redni broj bajta je novi nasumice izabran broj B, a broj potvrde je (A+1).
3. **ACK:** Klijent, po prijemu SYN-ACK paketa, šalje serveru paket sa postavljenim ACK flagom i brojem bajta (B+1).

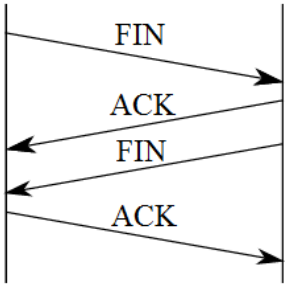


Slika 2.4.
Uspostavljanje veze

U trenutku nakon ova tri koraka i klijent i server su primili potvrdu uspostave veze. I time je uspostavljena dvosmerna veza između njih.

2.1.3.2. Zatvaranje veze

Zatvaranje veze zahteva upotrebu metoda "četvorostepenog rukovanja", gde svaka strana nezavisno od druge zatvara vezu. Kada jedna strana poželi da zatvori svoju vezu ona pošalje paket sa postavljenim FIN flagom. Po prijemu tog paketa druga strana ga potvrđuje paketom sa postavljenim ACK flagom. Veza može biti i poluotvorena, tako što strana koja je zatvorila svoju vezu može samo da prima podatke, ali ne može više da ih šalje. Veza ostaje u ovom stanje sve dok i druga strana ne zatvori svoju vezu. Nakon dvostranog zatvaranja veze, veza ostaje još neko vreme otvorena i priključak zauzet. Ovako se sprečavaju problemi za sledeće veze eventualnim zaostalim segmentima.



Slika 2.5. Zatvaranje veze

2.1.3.3. Pouzdani prenos

TCP označava svaki segment rednim brojem u opsegu od 0 do (2^{32} - 1). Primalac na osnovu brojeva bajtova može ustanovi koji su bajtovi stigli a koji su se izgubili u prenosu, da li su neki segmenti stigli više puta ili su stigli u pogrešnom redosledu. Na kraju, kada svi segmenti uspešno stignu, primalac rekonstruiše podatke. Radi obezbeđivanja pouzdanosti prenosa se koristi tehnika potvrdnih izveštaja (*positive acknowledgement*). Posle svakog primljenog segmenta primalac šalje potvrdu o prijemu. Svaki potvrdni izveštaj sadrži kumulativni broj primljenih bajtova – najveći broj bajtova takav da su svi bajtovi pre njega primljeni. Pošiljalac, naravno, vodi evidenciju o brojevima poslatih bajtova - po slanju ih dodaje na spisak dok ih uklanja po prijemu potvrdnog izveštaja od strane primaoca. Takođe, pošiljalac ima brojač za svaki segment, koji se pokreće po slanju segmenta. Ako se brojač aktivira podrazumeva se da je odgovarajući segment izgubljen i vrši se ponovno slanje. Brojač je neophodan u slučaju gubitka ili korupcije segmenata. Određivanje vremena za ovaj brojač je poseban problem, veoma bitan za dobro funkcionisanje TCP-a, i opisan je detaljno u [13].

2.1.3.4. Otkrivanje grešaka

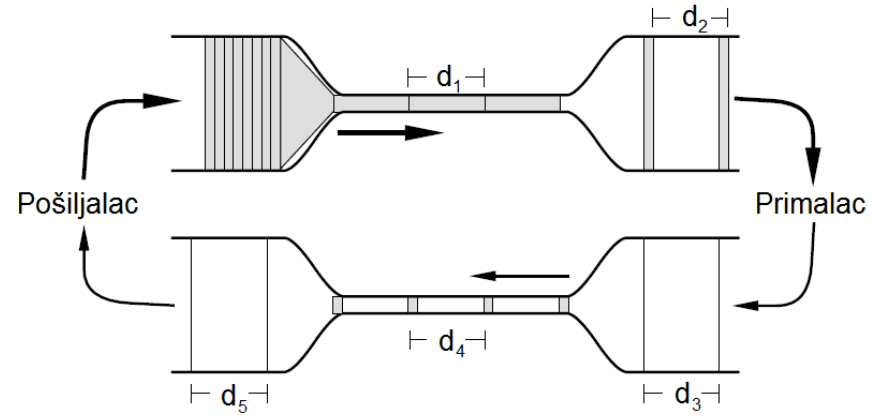
Otkrivanje grešaka se mahom vrši u sloju veze, gde se vrši CRC32 provera okvira. Mogućnost propusta greške je $1/2^{32}$. Ali postojanje kontrolnog zbira u transportnom sloju dalje smanjuje mogućnost propusta i u skladu je sa principom „sa kraja na kraj", čiji značaj je objašnjen u [14].

2.1.3.5. Kontrola toka

TCP izbegava da šalje podatke brže nego što primalac može da ih obradi. Koristi se tehnika „kliznog prozora". Primalac preko potvrdnih izveštaja izveštava pošiljaoca o količini podataka koju u datom trenutku može da obradi. Pošiljalac tada šalje podatke uz tu gornju granicu. Ako se dostigne ta gornja granica pošiljalac čeka novi potvrdni izveštaj kako bi nastavio da šalje podatke.

Veličina "kliznog prozora" označava broj nepotvrđenih bajtova koji u svakom trenutku mogu da budu na vezi. Po pristizanju potvrdnog izveštaj prozor "klizi" dalje i omogućava slanje sledećih segmenata.

TCP tok poštuje princip očuvanja paketa – veza u ravnotežnom stanju tj. sa punim prozorom podataka je "konzervativna". Drugim rečima – takva veza ne može primiti nov paket dok je jedan od starih paketa ne napusti. Poštovanje ovog principa obezbeđuje veću robustnost veze u slučaju zagušenja.



Slika 2.6. Princip očuvanja paketa. Vremenski razmak između uzastopna dva paketa na uskom grlu (d_1) je jednak razmaku između prijema ta dva paketa (d_2), što je opet jednako razmaku između slanja uzastopnih potvrdnih izveštaja (d_3), njihovom putovanju kroz mrežu (d_4) i konačnom prijemu na strani pošiljaoca (d_5).

2.1.3.6. Kontrola zagušenja

2.1.3.6.1. Prozor zagušenja

Radi obezbeđivanja kontrole zagušenja [4], [5], [6], TCP uvodi prozor zagušenja (*Congestion Window*), u oznaci *cwnd*. Koristi se analogno kliznom prozoru u kontroli toka, ali dok klizni prozor određuje koliko podataka primalac može da obradi u datom trenutku, prozor zagušenja određuje koliko podataka u datom trenutku veza može da prenese. Uz kontrolu zagušenja, količina nepotvrđenih podataka koja može da se nađe na vezi u datom trenutku je minimum vrednosti kliznog prozora i prozora zagušenja.

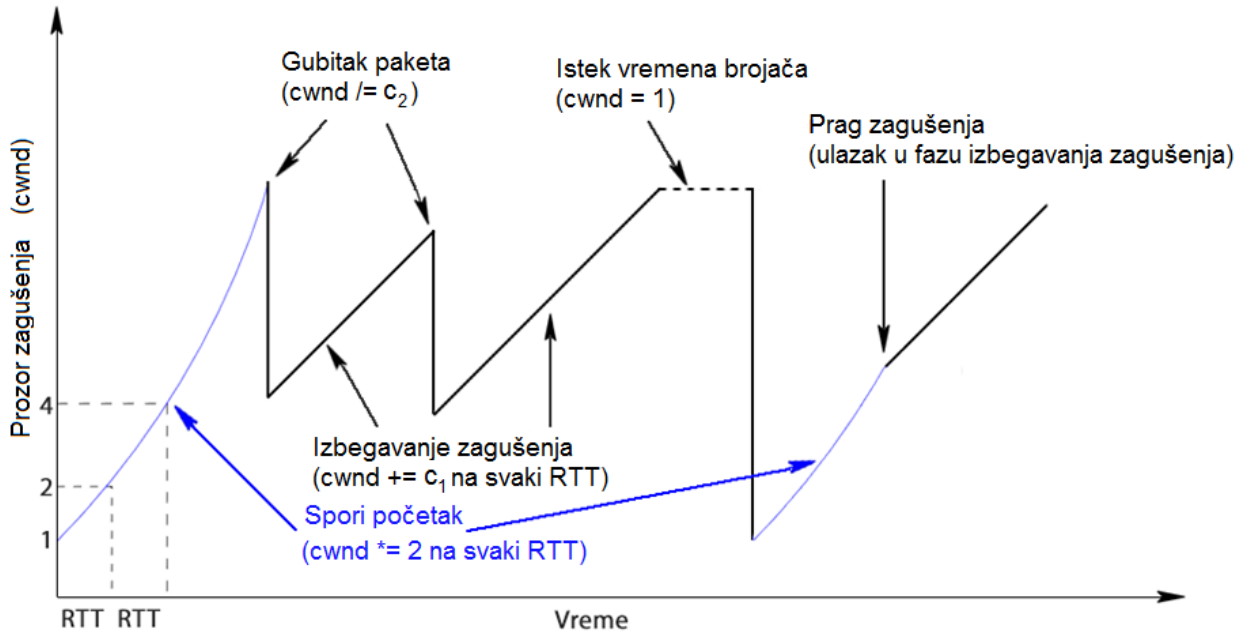
2.1.3.6.2. Spori početak, prag zagušenja i izbegavanje zagušenja

Kontrola zagušenja se odvija u dve faze: faza sporog početka i faza izbegavanja zagušenja.

Faza sporog početka se odvija na samom početku veze kada se efektivno duplira prozor zagušenja posle svakog vremena obilaska paketa (*Round Trip Time*), u oznaci *RTT*, sve dok se primaju potvrdni izveštaji. Čim se primeti gubitak, podrazumeva se da je prekoračen kapacitet veze. Tada

se prepolovljava prozor zagušenja, prag zagušenja se postavlja na novu vrednost prozora zagušenja i ulazi se u fazu izbegavanja zagušenja.

Faza izbegavanja zagušenja se karakteriše AIMD (*Additive Increase Multiplicative Decrease*) šemom. Prozor zagušenja se posle svakog vremena obilaska efektivno povećava za konstantnu vrednost, ako se ne primeti gubitak paketa. Pri gubitku paketa veličina prozora zagušenja se deli drugom konstantom i prag zagušenja se postavlja na novu vrednost prozora zagušenja. Ove dve konstante su unapred određene. U slučaju isteka vremena brojača za ponovno slanje segmenata, ulazi se u fazu sporog starta dok prozor zagušenja ne pređe prag zagušenja. AIMD šema obezbeđuje ravnomerno deljenje kapaciteta veze kada je prisutno više tokova [16].



Slika 2.7. Kontrola zagušenja. AIMD šema.

2.1.3.7. TCP opcije

Radi unapređenja funkcionalnosti standardnog TCP-a uvedene su nove opcije sa krajnjim ciljem poboljšanja performansi. U situaciji kada se jedan segment izgubi a stigne dosta segmenata posle njega standardni TCP, zbog kumulativnog broja primljenih bajtova, privremeno zaustavlja slanje. Tade se klizni prozor ne pomera sve dok se taj izgubljeni segment ne isporuči uspešno. Ovo se radi iako veza i dalje funkcioniše i novi segmenti pristižu. Da bi se prevazišao taj problem, uvedena je opcija SACK (*Selective Acknowledgement*) [10], koja omogućava navođenje više opsega primljenih bajtova i nakon "rupe" u toku podataka. Kao i opcija FACK (*Forward Acknowledgement*) [11], koja u slučaju takvog problema omogućava povećanje kliznog prozora za broj bajtova potvrđenih opcijom SACK. Ove dve opcije pružaju mogućnost nastavljanja neprekidnog slanja čak i u slučaju da tok podataka bude sa prekidima. Poboljšanje je primetnije na brzim vezama sa većim vremenom obilaska.

Pojedini algoritmi za kontrolu zagušenja se oslanjaju na merenje varijacija vremena obilaska [7]. Kako bi se takav pristup omogućio uvedena je opcija vremenskih oznaka [12] kako bi se te varijacije mogle preciznije izmeriti.

