Demonstracija rada alternativnih transportnih protokola korištenjem mrežnog simulatora IMUNES

Kotarac, Alen

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1368

DEMONSTRACIJA RADA ALTERNATIVNIH TRANSPORTNIH PROTOKOLA KORIŠTENJEM MREŽNOG SIMULATORA IMUNES

Alen Kotarac

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1368

DEMONSTRACIJA RADA ALTERNATIVNIH TRANSPORTNIH PROTOKOLA KORIŠTENJEM MREŽNOG SIMULATORA IMUNES

Alen Kotarac

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1368

Pristupnik:	**Alen Kotarac (0036528023)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	izv. prof. dr. sc. Miljenko Mikuc

Zadatak: **Demonstracija rada alternativnih transportnih protokola korištenjem mrežnog simulatora IMUNES**

Opis zadatka:

U današnjim IP mrežama standardno se koriste transportni protokoli TCP, za konekcijski orijentirane aplikacije i UDP, za beskonekcijske. Oni ujedno čine većinu mrežnog prometa. Pored njih standardizirani su i podržani dodatni transportni protokoli prilagođeni zahtjevima pojedinih usluga. Najpoznatiji je SCTP (Stream Control Transmission Protocol) koji se intenzivno koristi u telekomunikacijama, a operacijski sustav Linux podržava još i protokole DCCP (Datagram Congestion Control Protocol) i UDP-Lite (The Lightweight User Datagram Protocol). Vaš zadatak je istražiti mogućnosti, prednosti i nedostatke alternativnih (eksperimentalnih) transportnih protokola u odnosu na odgovarajuće slične standardne i protokole. Rad protokola demonstrirajte jednostavnim klijentskim i poslužiteljskim aplikacijama korištenjem sustava IMUNES tako da se pojedine komponente mogu izvoditi na virtualnim čvorovima.

Rok za predaju rada: 14. lipnja 2024.

Hvala Nari.

Sadržaj

Uvod

Internetski protokoli su važan dio komunikacije između uređaja na mrežama širom svijeta, a protokoli transportnog sloja čine temelj te komunikacije. Danas najčešće korišteni transportni protokoli su TCP (Transmission Control Protocol) i UDP (User Datagram Protocol). TCP se koristi u slučajevima koji zahtijevaju pouzdanost komunikacije i točan redoslijed poslanih podataka, dok se UDP koristi kada TCP i njegove karakteristike nisu potrebne i želi se koristiti jednostavniji protokol. Iako ti protokoli čine većinu prometa i pokrivaju širok raspon potreba korisnika, alternativni protokoli su razvijeni kako bi se pružile dodatne mogućnosti korisnicima u slučajevima gdje korištenje TCP i/ili UDP protokola nije efikasno. Možda i najpoznatiji takav protokol naziva se SCTP (Stream Control Transmission Protocol). DCCP (Datagram Congestion Control Protocol) i UDP-Lite (Lightweight User Datagram Protocol) su također protokoli razvijeni kao alternativa TCP-u i UDP-u. Naravno, postoji više protokola transportnog sloja koji služe kao alternativa standardnim protokolima te ćemo se njih dotaknuti kasnije.

SCTP je prvotno bio dizajniran za korištenje u SS7 protokolu zato što su već postojeći protokoli pokazali nezadovoljavajućima u tom sustavu. SCTP je pouzdan protokol koji nudi slične prednosti kao TCP protokol, uključujući točnost redoslijeda poslanih podataka i kontrolu zagušenja. DCCP se koristi na operacijskim sustavima Linux i FreeBSD i razvijen je kao alternativa UDP protokolu, nudi karakteristike UDP protokola i dodaje kontrolu zagušenja.

Ovaj rad se bavi istraživanjem alternativnih protokola transportnog sloja, njihovih nedostataka, prednosti i razlika u usporedbi s već ustaljenim TCP i UDP protokolima te će demonstrirati rad nekih od tih protokola na mrežnom simulatoru IMUNES.

1. Pregled standardnih protokola transportnog sloja

U ovom poglavlju ćemo istražiti značajke transportnih protokola koji se najčešće koriste danas, točnije TCP i UDP.