2.1.4. Primene i osobine TCP protokola

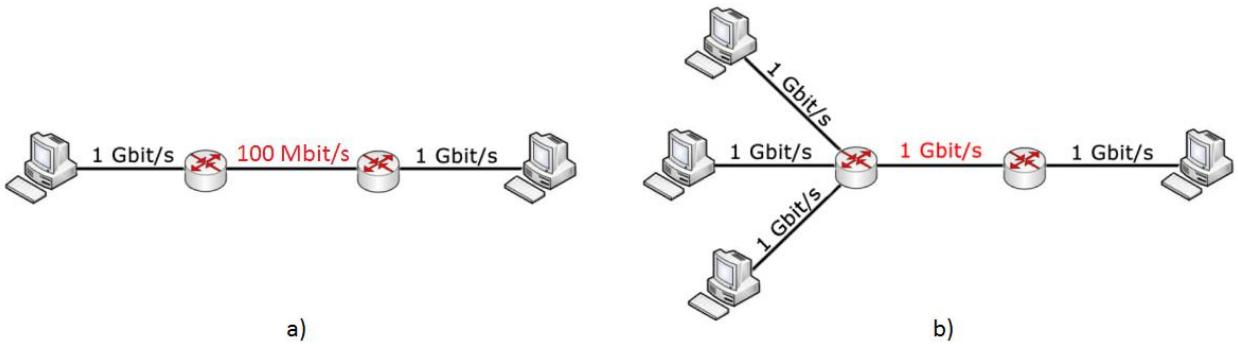
TCP je danas u širokoj primeni. Na njemu se zasniva WWW (World Wide Web), e-mail, FTP (File Transfer Protocol), SSH (Secure Shell) i mnoge druge aplikacije koje zahtevaju pouzdan prenos podataka.

TCP je optimizovan za pouzdanu dostavu podataka, brzina prenosa je u drugom planu. Tako da je u praksi moguće primetiti znatno kašnjenje dok se čeka na ponovno slanje izgubljenih paketa ili njihovo sortiranje (reda sekunda). Zato TCP nije prikladan za primene u kojima je bitna brzina (direktan prenos video zapisa, razgovori, Skype i sl.). U takvim slučajevima se koristi UDP ili neki drugi protokol zasnovan na UDP-u, među kojima je najzastupljeniji RTP (*Real-time Transport Protocol*) [19].

2.1.5. Problemi pri korišćenju TCP transfer protokola

2.1.5.1. Zagušenje veze

Zagušenje je najveći problem u mrežama sa većim brojem korisnika. Može da se javi kao posledica "uskog grla" negde na vezi ili prilikom deljenja dela veze na više korisnika. Zbog principa "sa kraja na kraj" TCP ima informacije samo o brzinama na krajevima veze, a nema nikakvih informacija o kapacitetu mreže [6]. Ako se TCP ne bi obazirao na kapacitet mreže velika količina paketa bi se gubila zbog preopterećenja i većina vremena bi se trošila na ponovno slanje izgubljenih paketa. Ovakvo stanje dovodi do kolapsa izazvanog zagušenjem.



Slika 2.8. Primer zagušenja. a) Usko grlo, b) Više korisnika

Ovaj problem drastično umanjuje performanse mreže, to je problem koji je najveći ograničavajući faktor TCP-a i pokazao se teškim za efikasno rešavanje. Kontrola zagušenja [4], [5] je oblast u koju je uloženo mnogo napora kako bi se poboljšala. Danas se koriste mnogi raznovrsni algoritmi za kontrolu zagušenja i prisutno je više pristupa ka otkrivanju zagušenja. Uobičajeni signal

zagušenja je gubitak segmenta, mada se može koristiti i porast vremena obilaska ili kombinacija ova dva pristupa [7].

2.1.5.2. Vreme obilaska

Na javnim vezama, gde je prisutno više tokova podataka, TCP ispoljava jaku zavisnost od vremena obilaska. Kao što je već pokazano, AIMD algoritam koji koristi TCP efektivno povećava prozor zagušenja posle svakog vremena obilaska. Samim tim, tokovi sa manjim vremenom obilaska brže dostižu veće brzine prenosa. Što se izražava neravnomernim raspoređivanjem dostupnog propusnog opsega različitim tokovima i preteranim favorizovanjem tokova sa manjim vremenom obilaska.

2.1.5.3. Iskorišćavanje propusnog opsega veze

Kada je nastao, TCP je bio prilagođen vezama malog kapaciteta kakve su bile prisutne u to doba. Ali danas, brzine veza kao i broj korisnika na javnim mrežama su mnogostruko veći i imaju tendenciju daljeg porasta. Čak se i osobine mrežnog saobraćaja menjaju – očekuje se od svakog korisnika da razmenjuje velike količine podataka. Na malim relacijama to ne uzrokuje probleme, ali kako se udaljenost između strana u komunikaciji povećava tako performanse drastično opadaju. Na velikim daljinama sa TCP-om postaje skoro nemoguće iskoristiti pun kapacitet mreže čak iako nema zagušenja [7], [8], [9], [16], [17].

2.2. UDP transfer protokol

2.2.1. Istorija UDP protokola

UDP omogućava da računari komuniciraju bez direktne uspostave veze. Dizajniran je 1980. od strane Davida Rida (*David Reed*). Za razmenu poruka koristi datagrame – osnovne jedinice prenosa podataka u mrežama zasnovanim na razmeni paketa gde dostava, vreme i redosled pristizanja poruka nisu garantovani mrežnom uslugom.

2.2.2. Osobine UDP protokola

UDP se bazira na malom zaglavlju i smanjenom kašnjenju a ne na proveri grešaka i potvrdi prijema. Koristi se jednostavan model sa minimumom garancija. Nema mehanizam rukovanja za uspostavu veze, pa ni za zatvaranje veze jer se veza nikad ne uspostavlja. Ne postoje nikakve garancije o redosledu podataka, njihovoj jedinstvenosti (može biti duplikata) pa čak ni da će poslate poruke biti isporučene drugoj strani. UDP obezbeđuje kontrolni zbir podataka i adrese (priključke) pošiljaoca i primaoca.

Pozicija Bajt	0								1								2								3							
Bajt	bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Izvorišni priključak																Odredišni priključak														
4	32	Dužina																Kontrolni zbir														

Slika 2.9. UDP zaglavlje.

Zaglavlje UDP segmenta sadrži sledeća polja:

- Izvorišni i odredišni priključak (16 bita) određuju priključke aplikacije na strani pošiljaoca i primaoca, respektivno. Izvorišni priključak nije obavezno koristiti, a ako se ne koristi mora se popuniti nulama.
- Dužina (16 bita) određuje veličinu UDP segmenta (zaglavlje i telo) i može biti broj između 8 i 65.535. U praktičnim primenama, to daje maksimalnu veličinu tela UDP segmenta od 65.507 bajtova (jer se rezerviše 8 bajtova za zaglavlje UDP segmenta i još 20 bajtova za zaglavlje IP paketa) uz primenu protokola IPv4. Uz protokol IPv6 telo UDP segmenta može biti i veće dužine [20].
- Kontrolni zbir se računa za zaglavlje i telo UDP segmenta, kako bi se otkrile eventualne greške. Ovo polje nije obavezno koristiti ali ako se ne koristi mora se popuniti nulama.

UDP nema stanja, već je transakcijski orijentisan. To ga čini pogodnim za jednostavne upit-odgovor protokole kao što je DNS (*Domain Name System*) kao i za istovremeno opsluživanje podacima većeg broja klijenata (*broadcasting*).

Jednostavnost UDP-a omogućava veću brzinu prenosa podataka i smanjeno vreme obilaska. Osnovna funkcionalnost pružena od strane UDP-a se po potrebi može nadograditi kako bi se obezbedile složenije primene – primer su protokoli RTP i UDT.

3. UDT transfer protokol

3.1. Istorija UDT protokola

Početkom 21. veka su jeftini optički kablovi postali popularni i time je omogućen brži saobraćaj preko Interneta. Tada su došle do izražaja ograničene performanse TCP-a. Iz tih razloga je započet UDT projekat.

UDT je razvio Janhong Gu (*Yunhong Gu*) u sklopu doktorskih studija na Univerzitetu Ilinois, SAD. Zbog velike popularnosti UDT-a, Gu je nastavio rad na UDT projektu i posle završetka studija.

Prva verzija UDT-a je nastala 2001. godine, pod nazivom SABUL (*Simple Available Bandwidth Utility Library*) i prvenstveno je korišćena za privatne mreže. SABUL je bio zasnovan na UDP-u - sam transfer podataka je išao preko UDP veze, ali su se kontrolni podaci razmenjivali preko odvojene TCP veze.

Projekat je preimenovan u UDT od svoje druge verzije, nastale 2004. godine. Kontrolni podaci su u ovoj verziji razmenjivani preko UDP veze, tako da se transfer u potpunosti odvijao preko UDP-a. Takođe, dodat je algoritam za kontrolu zagušenja kako bi se omogućile efikasne konkurentne UDT i TCP veze.

Treća verzija UDT projekta je razvijena 2006. godine i bila je prilagođenija javnim mrežama (Internetu). Kontrola zagušenja je dodatno optimizovana i za veze niskog kapaciteta i korisnicima je omogućeno lako definisanje i korišćenje novih algoritama za kontrolu zagušenja. Takođe, znatno je smanjeno korišćenje sistemskih resursa u toku rada.

UDT verzije četiri je nastao 2007. godine. Uvedena je veća podrška za konkurentne veze, omogućeno je da se više UDT veza veže za isti UDP priključak. Dodata je i mogućnost „randevu“ (*rendezvous*) uspostavljanja veza radi zaobilaženja zaštitnog zida (*firewall*).

Trenutno se koristi verzija UDT projekta 4.11, dostupna kao *open source* aplikacija [26].

U planu je i peta verzija projekta.

Sam UDT projekat je razvijen kao C++ biblioteka. Dostupni su i dodaci koji omogućuju korišćenje biblioteke u JAVA, Python i .NET programima. Težilo se tome da se omogući isti interfejs kao za TCP, kako bi se lakše razumela biblioteka i omogućilo prebacivanje na UDT sa TCP-a uz što manje napora.

UDT projekat je osvojio više nagrada na takmičenjima gde je primarni cilj bio što bolje iskorišćenje dostupnog propusnog opsega.

3.2. Osobine UDT protokola

UDT je transfer protokol koji zahteva uspostavljanje veza za funkcionisanje. UDT veze su potpuno dvosmerne – saobraćaj može istovremeno teći u oba smera. Podržava pouzdani i delimično pouzdani prenos podataka.

UDT je prvenstveno projektovan za veze velikog propusnog opsega (1 Gb/s ili više), za slučajeve gde se razmenjuju ogromne količine podataka (reda terabajt). Zamišljen je kao transfer protokol koji će se koristiti od strane načnih institucija za razmenu velikih količina podataka preko privatnih mreža. Testiranjem UDT-a se pokazalo da je moguće vrlo dobro ga koristiti i na vezama manjeg kapaciteta, za prenos manje količine podataka, pa čak i na javnim mrežama. Samim tim, UDT se približio širokoj ciljnoj publici – prosečnom korisniku Interneta.

Funkcionalnost UDT-a nadograđena na UDP se obavlja kao korisnički proces dok se recimo slične funkcionalnosti kod TCP-a obavljaju kao sistemski proces. Samim tim, UDT troši više sistemskih resursa pri radu. Ali sa druge strane, to ima i svojih prednosti. Kao korisnički proces UDT se može lako programski menjati čak i u toku rada i prilagođavati promenljivim uslovima na mreži. Takođe, više informacija o radu protokola je dostupno korisniku na uvid, što omogućava lakšu analizu rada UDT protokola.

UDT je jedini transfer protokol zasnovan na UDP-u koji ima programabilnu kontrolu zagušenja. Upravo to omogućuje primenu na javnim mrežama bez opasnosti da UDT tok zauzme najveći deo dostupnog propusnog opsega i time onemogući ostale tokove da pravilno funkcionišu. UDT pruža visok nivo tolerancije prema drugim tokovima podataka.