1.1. TCP (Transmission Control Protocol)

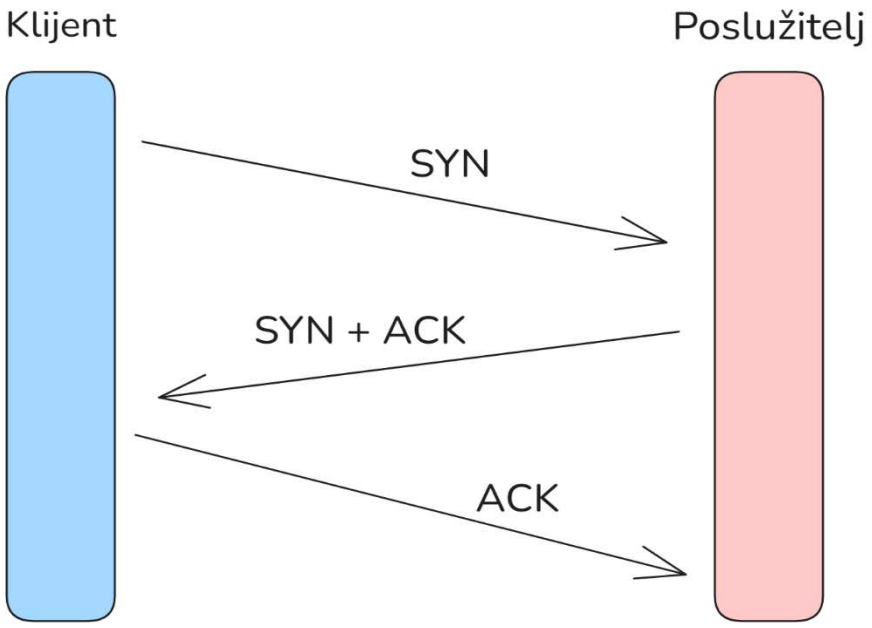
Početni rad na internetskim protokolima, iz kojih će poslije biti razvijen TCP, objavljen je 1974. Radi se o specifikaciji RFC 675 (Specification of Internet Transmission Program) čiji su autori Vint Cerf, Yogen Dalal i Carl Sunshine. Nešto kasnije program je podijeljen u dva dijela, TCP i IP, kako bi se mogli koristiti kao modularna arhitektura.

TCP je konekcijski orijentiran protokol, što znači da prije razmjene podataka dva uređaja moraju uspostaviti vezu.

Uspostavljanje veze se ostvaruje tzv. „3-way handshake“ procesom, gdje klijent i poslužitelj razmjenjuju podatke potrebne za uspostavu veze.

Da bi objasnili uspostavu vezu trebamo spomenuti zastavice koje se koriste u tom procesu. SYN, ACK i FIN predstavljaju zastavice u segmentima koji se šalju tijekom uspostave veze. SYN (Synchronize Sequence Number) je prva po redu zastavica i označava da će doći do početka komunikacije i kojim rednim brojem će segment započeti. Ako segment s SYN zastavicom šalje klijent to označava kojim će rednim brojem započeti njegovi segmenti, a ako segment s SYN zastavicom šalje poslužitelj to označava kojim će rednim brojem započeti segmenti koji šalje poslužitelj. ACK (Acknowledgement) zastavica označava da je neka poruka primljena. FIN (Finish) je zastavica koja označava zahtjev jednog od dva sudionika za prekidom veze.

Klijent uspostavlja komunikaciju tako da pošalje segment sa SYN zastavicom, ako je poslužitelj spreman i sluša na vratima gdje klijent šalje segment sa SYN zastavicom, onda odgovara sa segmentom koji sadrži SYN/ACK zastavicu, gdje ACK zastavica označava da je poruka primljena, a SYN označava kojim će rednim brojem započeti segment koji šalje poslužitelj. Klijent ACK zastavicom daje do znanja poslužitelju da je sve spremno za razmjenu podataka i veza je uspostavljena (Slika 1.1).



Slika 1.1

Sada kada znamo kako funkcionira proces uspostave veze kod TCP protokola, dotaknut ćemo se i tzv. „SYN flood“ napada, koji spada u DoS (Denial of Service) napade. „SYN flood“ iskorištava standardni proces uspostave veze kod TCP protokola kako bi ometao poslužitelj u radu. Napadač šalje veliki broj segmenata sa SYN zastavicom prema poslužitelju, ali sprječava poslužitelj i klijent da uspostave veze. To može činiti tako da TCP segmente šalje bez IP adrese i tako sprječava da segment sa SYN + ACK zastavicom dođe nazad do klijenta. Napadač se može koristiti jednim računalom ili može koristiti mrežu od više računala kako bi slao veliki broj segmenata sa SYN zastavicom prema poslužitelju. Kada dođe do početka uspostave veze, ali se ona ne može u potpunosti uspostaviti dolazi do alokacije računalnih resursa na poslužitelju koji čekaju potpuno spajanje na klijenta, ali zato što se ni jedna veze ne uspostavlja u potpunosti poslužitelj će iskusiti velike probleme s performansama i spojivosti.

Sprječavanje ovakvog napada najčešće se radi povećavanjem kapaciteta reda u kojem zahtjevi za spajanje čekaju ili korištenje nekakve politike uređivanja toga reda. Na primjer, kada se red ispuni, najstariji zahtjevi za spajanje mogu se poništavati i najnoviji staviti na njihovo mjesto. Postoje također, i vanjski servisi, poput servisa Cloudflare, koji sprječavaju ovakve i slične napade tako da prije potpune uspostave veze klijenta sa poslužiteljem prvo proces uspostave provode kroz vlastite poslužitelje, te djeluju kao posrednik.

Postoje dva načina raskidanja veze u TCP protokolu. Prvi je tzv. „graceful“ način gdje se veza prekida tako da je ona otvorena sve dok oba uređaja ne zatvore vezu sa svoje strane. To se radi s pomoću FIN signala. Svaka strana šalje signal FIN, a druga odgovara signalom ACK. Nakon što je poslan signal ACK veza je zatvorena na strani koja je poslala signal FIN.

Drugi način je nagli raskid veze. Do toga dolazi kada je jedna strana iz nekog razloga primorana zatvoriti vezu ili jedan uređaj zatvara vezu s obje strane. Do takvog raskida dolazi kada se pošalje signal RST.

Neke od bitnih značajki TCP-a su:

- Pouzdanost: TCP koristi već spomenute ACK potvrde kako bi osigurao da je podatak primljen, a ako nije primljen dolazi do retransmisije gdje se podatci za koje nije primljena potvrda ponovno šalju
- „byte-oriented“: Za razliku od ostalih protokola koje danas spominjemo, TCP podatke broji bajt po bajt tijekom slanja i primanja, te ih šalje kontinuiranim tokom podataka
- Kontrola zagušenja: U situacijama gdje dolazi do zagušenja mreže TCP može koristit određene algoritme kako bi pokušao predvidjeti koju količinu podataka može poslati kojem primatelju u nekom vremenskom razdoblju, te po potrebi može vršiti retransmisiju izgubljenih podataka
- Kontrola toka: Protokol može ograničiti količinu podataka koji se šalju kako ne bi došlo do preopterećenja primatelja
- Full Duplex: Podatci se mogu slati istodobno od klijenta prema poslužitelju i obrnuto

TCP segment se sastoji od zaglavlja i podataka. Zaglavlje se sastoji od:

- Vrata izvorišta i odredišta
- Redni broj
- Potvrdni broj
- Kontrolne zastavice
- Veličina prozora
- Opcije i ispunjavanje

1.2. UDP (User Datagram Protocol)

Protokol UDP je opisan specifikacijom RFC 768 1980. godine, a autor je David P. Reed.

UDP nije, za razliku od TCP-a, konekcijski orijentiran protokol što znači da nije potrebna uspostava veze prije slanja podataka. Također, to znači da UDP ne prati status poslanih podataka i ne može znati jesu li ti podatci primljeni na drugom kraju veze ili jesu li došli u pravilnom redoslijedu.

UDP je namijenjen za slanje podataka većom brzinom nego u TCP-u, za slučajeve kada potpuna pouzdanost i redoslijed podataka nije primarni cilj.

Osnovne značajke UDP protokola su:

- Ne postoji uspostava veze („connectionless“)
- Nepouzdan: Nema garancije da su svi poslani podatci primljeni
- „message-oriented“: Šalje podatke u obliku diskretnih poruka, skup podataka se razdvaja na poruke koji se onda šalju poruku po poruku
- Brz prijenos podataka: Zbog malog zaglavlja i jednostavnog dizajna brzina prijenosa podataka je brza
- Nema ugrađenu kontrolu zagušenja: Protokol sam po sebi nema način nošenja sa zagušenjem mreže

Kao i kod TCP-a, UDP segment sastoji se od zaglavlja i podataka (Slika 1.2). UDP zaglavlje sastoji se od:

- Vrata izvorišta i odredišta
- Dužina datagrama
- Kontrolni zbroj (koristi se za provjeru integriteta podataka)

Slika 1.2

Primarna upotreba ovog protokola je u situacijama gdje nam je prioritet brzina prijenosa podataka i gdje možemo tolerirati pojedine izgubljene pakete ili pakete koji dolaze do odredišta u nepravilnom redoslijedu. Primjeri takve upotrebe su video ili audio pozivi, video igre povezane na internet. UDP se često koristi za DNS upite zato što je za DNS upite najčešće potreban samo jedan upit za koji dolazi jedan odgovor. Ako je za DNS upit potrebno više informacije može se koristiti i TCP.