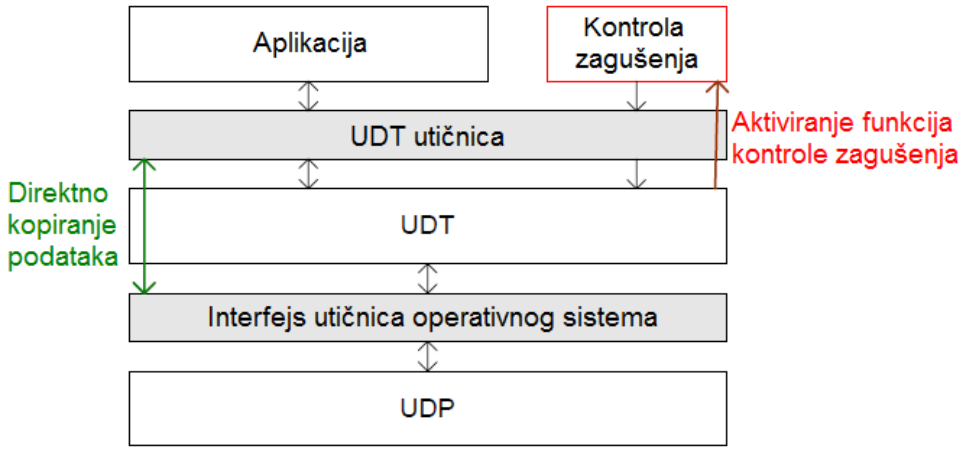
UDT se razvija sa težnjom da reši dva problema:

1) dizajn funkcionalnog i efikasnog transfer protokola
2) kontrola zagušenja na Internetu koja treba da pruži stabilnost, efikasnost i toleranciju prema drugim tokovima.

3.3. Arhitektura UDT protokola

3.3.1. UDT sloj

Formalno, UDT funkcionalnost se omogućava u sloju aplikacija TCP/IP modela. Ipak, UDT je projektovan kao biblioteka koja se može koristiti od strane drugih aplikacija, pa tako zavređuje posebno mesto. UDT se prilagodio slojevitoj arhitekturi i funkcioniše kao posrednik između aplikacije i UDP sloja. Tačnije UDT apstrahuje transfer podataka i obezbeđuje funkcionalnost pouzdanog prenosa preko nepouzdanog UDP-a koji se koristi u pozadini, van videla aplikacije.

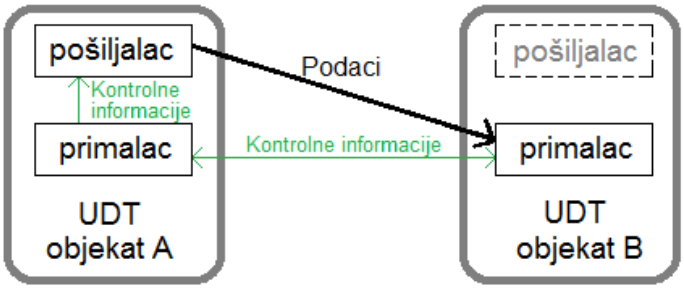


Slika 3.1. UDT sloj.

Aplikacija prenosi podatke preko UDT sloja koji u pozadini koristi UDP za slanje i primanje podataka. Izbegava se nepotrebno kopiranje, podaci se direktno kopiraju između UDT i UDP utičnica. Kontrola zagušenja se može menjati iz aplikacije, ali se aktivira na događaje iz UDT-a.

3.3.2. Komponente. Komunikacija UDT objekata

UDT, kao dvosmerni transfer protokol, ima istu arhitekturu objekata bez obzira na to da li u datom slučaju šalju ili primaju podatke. Sam transfer podataka se odvija preko komponenti UDT objekta – pošiljaoca i primaoca.



Slika 3.2. Osnovni model razmene podataka između dva UDT objekta.

Na slici 3.2. UDT objekat A preko svog pošiljaoca šalje podatke UDT objektu B. U UDT objektu B pošiljalac je neaktivan, dok primanje podataka obavlja primalac. Primaoci ovih objekata razmenjuju kontrolne informacije. Kontrolne informacije se u UDT objektu A prenose od primaoca ka pošiljaocu kako bi se vršile potrebna prilagođavanja slanja podataka.

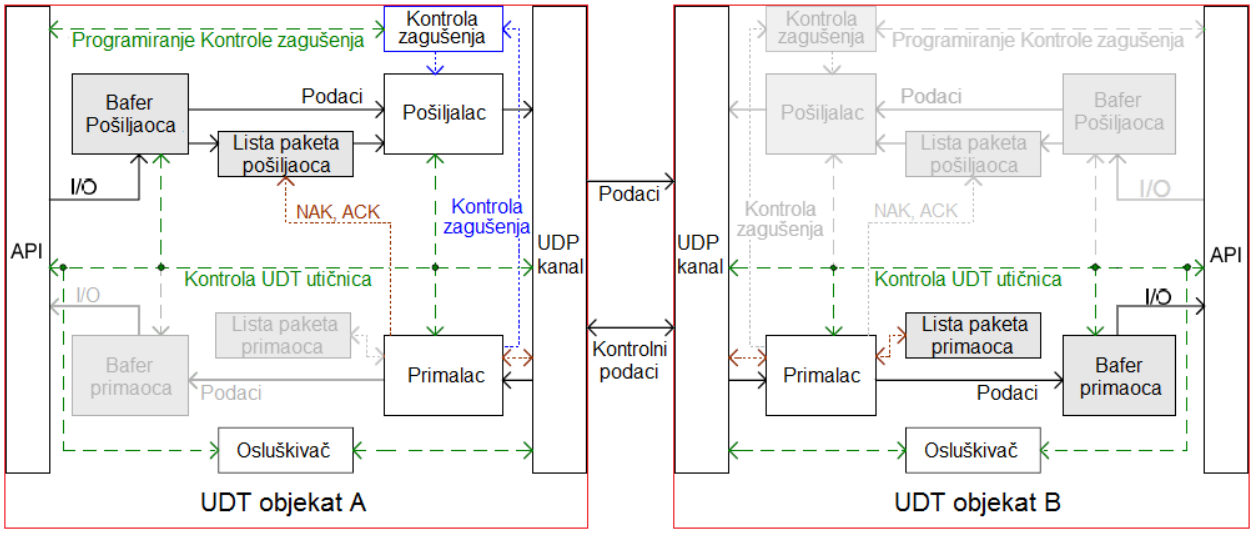
Arhitektura UDT-a se sastoji od 6 funkcionalnih komponenti:

1) aplikativni programski interfejs (API)
2) pošiljalac
3) primalac

4) UDP kanal
5) kontrola zagušenja
6) osluškivač događaja,

kao i od 4 komponente vezane za podatke:

1) bafer pošiljaoca
2) bafer primaoca
3) lista paketa pošiljaoca
4) lista paketa primaoca.



Slika 3.3. Model toka UDT komunikacije. UDT objekat A šalje podatke UDT objektu B. Funkcionalne komponente su u belim pravougaonicima. Komponente vezane za podatke su u sivim pravougaonicima. Komponente koje se ne koriste u komunikaciji su osenčene. Tok podataka je obeležen punom crnom linijom, dok su kontrolni tokovi obeleženi isprekidanim linijama.

API koji se pruža korisniku UDT-a je baziran na interfejsu TCP utičnica. Interfejs se najvećim delom oslanja na UDT utičnice koje su nosioci funkcionalnosti. Aplikacija interaguje sa UDT-om preko API-a. Korisnik pristupa UDT utičnici preko njene identifikacije i sva funkcionalnost je apstrahovana. Unutrašnja struktura UDT utičnice nije dostupna korisniku. Detaljnije o interfejsu u sekciji 4.2.

UDT objekat koji šalje podatke dobija podatke od aplikacije preko API-a. Dalje ih skladišti u bafer pošiljaoca i prosleđuje sledeći paket pošiljaocu. Takođe, unosi informacije o tom paketu u listu paketa pošiljaoca. Pošiljalac preko UDP kanala vrši prenos paketa do drugog UDT objekta. Kontrola zagušenja upravlja slanjem paketa pošiljaoca.

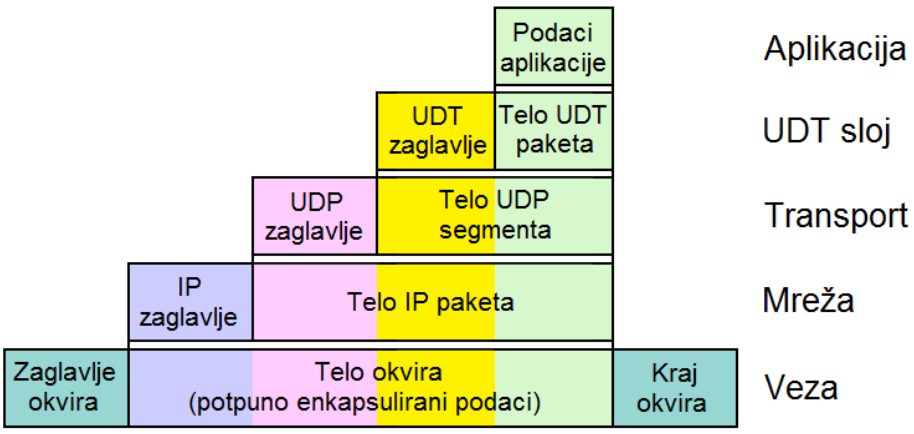
UDT objekat koji prima podatke koristi primaoca za sam prijem podataka preko UDP kanala. Primalac prosleđuje pakete baferu primaoca. U baferu primaoca se paketu uređuju i određuje se koji paketi su izgubljeni a koji primljeni. Aplikacija može u svakom trenutku preko API-a da traži podatke iz bafera primaoca, odakle se oni brišu posle prosleđivanja. Informacije o paketima, uz ostale kontrolne podatke, se šalju UDT objektu od koga su paketi primljeni. Takođe, informacije o izgubljenim paketima se dodaju u listu paketa primaoca. Informacije o paketima koji su uspešno primljeni se brišu iz liste paketa primaoca, ako su tamo bile ranije unete.

UDT objekat koji šalje podatke prima kontrolne podatke od drugog UDT objekta preko UDP kanala. Kontrolne podatke prosleđuje svom primaocu. Primalac obrađuje primljene kontrolne podatke. Na osnovu obrađenih podataka primalac može aktivirati komponentu kontrole zagušenja, mehanizme vezane za pouzdanost prenosa ili druge kontrolne mehanizme. Takođe, primalac može poslati nove kontrolne podatke UDT objektu na drugoj strani komunikacije. Izveštaj o izgubljenim paketima tj. negativni izveštaj (*Negative Acknowledgement*), se označava sa NAK. Negativni izveštaj nosi informacije o izgubljenim paketima. Potvrdni izveštaj (*Positive Acknowledgement*), u oznaci ACK, označava uspešno isporučene pakete. Po prijemu potvrdnog izveštaja se brišu informacije o isporučenim paketima iz liste paketa pošiljaoca.

Osluškivač reaguje na određene događaje na UDP kanalu i obaveštava API o njima.

3.3.3. UDT paketi

UDT koristi dve vrste paketa – pakete za prenos podataka i kontrolne pakete. UDT razlikuje tip paketa po prvom bitu njegovog zaglavlja. Zaglavlje obezbeđuje funkcionalnost nadograđenu na UDP. Pošto se UDT sloj nalazi između sloja aplikacije i UDP sloja, shodno tome se i enkapsulira.



Slika 3.4. Enkapsulacija UDT paketa.

3.3.3.1. Paketi za prenos podataka

UDT paket za prenos podataka na prvom bitu ima postavljenu nulu.

Pozicija	Bajt	0								1								2								3							
Bajt	bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	0	Redni broj paketa																														
4	32	OP		U	Redni broj poruke																												
8	64	Vremenska oznaka																															
12	96	ID odredišne UDT utičnice																															

Slika 3.5. Zaglavlje UDT paketa za prenos podataka.