2. Pregled alternativnih protokola transportnog sloja

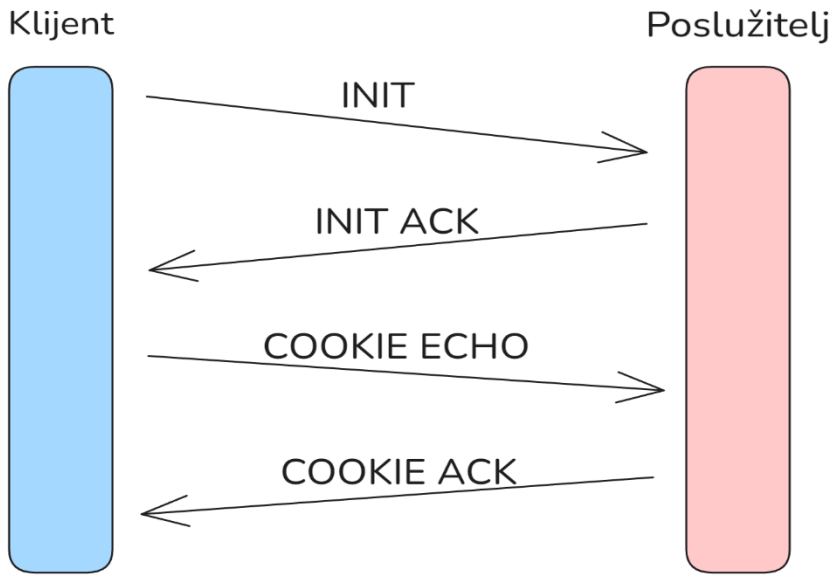
2.1. SCTP (Stream Control Transmission Protocol)

SCTP ili Stream Control Transmission Protocol je protokol prvi put opisan 2000. godine. Izvorno je protokol bio zamišljen za sustav SS7 za prijenos poruka. Prvi prijedlog za ovaj protokol došao je od strane IETF SIGTRAN grupe. SCTP je specifikacijom RFC 9260 standardiziran od strane IETF (Interne Engineering Task Force).

Glavna značajka SCTP je to što kombinira neke značajke TCP-a i UDP-a u jedan protokol, te omogućuje njegovu primjenu u drugačijim situacijama. SCTP je zamišljen da pruži dodatne i bolje mogućnosti u odnosu primarno na TCP protokol te ćemo ga zato primarno uspoređivati s TCP protokolom u ovom radu.

Ovaj protokol je, poput TCP-a, konekcijski orijentiran protokol i mora uspostaviti vezu između dva uređaja prije nego započne slanje podataka.

SCTP koristi tzv. „4-way handshake“ gdje klijent započinje uspostavu vezu slanjem signala INIT. Poslužitelj odgovara signalom INIT-ACK koji također sadrži i „kolačić“ (cookie) u kojem se nalazi MAC (Message Authentication Code) i još neke informacije potrebne za uspostavu veze. MAC se u ovom slučaju koristi za potvrdu autentičnosti odgovora tako da klijent odgovara COOKIE-ECHO signalom koji sadrži isti „kolačić“ koji mu je poslužitelj poslao. Zadnji korak se sastoji od provjere autentičnosti COOKIE-ECHO signala, tj. „kolačića“ u tom signalu te poslužitelj šalje zadnji signal COOKIE-ACK i veza se uspostavlja(Slika 2.1).



Slika 2.1

SCTP također ima mehanizam zatvaranja veze koji se sastoji od 3 koraka. Ovdje se radi o tzv. „graceful“ prekidu veze. Nakon što je poslao sve podatke i nema potrebe za održavanjem uspostavljene veze, klijent poslužitelju šalje signal SHUTDOWN, poslužitelj odgovara signalom SHUTDOWN-ACK te se proces završava kada klijent pošalje zadnji signal, a to je signal SHUTDOWN-COMPLETE.

SCTP, kao i TCP, podržava nagli prekid veze. Kada se signal ABORT pošalje s bilo koje strane veze ona se prekida.