UDT paket sadrži sledeća polja u svom zaglavlju:

- Redni broj UDT paketa (31 bit) se koristi radi obezbeđivanja pouzdanosti prenosa, slično kao kod TCP-a.
- Redni broj poruke (29 bita) omogućava sistem razmena poruka, u slučaju delimično pouzdanog prenosa.
- Polje OP (2 bita) je za oznaku poruke ili njenog dela ako je poruka prevelike dužine za jedan UDT paket, i može označavati:
  - Prvi paket – 10
  - Poslednji paket – 01
  - Celu poruku u paketu – 11.
- Flag U (1 bit) kontroliše uređenost poruka. Kada je flag postavljen poruka se ne može dostaviti dok se sve poruke poslate pre nje ne dostave ili odbace.
- Vremenska oznaka (32 bita) meri relativno vreme od uspostavljanja UDT veze u mikrosekundama.
- ID odredišne UDT utičnice (32 bita) određuje kojoj se UDT utičnici prosleđuje paket. Kao i u slučaju TCP-a, jedna aplikacija može koristiti više UDT utičnica istovremeno, što može doprineti boljim performansama.

3.3.3.2. Kontrolni paketi

Kontrolni UDT paket na prvom bitu ima postavljenu jedinicu.

Pozicija	Bajt	0								1								2								3							
Bajt	bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	1	Tip paketa															Prošireni tip															
4	32	Dodatne informacije																															
8	64	Vremenska oznaka																															
12	96	ID odredišne UDT utičnice																															
16	128	Kontrolne informacije																															
...	...	...																															

Slika 3.6. Zaglavlje UDT kontrolnog paketa.

Zaglavlje kontrolnog UDT paketa sadrži sledeća polja:

- Tip paketa (15 bita) sadrži informacije o tipu kontrolnog paketa. Postoji 7 osnovnih tipova kontrolnih paketa:
  1) rukovanje, prenos informacija neophodnih za uspostavljanje veze
  2) zatvaranje, označava zatvaranje veze od strane jednog UDT objekta
  3) održavanje veze, ovi paketi se povremeno stvaraju i šalju drugoj strani veze ako se ne ostvaruje druga vrsta saobraćaja u tom smeru. Ako UDT objekat ne primi nikakav paket određeno vreme, podrazumevaće da je veza prekinuta. Kontrolni paketi za održavanje veze to sprečavaju.
  4) potvrdni izveštaj (ACK)
  5) potvrda potvrdnog izveštaja (ACK2)
  6) negativni izveštaj (NAK)
  7) odbacivanje poruke, koristi se samo pri delimično pouzdanom prenosu.
- Prošireni tip (16 bita) služi za identifikovanje takvih posebnih kontrolnih paketa koje je korisniku prethodno definisao.
- Dodatne informacije (32 bita) čuvaju informacije zavisne od tipa kontrolnog paketa.
- Kontrolne informacije ne postoje za neke tipove kontrolnih paketa. Tada nisu uključene u zaglavlje. Ako tip kontrolnog paketa zahteva kontrolne informacije onda su one dužine određene za taj tip paketa.
- Vremenska oznaka i ID odredišne UDT utičnice – imaju svrhu objašnjenu u sekciji 3.3.3.1.

3.4. Funkcionalnost UDT protokola

3.4.1. Uspostavljanje veze

UDT podržava dva načina uspostavljanja veze – standardni klijent/server i randevu način.

3.4.1.1. Klijent/server uspostava veze

U klijent/server načinu uspostavljanja veze, jedan UDT objekat mora početi sa radom kao server nakon čega pasivno čeka klijenta. Klijent koji želi da uspostavi vezu mora slati redovno kontrolne pakete rukovanja dok ne dobije odgovor od servera ili istekne dozvoljeno vreme za čekanje odgovora.

Kontrolni paket rukovanja sadrži sledeće informacije:

- verziju UDT-a
- tip UDT utičnice (*SOCK_STREAM* ili *SOCK_DGRAM*)
- nasumice izabran redni broj paketa

- najveću dozvoljenu veličinu paketa
- najveću dozvoljenu veličinu kliznog prozora kontrole toka.

Server proverava UDT verziju i tip UDT utičnice po prijemu kontrolnog paketa rukovanja. Ako se te vrednosti ne poklapaju sa njegovim odbija se povezivanje. U suprotnom, server proverava veličinu paketa koju podržava klijent, upoređuje je sa svojom dozvoljenom veličinom paketa i minimum te dve vrednosti postavlja kao svoju novu vrednost. Ta nova vrednost se vraća klijentu zajedno sa serverovim nasumice izabranim rednim brojem paketa i serverovom dozvoljenom veličinom kliznog prozora. Ovi podaci se takođe prenose preko kontrolnog paketa rukovanja. Ovaj postupak se ponavlja dok god stižu novi kontrolni paketi rukovanja od klijenta. Server je sada spreman za razmenu podataka.

Klijent može da počne razmenu podataka čim primi kontrolni paket rukovanja kao odgovor od servera. Svaki sledeći kontrolni paketi rukovanja koji stignu se ignorišu.

3.4.1.2. Randevu uspostava veze

Ako su oba UDT objekta pod zaštitnim zidom (*firewall*) standardni klijent/server način uspostavljanja veze ne može funkcionisati jer zaštitni zid servera odbacuje pakete od klijenta.

UDT podržava randevu (*rendezvous*) način uspostavljanja veze koji zaobilazi zaštitni zid. U randevu načinu uspostavljanja veze ne postoje ni server ni klijent, već su korisnici ravnopravni i istovremeno pokušavaju da uspostave vezu. Korisnik koji prvi primi kontrolni paket rukovanja šalje drugom odgovor. Dalje se veza uspostavlja slično načinu opisanom u sekciji 3.4.1.1.

3.4.2. Zatvaranje veze

Ako jedan UDT objekat završi sa razmenom podataka i poželi da zatvori vezu, poslaće kontrolni paket zatvaranja drugoj strani. Nakon toga prekida sa slanjem podataka i kontrolnih paketa održavanja veze.

Kontrolni paket zatvaranja se šalje samo jednom i ne garantuje se njegovo isporučivanje.

Ako druga strana primi kontrolni paket zatvaranja, takođe će, sa svoje strane, zatvoriti vezu. U suprotnom, veza će se prekinuti posle isteka određenog vremena pošto je prekinut dotok podataka i kontrolnih paketa održavanja veze.

3.4.3. Pouzdanost prenosa

Pouzdani prenos podataka nije primarni cilj svih aplikacija. Postoje aplikacije koje više računa vode o brzini prenosa nego o njegovoj pouzdanosti.

UDT može da pruža različite garancije što se tiče pouzdanosti prenosa. Tako UDT obezbeđuje funkcionalnost pouzdanog prenosa podataka i delimično pouzdanog prenosa podataka.

Dovoljno je pri stvaranju UDT utičnice navesti tip utičnice kao parametar.

Tip utičnice može biti:

- SOCK_STREAM, za pouzdan prenos podataka
- SOCK_DGRAM, za delimično pouzdan prenos podataka.

3.4.3.1. Pouzdani prenos

UDT koristi izveštaje o paketima kako bi obezbedio pouzdanost prenosa i kontrolu zagušenja, slično kao i TCP. UDT koristi potvrdne izveštaje i negativne izveštaje, koji se proizvode na strani UDT objekta koji prima podatke, kao i potvrde o potvrdnim izveštajima koje se proizvode na strani UDT objekta koji šalje podatke.

Potvrdni izveštaji (ACK) se proizvode na određeno vreme ako su podaci primljeni neprekidno. ACK paket nosi informaciju o najvećem rednom broju paketa, od onih koji su primljeni uređeno i bez gubitaka. Tako jedan ACK paket može da potvrdi prijem čitavog niza paketa odjednom. Ovo znači da što je veća brzina prenosa podataka to se manji procenat propusnog opsega troši na kontrolne podatke. Takođe, uz manje kontrolnih paketa, manje je trošenje sistemskih resursa na njihovu obradu. Ovo je značajna prednost u odnosu na TCP, koji proizvodi ACK posle svakog primljenog paketa.

Podrazumevani interval proizvodnje ACK-a je 10ms, pa se na vezi kapaciteta 1Gb/s stvara 1 ACK na svakih 833 paketa veličine 1500 bajtova (veličina Eternet okvira).

UDT numeriše ACK pakete, kako bi se izbeglo višestruko slanje. Svaki ACK ima svoj redni broj (31 bit) koji je nezavisan od rednih brojeva paketa za prenos podataka.

Potvrda potvrdnog izveštaja (ACK2) se može stvarati po prijemu svakog ACK paketa i slati kao odgovor drugoj strani. Prijem ACK2 paketa označava isporuku odgovarajućeg ACK paketa, koji se nakon toga ne mora više slati. Takođe, na osnovu rednog broja ACK paketa, može se računati vreme obilaska veze (RTT). Vreme obilaska se dobija oduzimanjem vremena slanja ACK paketa od vremena pristizanja odgovarajućeg ACK2 paketa.

Negativni izveštaj (NAK) se proizvodi po otkrivanju gubitka paketa i odmah se šalje UDT objektu koji šalje podatke. Na taj način se izgubljeni paketi šalju opet i ubrzava se reagovanje na potencijalno zagušenje veze.

NAK nosi informacije o izgubljenim paketima, tačnije, njihove redne brojeve. Ako paketi označeni u NAK-u ne budu isporučeni posle određenog vremena, NAK se šalje opet. Ovaj postupak se može ponavljati više puta, ali posle svakog slanja NAK-a se povećava vreme čekanja pre narednog ponovnog slanja.

Obaveštavanjem o izgubljenim paketima UDT obezbeđuje mehanizam sličan TCP opciji SACK. Ali, NAK može sadržati više informacija od polja TCP opcije SACK pa tako obezbeđuje još bolje reagovanje na zagušenje.

3.4.3.2. Delimično pouzdani prenos

Delimično pouzdan prenos se odlikuje visokim performansama.

Zasniva se na razmeni poruka, a ne paketa.

Svakoj poruci se dodeljuje vremenska oznaka, vreme života i flag uređenosti.

Ako vreme života poruke istekne pre nego što se poruka pošalje, poruka se uklanja iz reda za slanje. Ako je poruka već ranije poslata ali nije potvrđen njen prijem, šalje se i kontrolni paket za odbacivanje poruke. Kada druga strana primi kontrolni paket za odbacivanje poruke svi delovi te poruke se odbacuju.

Vreme života može biti i negativna vrednost i u tom slučaju se garantuje pouzdan prenos poruke.

3.4.4. Kontrola toka

UDT koristi kontrolu toka zasnovanu na korišćenju prozora za ograničavanje broja nepotvrđenih paketa koji mogu da se nađu na vezi. Osnovna primena kontrole toka je da ograniči slanje paketa prilikom zagušenja kako se veza ne bi dodatno opteretila.

Prozor kontrole toka se dinamički računa pri prijemu svakog ACK paketa, po jednačini:

$$PKT = BP * (RTT + IKZ) \tag{3.1}$$

PKT je veličina prozora kontrole zagušenja u paketima. *RTT* je procenjeno vreme obilaska. *IKZ* je interval kontrole zagušenja – vreme posle kojeg se proizvode ACK paketi. *BP* je brzina pristizanja paketa – UDT objekat koji prima podatke beleži intervale pristizanja paketa nakon slanja prethodnog ACK paketa, uzima medijanu *M* tih intervala i računa brzinu pristizanja:

$$BP = 1/M \tag{3.2}$$

Koristi se medijana intervala jer se mogu javiti prekidi u slanju paketa, pa pojedini intervali između pristizanja paketa mogu biti preveliki.

Tada se računa veličina prozora kontrole toka u paketima, po jednačini (3.1). Dalje se računa minimum između veličine prozora kontrole toka u bajtovima i slobodnog prostora u baferu primaoca. Taj minimum se šalje UDT objektu koji šalje podatke preko novog ACK paketa kako bi prilagodio veličinu svog prozora kontrole toka.

3.4.5. Kontrola zagušenja

UDT koristi kontrolu zagušenja zasnovanu na podešavanju brzine slanja paketa. Paketi se šalju jedan po jedan, na određeni vremenski interval. Taj interval slanja paketa je promenljiv i njime upravlja kontrola zagušenja.

Kontrola zagušenja se pokreće na određeni konstantni vremenski interval. Na taj način se uklanja zavisnost toka od vremena obilaska veze i povećava se tolerancija prema tokovima sa različitim vremenima obilaska.

3.4.5.1. Algoritam kontrole zagušenja

UDT koristi algoritam kontrole zagušenja iz klase algoritama DAIMD (*AIMD with Decreasing Increases*) koji garantuje stabilnost i toleranciju prema drugim tokovima na mreži [24].

U slučaju da tokom poslednjeg intervala pokretanja kontrole zagušenja nije bilo gubitaka, već samo potvrdnih izveštaja, brzina slanja paketa *v* se povećava za funkciju *α(v)*:

$$v' = v + \alpha(v) \tag{3.3}$$

gde je *α(v)* opadajuća funkcija, i važi $\lim_{n\to\infty} \alpha(v) = 0$. Takođe je poželjno da *α(v)* bude velika u okolini tačke *v* = 0, i da *α(v)* brzo opada kako bi se smanjile oscilacije performansi.

U slučaju da se tokom poslednjeg intervala pokretanja kontrole zagušenja javio gubitak tj. primljen je makar jedan negativni izveštaj, brzina slanja paketa *v* se menja na sledeći način:

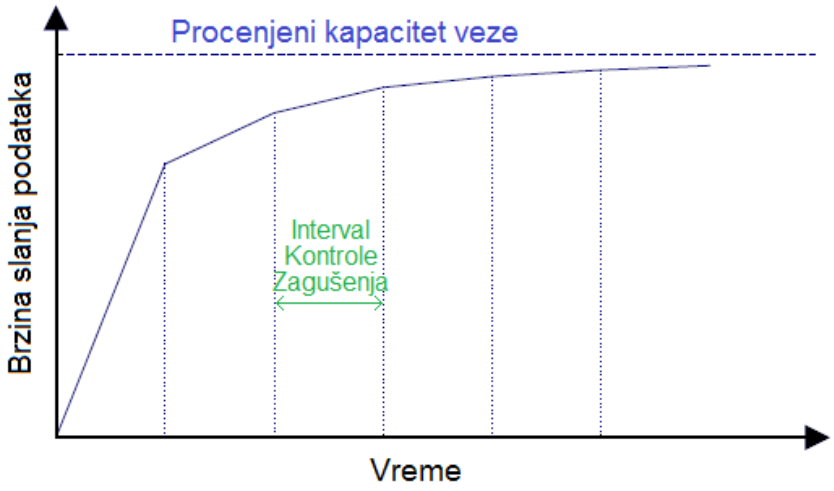
$$v' = \beta v \tag{3.4}$$

gde je β broj, $0 < \beta < 1$.

Kod UDT-a, funkcija α se računa na sledeći način:

$$\alpha(v) = 10^{\lceil \log_{10}(K - B(v)) - 9 \rceil} \frac{1500}{P} \frac{1}{IKZ} \tag{3.5}$$

K je procenjeni kapacitet veze u bitovima po sekundi. *B(v)* je brzina slanja u bitovima po sekundi. *P* je veličina paketa, činilac *1500/P* se koristi kako povećanje ne bi zavisilo od veličina paketa, podrazumevana veličina paketa u UDT-u je 1500. *IKZ* je interval kontrole zagušenja.



Slika 3.7. Prikaz porasta brzine slanja podataka u UDT-u.

Primer povećanja brzine slanja paketa je prikazan u tabeli 3.1. koristeći oznake iz jednačine 3.5. Za D manje od 0.1 se koristi minimalna moguća vrednost povećanja. Za veće vrednosti - ako se D poveća do sledećeg stepena broja 10, brzina se povećava 10 puta.

Tabela 3.1. Povećanje brzine slanja paketa u zavisnosti od procenjenog dostupnog propusnog opsega D.

Procena dostupnog propusnog opsega D = K – B(v), mereno u Mb/s	Povećanje brzine, mereno u Mb/s
$D \leq 0.1$	0.0008
$0.1 < D \leq 1$	0.0012
$1 < D \leq 10$	0.012
$10 < D \leq 100$	0.12

Radi objašnjenja reagovanja na gubitke, uvode se sledeći pojmovi:

- gubitak paketa usled zagušenja, ako je izgubljen paket poslat posle prethodnog smanjivanja brzine. Takav slučaj se otkriva upoređivanjem najvećeg rednog broja paketa poslatih neposredno pre prethodnog smanjivanja brzine sa rednim brojevima paketa u novom negativnom izveštaju.
- gubitak paketa, ako je primljen negativan izveštaj i ustanovljeno je da se nije desio gubitak paketa usled zagušenja.
- epoha zagušenja, vremenski interval između dva uzastopna gubitka paketa usled zagušenja.

Reagovanje na gubitke se realizuje po prijemu negativnog izveštaja i razlikujemo dva slučaja:

1) Ako se desio gubitak paketa usled zagušenja, to označava početak nove epohe zagušenja.

    a. Ako je izgubljen samo jedan paket brzina slanja paketa se ne umanjuje. Ovo poboljšava performanse u slučajevima kada veza ima „šum“ tj. gube se paketi nevezano za zagušenje na vezi.

    b. Ako je izgubljeno više paketa β se postavlja na 8/9.

2) Ako se desio gubitak paketa, nastavlja se tekuća epoha zagušenja i broj β se bira nasumice, tako da bude u intervalu:

$$\beta \in \left(\frac{8}{9}, \sqrt[N]{\frac{8}{9}}\right) \tag{3.6}$$

gde je N broj gubitka paketa u prethodnoj epohi zagušenja. Ovim nasumičnim biranjem broja β se uklanja mogućnost usklađivanja tokova – pojava da više UDT tokova deluju u harmoniji. Usklađenim tokovima brzine slanja podataka istovremeno rastu i opadaju što umanjuje prosečnu iskorišćenost propusnog opsega veze.

Opisani algoritam kontrole zagušenja za UDT je implementiran u klasi CUDTCC.

3.4.5.2. Procena propusnog opsega

UDT koristi vezane parove paketa da proceni kapacitet veze. Paketi koji su vezani se šalju jedan za drugim, bez pauze. UDT objekat koji šalje podatke šalje vezani par paketa posle svakih 16 paketa za prenos podataka. UDT objekat koji prima podatke beleži vremena između uzastopnih prijema vezanih parova paketa i pomoću toga računa procenu trenutnog kapaciteta veze. Procenjeni kapacitet veze se prosleđuje drugoj strani preko ACK paketa.

Ovim pristupom, tokovi sa većim brzinama slanja dobijaju manja povećanja brzine i obratno, što omogućava da više tokova ravnopravno podeli kapacitet mreže. Ova ravnopravnost ne zavisi od vremena obilaska, zbog konstantnog intervala kontrole zagušenja kada se brzina prilagođava.

Na javnim mrežama, zbog nepredviđenih okolnosti, može doći do greške u proceni kapaciteta veze. Tako prisustvo pozadinskog saobraćaja može umanjiti procenu kapaciteta [25]. Neke druge okolnosti mogu povećati tu procenu. Greška u proceni kapaciteta veze se može umanjiti korišćenjem proseka ili medijane vremena pristizanja više vezanih parova paketa. Takođe, u jednačini 3.5. se koristi vrednost logaritma zaokružena na gore kako bi se smanjio uticaj eventualne greške u proceni dostupnog propusnog opsega.

3.4.6. Programabilna kontrola zagušenja

UDT pruža mogućnost definisanja novih algoritama kontrole zagušenja. Novi algoritmi se mogu smenjivati u toku radu, zajedno sa podrazumevanim UDT algoritmom, opisanim u sekciji 3.4.5. Time je omogućena:

• Implementacija i korišćenje specifičnih algoritama kontrole zagušenja

Postoje algoritmi koji se mogu koristiti na privatnim mrežama ali nisu prikladni za javno korišćenje zbog male tolerancije prema drugim tokovima. UDT omogućava korišćenje ovakvih algoritama bez njihovog ugrađivanja u jezgro operativnog sistema, što bi načelno trebalo izbegavati.

• Dinamička prilagodljivost aplikacije mrežnim uslovima

U toku rada se mogu smenjivati razni algoritmi kontrole zagušenja zavisno od vremena korišćenja, korisnika, trenutnih uslova na mreži i sl.

• Testiranje performansi novih algoritama zagušenja

Da bi se neki algoritam ugradio u jezgro operativnog sistema mora pre toga biti temeljno testiran. Mnogo je lakše testirati algoritme preko UDT-a nego menjanjem jezgra operativnog sistema.

Da bi se omogućila programabilna kontrola zagušenja, koriste se četiri kategorije podešavanja:

1) Akcije kontrole zagušenja
2) Metode podešavanja protokola
3) Dodatni tipovi paketa
4) Praćenje performansi.

UDT, kao objektno orijentisan, definiše algoritme kontrole zagušenja preko klasa. Obezbeđena je osnovna klasa kontrole zagušenja (CCC) koja sadrži definisane sve potrebne metode, akcije i promenljive. Novi algoritam može naslediti ovu klasu i da predefiniše potrebno.

Implementacija novog algoritma kontrole zagušenja mora u toku rada menjati makar jedan kontrolni parametar. Kontrolni parametri definisani u klasi CCC su:

- prozor zagušenja, ako je algoritam zasnovan na korišćenju prozora
- interval slanja paketa, ako je algoritam zasnovan na prilagođavanju brzine slanja paketa.

UDT takođe ima ugrađene algoritme kontrole zagušenja, preko klasa:

- CUDPBlast, gde je brzina slanja određena parametrom koji se može menjati u toku rada ali samo od strane korisnika. Uslovi na mreži ne utiču na ovaj algoritam.
- CTCP, algoritam koji ima funkcionalnost standardnog TCP algoritma kontrole zagušenja. Ostali TCP algoritmi se mogu implementirati nasleđivanjem ove klase – u praksi se pokazalo da je za to potrebno malo truda (manje od 100 linija novog koda).

3.4.6.1. Akcije kontrole zagušenja

Postoji 7 osnovnih akcija kontrole zagušenja definisanih u klasi CCC. UDT aktivira ove akcije kada se desi neki kontrolni događaj.

1. ***init,*** aktivira se po uspostavljanju veze. Alocira potrebnu memoriju i postavlja početne vrednosti struktura podataka.
2. ***close***, aktivira se po zatvaranju veze. Oslobađa memoriju korišćenih struktura podataka.
3. ***onACK***, ova akcija se aktivira po prijemu ACK paketa na strani UDT objekta koji šalje podatke. Dostupna je informacija o potvrđenim paketima.
4. ***onLoss***, akcija se vrši po otkrivanju gubitka paketa. Akciji su dostupne informacije o izgubljenim paketima.
5. ***onTimeout,*** isticanje vremena određenog za čekanje pre ponovnog slanja paketa aktivira ovu akciju. UDT obezbeđuje dinamičko računanje tog vremena na način opisan u [13], ali korisnik može računati to vreme na drugi način.
6. ***onPktSent,*** aktivira se neposredno pre slanja paketa. Dostupne informacije su redni broj paketa, njegova vremenska oznaka, veličina itd.
7. ***onPktReceived***, aktivira se neposredno po prijemu paketa. Slično akciji *onPktSent*, korisniku su dostupne informacije o paketu.

Akcije *onPktSent* i *onPktReceived* su dve najmoćnije akcije jer omogućavaju proveravanja slanja odnosno prijema svakog paketa i time je moguće detaljno ispitati ponašanje algoritma.

3.4.6.2. Metode podešavanja protokola

Pojedini algoritmi za kontrolu zagušenja zahtevaju drugačije ponašanje od podrazumevanog u UDT-u. Zato su obezbeđene metode koje prilagođavaju ponašanje UDT-a:

- ***setACKTimer***, određuje interval slanja potvrdnih izveštaja.
- ***setACKInterval***, određuje posle koliko primljenih paketa za prenos podataka se proizvodi potvrdni izveštaj.
- ***sendCustomMsg***, šalje kontrolni paket koji je definisao korisnik drugoj strani.
- ***processCustomMsg,*** ovaj metod obrađuje kontrolne pakete definisane od strane korisnika, isporučene metodom *sendCustomMsg*.

Takođe, moguće je definisati metode za računanje prosečnog vremena obilaska ili vremena čekanja pre ponovnog slanja paketa. Na taj način korisnik može da računa te vrednosti na drugačiji način od podrazumevanog načina podržanog u UDT-u.