Osnovne značajke SCTP protokola su:

- Jednostavnija implementacija: U usporedbi s TCP-om SCTP koristi jednostavniji model uspostave i održavanje veze
- „Message-oriented“: Šalje diskretne poruke
- „Multi streaming“: Unutar jedne veze SCTP može prenositi podatke na više nezavisnih tokova
- „Multihoming“: SCTP može imati više od jedne IP adrese na svakom kraju veze, prijenos podataka može se odvijati na jednoj adresi ako je druga nedostupna
- Kontrola zagušenja: Kao i TCP, SCTP može nadoknaditi izgubljene pakete i nepravilno razvrstane pakete do kojih dolazi zbog zagušenja mreže
- Kontrola toka: Brzina prijenosa podataka može se mijenjati u ovisnosti o mogućnostima primatelja

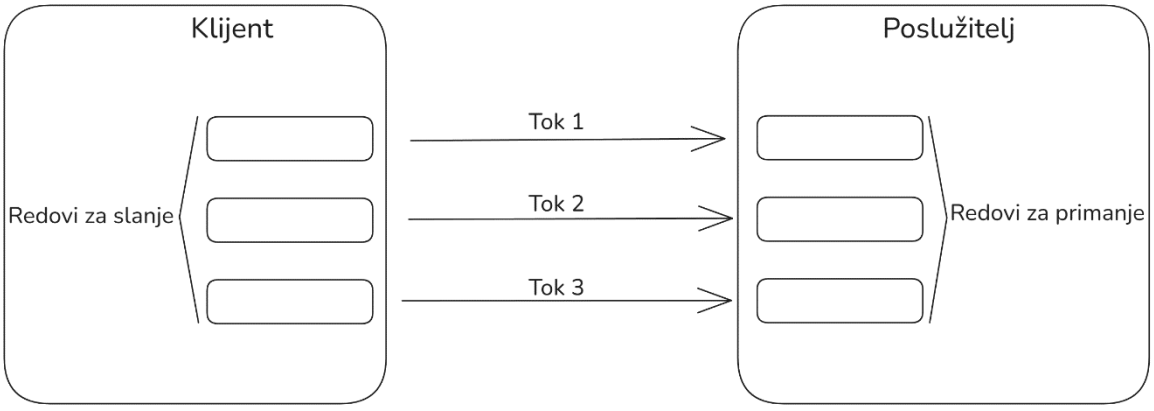
Slika 2.2

SCTP segment (Slika 2.2) sastoji se od zaglavlja i podataka. Blok podataka sastoji se od „chunkova“ koji mogu biti podatkovni ili kontrolni. Zaglavlje segmenta sastoji se od:

- Vrata izvorišta i odredišta
- Verifikacijska oznaka (služi za razlikovanje paketa)
- Kontrolni zbroj (koristi se za provjeru integriteta podataka)

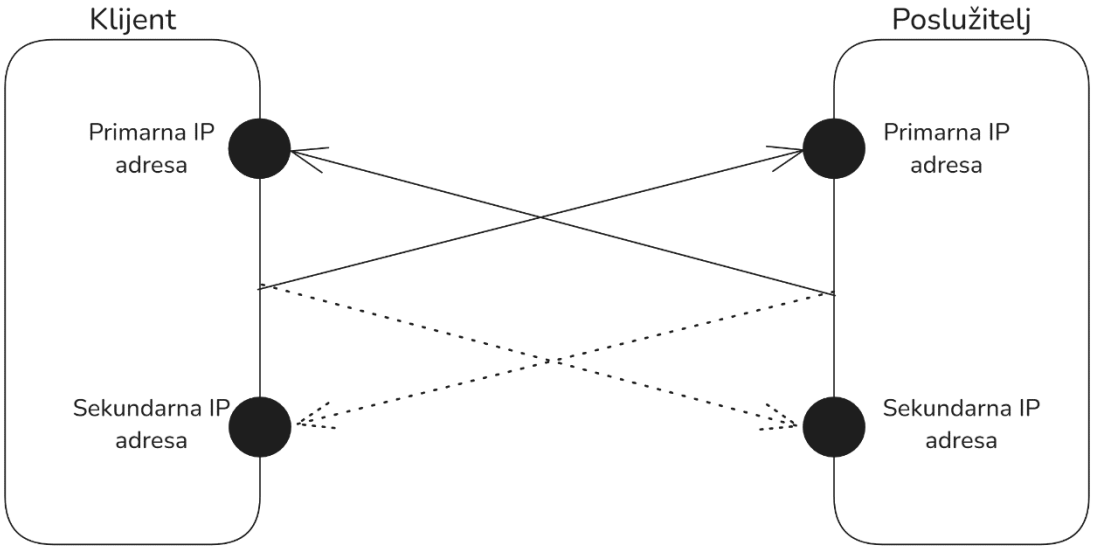
U svakom segmentu postoji više „chunkova“. Jedan od njih se sastoji od:

- Tip „chunka“
- Zastavica
- Dužina
- Podatci



Slika 2.3

Dvije najzanimljivije značajke SCTP protokola su „multi streaming“ i „multihoming“. „Multi streaming“ (Slika 2.3) omogućava više nezavisnih tokova podataka unutar jedne SCTP veze, kroz svaki tok podatci mogu teći u samo jednom smjeru. Svaki tok je nezavisan što znači da problem na jednom toku ne ometa protok podataka na drugom. Više istovremenih tokova omogućava efikasniju upotrebu mrežnih resursa. Različitim tokovima mogu se slati različite vrste podataka.



Slika 2.4

„Multihoming“ (Slika 2.4) pruža redundanciju SCTP vezi. Svaka strana veze ima više IP adresa koje se mogu koristiti. Primarna IP adresa je ona koja se koristi ako sve funkcionira očekivano, ali ako dođe do problema sa tom IP adresom, SCTP može početi slati podatke na drugu IP adresu primatelja bez da to korisnik primijeti. Koriste se poruke tzv. „otkucaja srca“ kako bi klijent i poslužitelj znali jesu li sekundarne IP adrese aktivne i spremne primati i slati podatke.

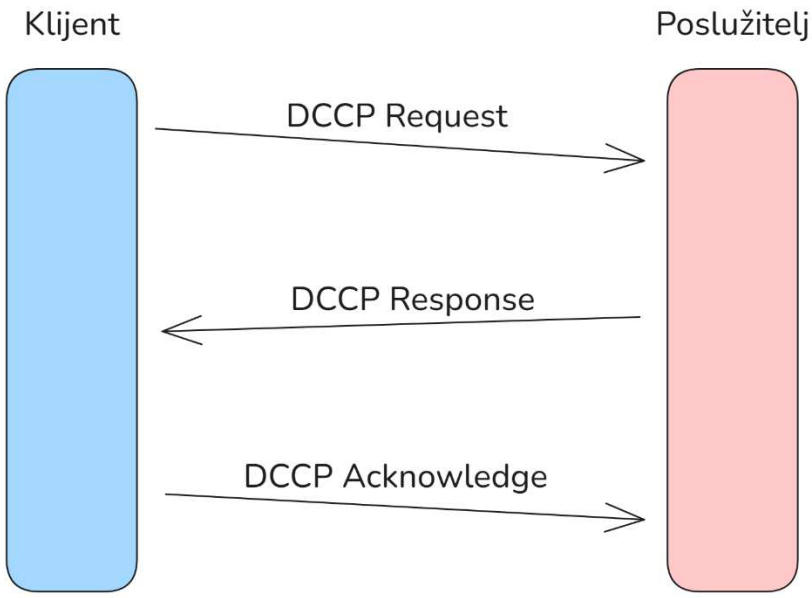
SCTP se najčešće koristi u slučajevima gdje je pouzdanost i sigurnost primarni cilj. Koristi se za prijenos signala u telefoniji, pozive preko interneta (Voice over IP), financijske aplikacije i sustave i ponekad za prijenos multimedije. „Multi streaming“ se može koristiti za učitavanja web stranica na način da se u isto vrijeme može učitavati i tekst i slike sa stranice korištenjem više odvojenih tokova unutar jedne veze. „Multihoming“ pruža redundanciju kada se koristi u sustavima gdje je važna pouzdanost kao već spomenute financijske aplikacije.

2.2. DCCP (Datagram Congestion Control Protocol)

DCCP je prvi put opisan specifikacijom RFC 4340 koja je objavljena 2006. godine što čini ovaj protokol mlađim čak i od SCTP-a. Autori specifikacije su Eddie Kohler, Mark Handley i Sally Floyd. Zamisao pri dizajniranju protokola bila je pružiti proširene mogućnosti u odnosu na UDP protokol.

DCCP je nepouzdani protokol što znači da nema sustav koji će osigurati da svi podatci stignu do odredišta u pravilnom redoslijedu. Primarni cilj DCCP-a je ponuditi prednosti UDP-a s kontrolom zagušenja koja je ugrađena u sam protokol te se ne treba implementirati u aplikacijskom sloju. TCP također ima ugrađenu kontrolu zagušenja, ali nije prikladan za iste primjene kao UDP. Slučajevi poput audio i video poziva ili video igara povezanih na internet nisu idealni za korištenje TCP-a zato što nije potrebno da svaki segment dođe na odredište, ali je što manje kašnjenje i pravovremenost dolaska podataka je poželjna. Zbog toga se u ovim slučajevima češće koristi UDP. DCCP dodaje kontrolu zagušenja i zadržava ostale karakteristike UDP-a.

Uspostavljanje veze funkcionira na sličan način kao i kod TCP-a, doduše proces je jednostavniji i brži. Proces je jednostavniji zato što se u procesu uspostave veze uspostavlja samo mehanizam kontrole zagušenja između klijenta i poslužitelja, dok se kod TCP-a uspostavlja više funkcija uključujući i kontrolu zagušenja. DCCP nije pouzdan protokol i zato je proces uspostave veze jednostavniji u usporedbi s TCP-om. Radi se o tzv. „3-way handshake“ procesu. Klijent šalje DCCP- Request poslužitelju, a on na taj signal odgovara signalom DCCP-Response, kako bi završili uspostavljanje veze klijent šalje signal DCCP-Ack (Slika 2.5).



Slika 2.5

Postoji i proces prekidanja uspostavljene veze koji se sastoji od minimalno 3 koraka, moguće i 4. Klijent šalje poslužitelju DCCP-CloseReq koji inicira zatvaranje veze, poslužitelj će zatim odgovoriti DCCP-Close kojim daje do znanja da je primio signal, te se veza zatvara signalom klijenta DCCP-Reset. Poslužitelj nakon ovoga može također poslati DCCP-Reset kao potvrdu, ali to nije obavezno za zatvaranje veze.

Signal DCCP-Reset se također može koristiti za naglo zatvaranje veze, ako je to potrebno.

Navesti ćemo neke od osnovnih značajki DCCP protokola:

- Beskonekcijski: iako je uspostava veze potrebna, DCCP nije konekcijski orijentiran protokol, tijekom uspostave veze se razmjenjuju parametri potrebni za kontrolu zagušenja
- „Message-oriented“: Šalje podatke diskretnim porukama
- Nepouzdan: ne garantira dolazak svih podataka u pravilnom redoslijedu do odredišta
- Kontrola zagušenja: sprječava gubitke nastale kao posljedica zagušenja mreže

DCCP datagram sastoji se od zaglavlja i podataka. Elementi zaglavlja su sljedeći:

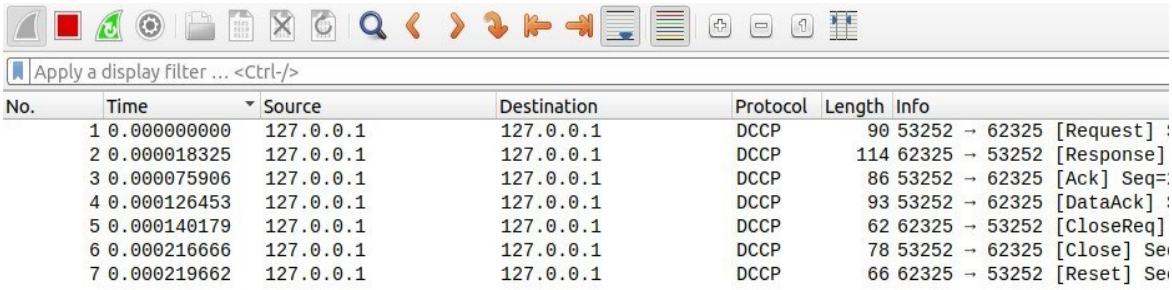
- Vrata izvorišta i odredišta
- Redni broj (identifikacija i redoslijed podataka)
- Kontrolni zbroj (provjera integriteta)
- Opcije za kontrolu zagušenja

Primjene DCCP su slične kao i za UDP, to znači prijenos audio i video podataka uživo, video igre povezane na internet, audio i video pozivi i slično. DCCP se koristi umjesto UDP upravo zbog mogućnosti kontrole zagušenja koju nije potrebno implementirati naknadno već je implementirana u protokol.