3.4.6.3. Dodatni tipovi paketa

Korisnicima je pružena mogućnost definisanja novih tipova kontrolnih paketa, koristeći polje "prošireni tip", prikazano na slici 3. 6.

Novodefinisani paketi se šalju metodom *sendCustomMsg*, a kontrolne informacije koje prenose ovi paketi se obrađuju metodom *processCustomMsg*. Ova dva metoda se u tom slučaju moraju predefinisati.

3.4.6.4. Praćenje performansi

Informacije o performansama obezbeđuju prilagodljivost algoritma dinamičkim mrežnim uslovima. Tako se može dobiti i statistika tog algoritma koja omogućava ocenjivanje algoritma i njegovo dalje podešavanje od strane korisnika.

Dostupne informacije uključuju: vreme trajanja veze, vreme obilaska, brzinu slanja paketa, brzinu primanja paketa, broj izgubljenih paketa, broj potvrdnih izveštaja, broj negativnih izveštaja, interval slanja paketa, veličinu prozora zagušenja i veličinu prozora toka. UDT beleži ove vrednosti pri svakoj njihovoj promeni.

Mogu se pružiti informacije o performansama u tri kategorije:

- Zbirne ili prosečne vrednosti od nastanka UDT utičnice
- Zbirne ili prosečne vrednosti od trenutka prethodnog merenja performansi
- Vrednosti u trenutku merenja.

4. Uporedno testiranje performansi

4.1. Metodologija testiranja

Okruženje u kojem je vršeno testiranje se sastojalo od dva računara. Jedan je bio u Beogradu (Srbija) dok je drugi bio *CLOUD* server u Oregonu (SAD). Veza ova dva računara je u proseku imala vreme obilaska od 185 milisekundi, uz raspoloživi propusni opseg od 100Mb/s.

Računar u Beogradu je pod operativnim sistemom *Linux Ubuntu 12.04* (*kernel v3.2*), sa *Intel Pentium IV @1.8GHz* procesorom i 2GB *RAM*-a, dok je *CLOUD* server pod *Linux Red Hat 6.3* (*kernel v2.6.3*) operativnim sistemom, sa *Intel Xeon @2.0GHz* procesorom i 4GB *RAM*-a.

U ovom okruženju su vršena dva tipa testiranja:

- sa jednim tokom prenosa podataka, vršen 500 puta
- sa 8 istovremenih tokova prenosa podataka, ponavljan 400 puta.

Svaki test se sastojao u prenosu 2GB podataka preko veze, nakon čega je merena prosečna ostvarena brzina protoka. Testovi su izvođeni u različita doba dana, kako bi se što verodostojnije prikazalo ponašanje protokola za prenos podataka preko Interneta, krajnje nepredvidive javne mreže. Na kraju, prikazane su distribucije ostvarenih prosečnih brzina prenosa podataka.

Za testiranje TCP-a je korišćen *iperf* [27] – specijalizovana aplikacija za merenje brzina prenosa podataka, uz korišćenje *TCP Illinois* [7] algoritma za kontrolu zagušenja. *TCP Illinois* je jedan od najnovijih algoritama u svojoj klasi, i pokazao je mnogostruko bolje rezultate od standardnog TCP algoritma za kontrolu zagušenja [7], [16].

Za UDT testiranje je napravljena nova aplikacija posebno za tu svrhu, detaljnije objašnjena u sekciji 4.3.

4.2. Aplikativni programski interfejs za rad u programskom jeziku C

API je veoma važan za implementiranje protokola za prenos podataka. Interfejs UDT utičnica je projektovan tako da bude sličan interfejsu TCP utičnica. Ovako korisnici mogu olakšano da savladaju korišćenje UDT-a kao i da vrše prebacivanje postojećih TCP aplikacija na UDT.

U cilju pravljenja što efikasnije UDT aplikacije je napravljen poseban dodatak kako bi se UDT biblioteka mogla koristiti u programskom jeziku C. Standardne funkcije za rad nad TCP utičnicama su implementirane i za UDT utičnice, uz isti potpis funkcije. Ime funkcije se razlikuje – UDT funkcije imaju prefiks „udt_“. Zbog specifičnosti UDT-a su neke funkcije morale biti izmenjene, ali minimalno. Takođe, UDT pruža veću funkcionalnost od TCP-a, pa tako postoje i dodatne funkcije koje nisu podržane u interfejsu TCP utičnica.

Funkcije dodatka za korišćenje UDT biblioteke u programskom jeziku C su implementirane tako da u slučaju uspeha fukcija ima povratnu vrednost 0. A ako se javi greška u izvršavanju funkcije povratna vrednost će biti negativni kôd greške. Ovo važi tamo gde nije drugačije naglašeno.

Funkcije i korišćene strukture su opisane po kategorijama u sledećim sekcijama 4.2.1., 4.2.2., 4.2.3., 4.2.4. Strukture su objašnjene u tabelama, dok su funkcije opisane u formatu: povratna_vrednost **ime_funkcije** (tip_argumenta1 ime_argumenta1, ...) : Opis_funkcije

- *ime_argumenta1* - Opis_argumenta1
- ...

4.2.1. Rad sa UDT bibliotekom

UDT biblioteka zauzima memoriju aplikacije pri korišćenju. Omogućeno je da se UDT biblioteka može pokretati i zaustavljati po potrebi. Na taj način se memorija aplikacije zauzima samo kada je to neophodno.

int **udt_startup** (void) : Vrši potrebnu inicijalizaciju UDT biblioteke.

int **udt_cleanup** (void) : Otpušta memoriju zauzetu UDT bibliotekom. Zatvara sve UDT veze.

4.2.2. Obrada grešaka

Za potrebe obrade grešaka je implementirana funkcija kojom dobijamo opis greške na osnovu njenog kôda.

char* **udt_error_message**(int udt_error_code) : Vraća odgovarajući tekstualni opis greške.

- *udt_error_code* – kôd greške za koju tražimo opis.

Tabela 4.1. Greške koje se mogu javiti pri pozivu UDT funkcija. Po kolonama: naziv greške, kôd greške i kratak opis.

Naziv	Kôd	Opis
UDT_SUCCESS	0	Uspešno izvršena operacija
UDT_ECONNSETUP	1000	Greška pri uspostavljanju veze
UDT_ENOSERVER	1001	Server ne postoji
UDT_ECONNREJ	1002	Zahtev za uspostavljanjem veze je odbijen
UDT_ESOCKFAIL	1003	Ne može se napraviti/izmeniti UDP utičnica
UDT_ESECFAIL	1004	Veza se ne može uspostaviti iz sigurnosnih razloga
UDT_ECONNFAIL	2000	Greška na vezi
UDT_ECONNLOST	2001	Veza je prekinuta
UDT_ENOCONN	2002	Veza nije uspostavljena
UDT_ERESOURCE	3000	Sistemska greška
UDT_ETHREAD	3001	Ne može se napraviti nova nit
UDT_ENOBUF	3002	Nema dovoljno memorije
UDT_EFILE	4000	Ne može se pristupiti datoteci

UDT_EINVRDOFF	4001	Pogrešna pozicija za čitanje
UDT_ERDPERM	4002	Nema dozvole za čitanje
UDT_EINVWROFF	4003	Pogrešna pozicija pisanja
UDT_EWRPERM	4004	Nema dozvole za pisanje
UDT_EINVOP	5000	Operacija nije podržana
UDT_EBOUNDSOCK	5001	Ne može se izvršiti operacija na vezanoj utičnici
UDT_ECONNSOCK	5002	Ne može se izvršiti operacija na utičnici sa uspostavljenom vezom
UDT_EINVPARAM	5003	Pogrešan parametar
UDT_EINVSOCK	5004	UDT utičnica nije ispravna
UDT_EUNBOUNDSOCK	5005	Utičnica nije vezana za lokalnu adresu. Nemoguće osluškivanje
UDT_ENOLISTEN	5006	Ne može se uspostaviti veza. Utičnica ne osluškuje dolazeće veze
UDT_ERDVNOSERV	5007	Randevu uspostavljanje veze ne podržava osluškivanje utičnice
UDT_ERDVUNBOUND	5008	Utičnica nije vezana pre pokušaja randevu uspostavljanja veza
UDT_ESTREAMILL	5009	Operacija nije podržana za SOCK_STREAM tip UDT utičnice
UDT_EDGRAMILL	5010	Operacija nije podržana za SOCK_DGRAM tip UDT utičnice
UDT_EDUPLISTEN	5011	Druga UDT utičnica već osluškuje isti UDP priključak
UDT_ELARGEMSG	5012	Poruka je prevelika za bafer pošiljaoca
UDT_EASYNCFAIL	6000	Neuspeh pri neblokirajućem pozivu
UDT_EASYNCSND	6001	Neblokirajuće slanje nije moguće. Nema dovoljno prostora u baferu
UDT_EASYNCRCV	6002	Neblokirajuće čitanje nije moguće. Nema podataka
UDT_ETIMEOUT	6003	Isteklo je vreme čekanja pre završetka operacije
UDT_EPEERERR	7000	Druga strana je doživela grešku

4.2.3. Interfejs UDT utičnica

Implementacija funkcija za rad sa UDT utičnicama. Ove funkcije su analogne funkcijama za rad sa TCP utičnicama.

UDTSOCKET **udt_socket** (int af, int type, int protocol) : Pravi novu UDT utičnicu, čiji deskriptor vraća kao povratnu vrednost u slučaju uspeha.

- *af* – IP familija: AF_INET ili AF_INET6
- *type* – Tip UDT utičnice: SOCK_STREAM ili SOCK_DGRAM
- *protocol* – Dodato zbog saglasnosti sa TCP interfejsom.

int **udt_bind** (UDTSOCKET u, struct sockaddr *name, int namelen) : Vezuje UDT utičnicu za lokalnu adresu.

- *u* – deskriptor UDT utičnice koju vezujemo
- *name* – struktura koja predstavlja lokalnu adresu
- *namelen* – veličina *name* strukture.

int **udt_listen** (UDTSOCKET u, int backlog) : Prebacuje UDT server u stanje osluškivanja nadolazećih veza.

- *u* – Deskriptor UDT utičnice servera
- *backlog* – Najveći broj veza koje mogu istovremeno čekati na uspostavljanje.

UDTSOCKET **udt_accept** (UDTSOCKET u, struct sockaddr *addr, int *addrlen) : Vrši pasivno povezivanje. Prihvata novu vezu iz reda čekanja UDT servera koji je u stanju osluškivanja. Vraća UDT deskriptor nove veze u slučaju uspeha.

- *u* – Deskriptor UDT utičnice servera
- *addr* – Adresa druge strane novouspostavljene veze
- *addrlen* – veličina strukture *addr*.

int **udt_connect** (UDTSOCKET u, const struct sockaddr *name, int namelen) : Vrši aktivno povezivanje.

- *u* – Deskriptor UDT utičnice koju povezujemo
- *name* – Adresa servera sa kojim se povezujemo
- *namelen* – Veličina strukture *name*.

int **udt_close** (UDTSOCKET u) : Zatvara UDT utičnicu i vezu te utičnice, ako postoji.

- *u* – Deskriptor UDT utičnice koju zatvaramo.

int **udt_send** (UDTSOCKET u, const char *buf, int len, int flags),
int **udt_recv** (UDTSOCKET u, char *buf, int len, int flags) : Šalje/Prima podatke preko uspostavljene UDT veze određene deskriptorom UDT utičnice. Vraća broj poslatih/primljenih bajtova u slučaju uspeha.

- *u* – Deskriptor UDT utičnice
- *buf* – Bafer za čitanje/čuvanje podataka za slanje/prijem
- *len* – Veličina bafera *buf*
- *flags* – Dodato zbog saglasnosti sa TCP interfejsom.

4.2.4. Opcije UDT utičnica

Opcije UDT utičnice se dele između UDT objekata pri uspostavljanju veze između njih. UDT pruža širok spektar podešavanja UDT utičnica. UDT opcije su prikazane u tabeli 4.2.

int **udt_getsockopt** (UDTSOCKET u, int level, UDT_SOCKOPT oname, char *oval, int *olen),
int **udt_setsockopt** (UDTSOCKET u, int level, UDT_SOCKOPT oname, char *oval, int olen) : Čita/Postavlja opcije UDT utičnice.

- *u* – Deskriptor UDT utičnice
- *level* - Dodato zbog saglasnosti sa TCP interfejsom

- *oname* – Ime UDT opcije
- *oval* – Vrednost UDT opcije koju čitamo/postavljamo
- *olen* – Veličina tipa UDT opcije *oname*.

Tabela 4.2. Podržane UDT opcije. Po kolonama: naziv UDT opcije, kratak opis i podrazumevana vrednost.

Naziv	Opis	Podrazumevano
UDT_MSS	Najveća veličina jedinice prenosa (B)	1500B
UDT_SNDSYN	Slanje sa ili bez blokiranja	Sa blokiranjem
UDT_RCVSYN	Prijem sa ili bez blokiranja	Sa blokiranjem
UDT_CC	Algoritam kontrole zagušenja	CUDTCC
UDT_FC	Najveća veličina prozora kontrole toka (B)	25600B
UDT_SNDBUF	Veličina bafera za UDT pošiljaoca (B)	10MB
UDT_RCVBUF	Veličina bafera za UDT primaoca (B)	10MB
UDP_SNDBUF	Veličina bafera za slanje UDP utičnice (B)	1MB
UDP_RCVBUF	Veličina bafera za prijem UDP utičnice (B)	1MB
UDT_LINGER	Vreme održavanja veze po zatvaranju utičnice (s)	180s
UDT_RENDEZVOUS	Randevu uspostavljanje veze	Isključeno
UDT_SNDTIMEO	Vreme čekanja slanja (ms)	-1 (neograničeno)
UDT_RCVTIMEO	Vreme čekanja prijema (ms)	-1 (neograničeno)
UDT_REUSEADDR	Korišćenje postojeće lokalne adrese za utičnicu	Uključeno
UDT_MAXBW	Gornja granica korišćenja propusnog opsega (B/s)	-1 (neograničeno)
UDT_STATE	Trenutno stanje UDT utičnice (samo za čitanje)	-
UDT_EVENT	Dostupni događaji za epol čekanje (samo za čitanje)	-
UDT_SNDDATA	Količina podataka u baferu pošiljaoca (samo za čitanje)	-
UDT_RCVDATA	Količina podataka u baferu primaoca (samo za čitanje)	-

4.2.5. Praćenje performansi UDT protokola

Činjenica da UDT utičnica nije sistemska, već se nalazi u korisničkom delu procesa, omogućava lak pristup UDT utičnici preko interfejsa aplikacije. Tako je moguće praćenje performansi UDT prenosa podataka u toku rada aplikacije na vrlo jednostavan način. Informacije navedene u tabeli 4.3. su dostupne korisniku u svakom trenutku. Time je omogućeno lako i brzo kontrolisanje i ocenjivanje različitih aspekata protokola, npr. algoritma kontrole zagušenja ili čak prilagođavanje UDT utičnice trenutnim uslovima na mreži izmenom UDT opcija, opisanim u sekciji 4.2.4.

int **udt_perfmon**(UDTSOCKET u, UDT_TRACEINFO * perf, int clear) : Prati performanse UDT veze određene deskriptorom UDT utičnice.

- *u* – Deskriptor UDT utičnice
- *perf* – Struktura za skladištenje informacija o performansama

- *clear* – Određuje da li se pojedine informacije mere iznova pri svakom merenju performansi ili ne.

Tabela 4.3. Informacije koje se mogu dobiti merenjem performansi. Po kolonama: naziv polja, tip vrednosti polja, kratak opis i kategorija. Kategorija može biti: 1) vrednosti merene od nastanka UDT utičnice, 2) vrednosti izmerene od prethodnog merenja performansi, 3) vrednosti u trenutku merenja.

Naziv	Tip	Opis	K.
msTimeStamp	int64_t	Vreme od nastanka UDT utičnice (ms)	1
pktSentTotal	int64_t	Ukupan broj poslatih paketa	1
pktRecvTotal	int64_t	Ukupan broj primljenih paketa	1
pktSndLossTotal	int	Ukupan broj paketa koji su izgubljeni u slanju	1
pktRcvLossTotal	int	Ukupan broj paketa koji su izgubljeni pri prijemu	1
pktRetransTotal	int	Ukupan broj paketa koji su ponovno poslati	1
pktSentACKTotal	int	Ukupan broj ACK paketa koji su poslati	1
pktRecvACKTotal	int	Ukupan broj ACK paketa koji su primljeni	1
pktSentNAKTotal	int	Ukupan broj NAK paketa koji su poslati	1
pktRecvNAKTotal	int	Ukupan broj NAK paketa koji su primljeni	1
pktSent	int64_t	Broj poslatih paketa	2
pktRecv	int64_t	Broj primljenih paketa	2
pktSndLoss	int	Broj paketa izgubljenih u slanju	2
pktRcvLoss	int	Broj paketa izgubljenih pri prijemu	2
pktRetrans	int	Broj ponovno poslatih paketa	2
pktSentACK	int	Broj poslatih ACK paketa	2
pktRecvACK	int	Broj primljenih ACK paketa	2
pktSentNAK	int	Broj poslatih NAK paketa	2
pktRecvNAK	int	Broj primljenih NAK paketa	2
mbpsSendRate	double	Brzina slanja podataka (Mb/s)	2
mbpsRecvRate	double	Brzina primanja podataka (Mb/s)	2
usPktSndPeriod	double	Interval slanja paketa, u mikrosekundama	3
pktFlowWindow	int	Veličina prozora kontrole toka, u paketima	3
pktCongestionWindow	int	Veličina prozora kontrole zagušenja, u paketima	3
pktFlightSize	int	Broj paketa koji su poslati a nisu još uvek potvrđeni	3
msRTT	double	Procenjeno vreme obilaska (ms)	3
mbpsBandwidth	double	Procenjeni propusni opseg (Mb/s)	3
byteAvailSndBuf	int	Dostupna memorija u baferu pošiljaoca (B)	3
byteAvailRcvBuf	int	Dostupna memorija u baferu primaoca (B)	3

4.3. UDT test aplikacija

Kako bi bile testirane performanse UDT-a napravljena je aplikacija posebno za tu svrhu.

Cilj aplikacije je prenos datoteke veličine 2GB blok po blok.

Aplikaciju odlikuje maksimalna prilagodljivost. Vrlo lako se podešava broj niti za prenos podataka, veličina bloka podataka koji svaka nit šalje, opcije UDT utičnice i razni drugi parametri koji mogu da utiču na performanse.

Za sve vreme rada aplikacije su dostupni periodični izveštaji o performansama koje nam isporučuje posebna nit za praćenje performansi, kako na strani koja šalje, tako i na strani koja prima podatke.

Pseudokôd aplikacije za slanje podataka je opisan u tabeli 4.4., dok je pseudokôd aplikacije za prijem podataka opisan u tabeli 4.5.

Tabela 4.4. Pseudokôd UDT test aplikacije za slanje podataka. U levoj koloni je predstavljen kôd upravljačke niti dok je u desnoj koloni predstavljen kôd pomoćnih niti koje obavljaju transfer.

```
Napravi i poveži utičnice
Razmeni informacije o datoteci koja se prenosi

Napravi niti (za svaku utičnicu se pravi odgovarajuća nit)
Prosledi sve niti u red niti

Ponavljati blok dok pozicija ne pređe veličinu datoteke
{
   Zaključaj muteks
     Dok je red niti prazan,
       Čekaj signal od neke niti da je red niti uvećan

     Preuzmi sledeću nit iz reda niti
     Ukloni preuzetu nit iz reda niti

     Prosledi niti poziciju i veličinu prenosa
     Uvećati poziciju za vrednost veličine prenosa

     Postaviti nit u radno stanje
   Otključaj muteks
}

Pošalji svakoj niti signal prekida

Dok nisu sve niti završile sa radom,
   Čekaj signal o završetku niti
```

```
Ponavljati blok u petlji
{
   Dok je nit u stanju čekanja
     Čekaj signal od glavne niti o promeni stanja

   Ako je primljen signal o prekidu od glavne niti
   {
     Pošalji signal prekida drugoj strani
     Sačekaj potvrdu prijema signala od druge strane
     Izađi iz petlje
   }

   Preuzmi poziciju i veličinu prenosa od glavne niti
   Pošalji poziciju i veličinu prenosa drugoj strani

   Pročitaj iz datoteke (broj bajtova jednak vrednosti
veličine prenosa, i to počevši od date pozicije)
   Pošalji drugoj strani podatke pročitane iz datoteke

   Zaključaj muteks
     Postavi nit u stanje čekanja
     Dodaj nit u red niti
     Pošalji signal glavnoj niti da je red niti uvećan
   Otključaj muteks
}

Pošalji signal glavnoj niti da je nit završila sa radom
```

Tabela 4.5. Pseudokôd UDT test aplikacije za prijem podataka. U levoj koloni je predstavljen kôd upravljačke niti dok je u desnoj koloni predstavljen kôd pomoćnih niti koje obavljaju transfer.

Napravi i poveži utičnice
Razmeni informacije o datoteci koja se prenosi
Napravi niti (za svaku utičnicu se pravi odgovarajuća nit)
 	
Dok nisu sve niti završile sa radom,
  Čekaj signal o završetku niti
	Ponavljati blok u petlji
	{
	  Ako je druga strana poslala signal prekida
	  {
	  Pošalji potvrdu prijema signala drugoj strani
	  Prekini petlju
	  }
	
	  Primi od druge strane poziciju i veličinu prenosa
	  Primi podatke datoteke (broj bajtova jednak vrednosti veličine prenosa)
	
	  Primljene podatke upiši u datoteku (počevši od dogovorene pozicije)
	}
	
	Pošalji signal glavnoj niti da je nit završila sa radom

Pošto UDT utičnica nije sistemska, nije moguće koristiti istu UDT utičnica u više različitih procesa. Paralelizam se ostvaruje pravljenjem niti, na čemu se i zasniva ova aplikacija.

Suština rada aplikacije je ciklično korišćenje niti za slanje podataka preko odvojenih UDT utičnica. Kada nit postane dostupna prosledi joj se sledeći blok podataka za prenos i nit postaje nedostupna za vreme slanja tog bloka.

Sa druge strane, prijem podataka se odvija praktično neprekidno. Svaka nit čeka prijem podataka preko svoje UDT utičnice i po prijemu upisuje u lokalnu datoteku.

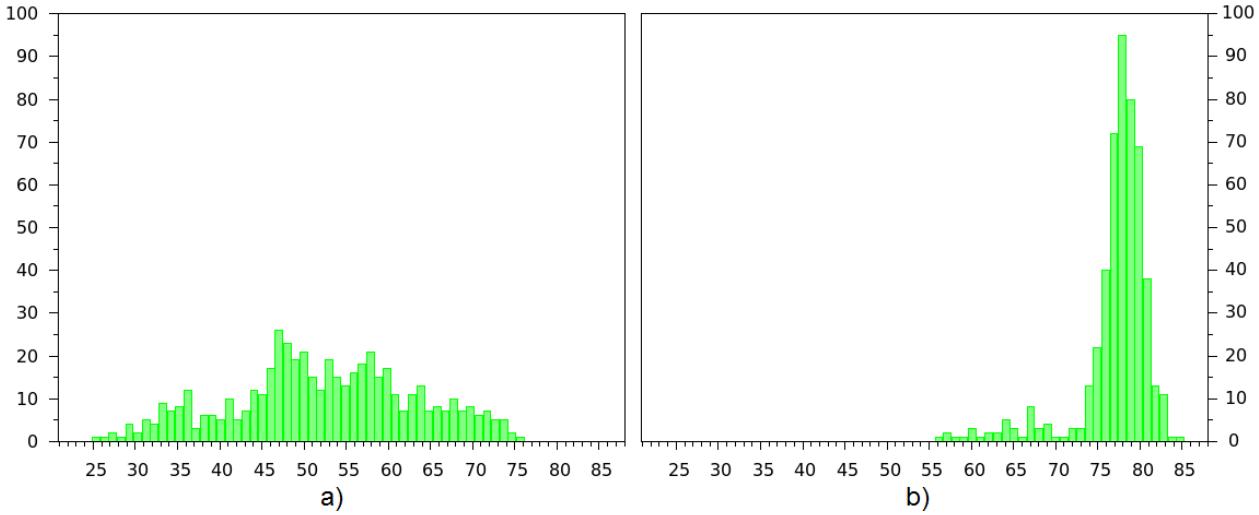
Zbog istovremenog izvršavanja više niti mora se obezbediti sinhronizacija tih niti. U tu svrhu, podaci deljeni između niti su zaštićeni *mutex* promenljivom.

Obezbeđen je mehanizam radi pravilnog završetka aplikacije. Po završetku slanja svih podataka, šalje se posebno definisan signal. Nakon prijema svih podataka učitava se signal i šalje povratni signal o potvrdi prijema signala. U tom trenutku obe strane mogu bezbedno da završe sa radom.

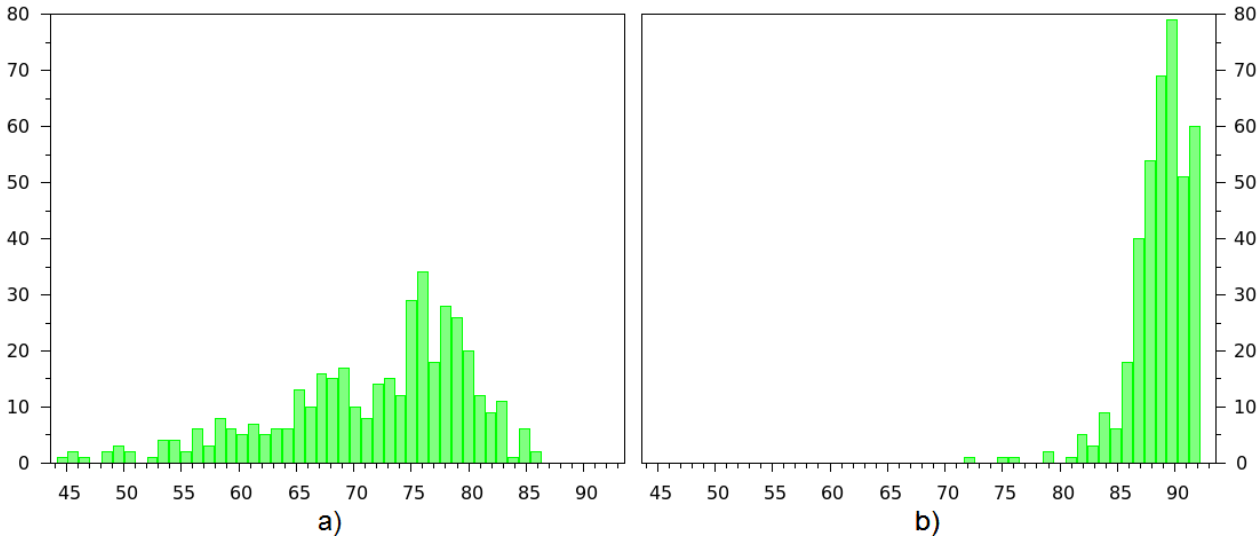
Takođe, po prijemu datoteke, računata je njena MD5 suma kako bi se ustanovilo da li je datoteka primljena pouzdano. Nakon svakog od ukupno 900 testova MD5 sume su se uvek poklapale. Što bi trebalo da otkloni sumnje u pouzdanost prenosa podataka preko UDT-a.

4.4. Rezultati testiranja

Na slikama 4.1. odnosno 4.2. su prikazane normalizovane distribucije ostvarenih brzina prenosa podataka i to za jedan odnosno osam istovremenih tokova, za 500 odnosno 400 izvršenih testova, respektivno.



Slika 4.1. Prikaz distribucija prenosa 2GB podataka sa jednom vezom. a) TCP prenos, b) UDT prenos. Na x-osi je prikazan izmereni protok u Mb/s, dok y-osa prikazuje broj puta kada je data brzina zabeležena, od ukupno 500 testova.



Slika 4.2. Prikaz distribucija prenosa 2GB podataka sa 8 paralelnih veza. a) TCP prenos, b) UDT prenos. Na x-osi je prikazan izmereni protok u Mb/s, dok y-osa prikazuje broj puta kada je data brzina zabeležena, od ukupno 400 testova.

Kao što je i očekivano, oba protokola za prenos podataka su pokazala bolje performanse uz korišćenje više istovremenih tokova za prenos podataka.

Uz jedan tok, TCP testovi daju prosek brzina prenosa podataka od 52.3Mb/s, dok nam UDT testovi daju prosečnu brzinu od 77.2Mb/s.

Kada je korišćeno 8 istovremenih tokova, TCP aplikacija je u proseku ostvarila brzinu od 71.6Mb/s, dok je UDT aplikacija prosečno slala podatke brzinom od 88.9Mb/s.

Na prikazu distribucija brzina se jasno vidi da brzine TCP-a jako variraju, dok je odstupanje brzina ostvarenih preko UDT-a od prosečne brzine minimalno. Što je takođe vrlo bitna stvar, jer nam ilustruje daleko veću stabilnost UDT-a.

Iako je TCP bio podešen da koristi jedan od novijih algoritama za kontrolu zagušenja – *TCP Illinois*. *TCP Illinois* se pokazao kao daleko bolji od standardnog TCP algoritma koji je u istim ovim mrežnim uslovima dostigao prosečnu brzinu od oko 15Mb/s [16], [17]. Iskazani rezultati sugerišu da ni uz dalja poboljšanja TCP ne može dostići brzine UDT-a.

Rezultati koje je postigao UDT su još više zapanjujući uzimajući u obzir da je čak i u kontrolisanim uslovima testiranja, na direktnoj vezi kapaciteta 100Mb/s između dva računara, najveća zabeležena brzina prenosa podataka 95Mb/s [16].

5. Zaključak

Tokom godina postalo je jasno da TCP kao isuviše konzervativan algoritam ne može još dugo održati primat među protokolima za prenos podataka. Iako je taj primat zaslužen višedecenijskim pružanjem kvalitetnih usluga, mora se naglasiti da je taj kvalitet bio u velikoj meri obezbeđen mrežnim uslovima bliskim optimalnim, sa malo korisnika i malom kapacitetu veza. Kako su broj korisnika i kapacitet veza rasli tako su opadale performanse TCP-a.

Brojna unapređenja TCP-a - od dodavanja raznih opcija do unapređivanja kontrole zagušenja su poboljšala performanse. Ipak je jasno da ta poboljšanja dostižu svoju gornju granicu preko koje TCP ne može preći. Ta granica, posmatrana u procentima iskorišćenosti dostupnog opsega, značajno opada sa porastom kapaciteta.

Savremene tehnologije se svakim danom dalje razvijaju i pitanje je vremena kada će se kapaciteti javnih mreža povećati višestruko. Tako da se bližimo trenutku kada će performanse TCP-a postati neprihvatljive.

UDT je prvobitno projektovan za privatne mreže, sa kapacitetom veza od 1Gb/s, 10Gb/s ili čak više. Očekuje se da će UDT na tim bržim vezama postići procentualno slične rezultate, dok će za TCP procenat iskorišćenosti propusnog opsega na vezama kapaciteta 1Gb/s ili više drastično opasti.

Takođe, u ovom radu se pokazalo da UDT može da funkcioniše jako dobro i na javnoj mreži kakav je Internet, uz brzine od 100Mb/s. To već sada potvrđuje kandidaturu UDT-a za mesto naslednika TCP-a. A očekuje se da će u godinama koje dolaze UDT samo poboljšati svoje performanse i time obezbediti svoje zasluženo mesto, mesto u jezgrima operativnih sistema korisnika širom sveta, mesto standardnog protokola za pouzdani prenos podataka.

Korišćena literatura

[1] Information Sciences Institute, University of Southern California, *Transmission Control Protocol*, Internet Engineering Task Force, RFC793, septembar 1981.

[2] Vinton G. Cerf, Robert E. Kahn, *A Protocol for Packet Network Intercommunication*, IEEE Transactions on Communications, broj 5, maj 1974.

[3] R. Braden, *Requirements for Internet Hosts - Communication Layers*, Internet Engineering Task Force, RFC1122, oktobar 1989.

[4] M. Allman, V. Paxson, *TCP Congestion Control*, Internet Engineering Task Force, RFC5681, septembar 2009.

[5] V. Jacobson, *Congestion Avoidance and Control*, ACM SIGCOMM, str. 314-329, avgust 1988.

[6] S. Floyd, K. Fall, *Promoting the Use of End-to-End Congestion Control in Internet*, IEEE/ACM Transactions on Networking, avgust 1999

[7] S. Liu, T. Basar, R. Srikant, *TCP-Illinois - A Loss and Delay-Based Congestion Control Algorithm for High-Speed Networks*, Proceedings of the First International Conference on Performance Evaluation Methodologies and Tools, 2006.

[8] K. Thompson, G. Miller, R. Wilder, *Wide-area internet traffic patterns and characteristics*, IEEE, tom 11, broj 6, str. 10–23, novembar 1997.

[9] T.V. Lakshman and U. Madhow, *The performance of TCP/IP for networks with high bandwidth-delay products and random loss*, IEEE/ACM Transactions on Networking, jun 1997.

[10] M. Mathis, J. Mahdavi, *TCP Selective Acknowledgment Options*, Internet Engineering Task Force, RFC2018, oktobar 1996.

[11] M. Mathis, J. Mahdavi, *Forward Acknowledgment: Refining TCP Congestion Control*, ACM SIGCOMM Computer Communication Review Homepage, tom 26, izdanje 4, str. 281-291, oktobar 1996.

[12] V. Jacobson, R. Braden, *TCP Extensions for High Performance*, Internet Engineering Task Force, RFC1323, maj 1992.

[13] V. Paxson, M. Allman, *Computing TCP's Retransmission Timer*, Internet Engineering Task Force, RFC2988, novembar 2000.

[14] J. Kempf, R. Austein, *The Rise of the Middle and the Future of End-to-End: Reflections on the Evolution of the Internet Architecture*, Internet Engineering Task Force, RFC3724, mart 2004.

[15] J. Postel, *User Datagram Protocol*, Internet Engineering Task Force, RFC768, avgust 1980.

[16] M. Šošić, V. Stojanović, *TCP Congestion Control Algorithms - Analysis, Experimental Evaluation and Comparison*, ETRAN, jun 2013.

[17] M. Šošić, V. Stojanović, *Resolving Poor TCP Performance on High-speed Long Distance Links*, SISY, septembar 2013.

[18] A. Tanenbaum, D. Wetherall, *Computer networks*, Pearson Prentice Hall, izdanje 5, 2010.

[19] H. Schulzrinne, S. Casner, *RTP: A Transport Protocol for Real-Time Applications*, Internet Engineering Task Force, RFC3550, jul 2003.

[20] D. Borman, S. Deering, *IPv6 Jumbograms*, Internet Engineering Task Force, RFC2675, avgust 1999.

[21] Y. Gu, R. Grossman, *UDT: UDP-based Data Transfer for High-Speed Wide Area Networks*, Computer Networks (Elsevier Press), tom 51, izdanje 7, maj 2007.

[22] S. Kishore, R. Chandra, D. Ganesh, *Exploring Configurable Congestion Control Feature of UDT Protocol*, International Journal of Computer Science and Telecommunications, tom 3, izdanje 5, maj 2012.

[23] R. Grossman, Y. Gu, X. Hong, *Teraflows over Gigabit WANs with UDT*, Journal of Future Computer Systems (Elsevier Press), tom 21, broj 4, str. 501-513, 2005.

[24] Y. Gu, X. Hong, R. Grossman, *An Analysis of AIMD Algorithms with Decreasing Increases*, First Workshop on Networks for Grid Applications, oktobar 2004.

[25] C. Dovrolis, P. Ramanathan, D. Moore, *What do packet dispersion techniques measure?*, *IEEE Infocom*, april 2001.

[26] Sajt za preuzimanje *UDT* biblioteke: http://udt.sourceforge.net/

[27] Sajt za preuzimanje *iperf* aplikacije: http://sourceforge.net/projects/iperf/