U našem mrežnom simulatoru nisu standardno dostupni alati kojima bi jednostavno mogli testirati DCCP, zbog toga ćemo demonstrirat jednostavnu komunikaciju između klijenta i poslužitelja koristeći Linux. Za ovu demonstraciju koristimo kod u programskom jeziku C preuzet s GitHub- a [10]. Ova demonstracija simulira slanje kratke poruke između klijenta i poslužitelj, simulirajući klijent i poslužitelj na istom računalu.

- Prvo moramo „kompajlirati“ naš program, to radimo u istom direktoriju gdje se nalazi naš kod:
  - `gcc dccptest.c -o server -Wall`
  - `ln -s server client`
- Nakon toga, možemo pokrenuti naš poslužitelj:
  - `./server`
- Zadnji korak je pokretanje klijenta, nakon pokretanja klijent automatski šalje poruku poslužitelju:
  - `./client`

Rezultat ovog kratkog testa je prikazan na Slici 2.6 kao zapis paketa u alatu Wireshark.



Slika 2.6

U prva 3 reda zapisa, vidimo da se događa proces uspostave veze, klijent je prvi koji šalje zahtjev za vezom (Request), poslužitelj odgovara signalom Response i veza se uspostavlja nakon što klijent pošalje signal Ack.

Četvrti paket predstavlja podatke koje se šalju na poslužitelj koristeći opcionalni DataAck paket koji sadrži podatke za potvrdu primanja na odredišnoj adresi. Ovdje je također mogao biti i Data paket koji ne sadrži podatke za potvrdu primitka.

Zadnja 3 reda zapisa predstavljaju zatvaranje DCCP veze. Poslužitelj šalje CloseReq signal kojim zahtjeva zatvaranje veze. Klijent odgovara signalom Close koji potvrđuje da će se veza zatvoriti. Veza se konačno zatvara nakon što poslužitelj pošalje signal Reset.

```
Datagram Congestion Control Protocol, Src Port: 53252, Dst Port: 62325 [DataAck] Seq=2
    Source Port: 52352
    Destination Port: 62325
    [Stream index: 0]
    Data Offset: 12
    CCVal: 0
    Checksum Coverage: 0
    Checksum: 0x911e [correct]
    [Checksum Status: Good]
    Type: DataAck (4)
    Extended Sequence Numbers: True
    Sequence Number: 2    (relative sequence number)
    Sequence Number (raw): 17867828702
    Acknowledgement Number: 0    (relative acknowledgement number)
    Acknowledgement Number (raw): 38401579842
    Options: (24 bytes)
```

Slika 2.7

Još želimo pobliže pogledati spomenuti DataAck paket, pa ćemo to napraviti uz predloženu sliku tog paketa (Slika 2.7) iz alata Wireshark. Prve dvije informaciju su izvorišna i odredišna vrata veze, sljedeće imamo pomak podataka i parametar CCVal koji se koristi za praćenje verzije algoritma za kontrolu zagušenja. Pokrivenost kontrolnog zbroja ili „checksum coverage“ određuje koji dijelovi paketa su pokriveni provjerom kontrolnog zbroja, te nakon toga odmah slijedi i sama vrijednost kontrolnog zbroja. Sljedeće vidimo parametar tip, koji označava koji je tip paketa u pitanju, u našem slučaju radi se o tipu 4, tj. DataAck paketu. Do kraja paketa imamo redne i potvrdne brojeve kojima se prate paketi i njihov redoslijed u određenoj DCCP vezi. Brojevi se broje redno od otvaranja do zatvaranja DCCP veze. Redni i potvrdni brojevi omogućavaju DCCP-u da potvrđuje je li neki paket stigao na odredište. Paketi s odgovorom Ack koriste te brojeve kako bi obavijestili pošiljatelja da je određeni paket stigao do poslužitelja.

2.3. Ostali alternativni protokoli

SCTP i DCCP su fokus ovog rada, ali postoje i druge alternative TCP-u i UDP-u. U ovom poglavlju ćemo kratko opisati neke od njih.

2.3.1. QUIC (Quick UDP Internet Connections)

QUIC je protokol transportnog sloja koji je razvio Google. Glavni cilj je smanjenje kašnjenja i povećanja performansi internetskih aplikacija. Radi nad UDP-om i kombinira funkcije transportnog i sigurnosnog sloja kroz primjenu TLS/SSL enkripcije. QUIC također koristi „3-way handshake“ proces, ali unutar samog procesa implementira i razmjenu ključeva za TLS kriptografski protokol. Koristi više tokova podataka unutar jedne veze tako da problemi na jednom toku ne utječu na ostale. Za svaki vezu koristi identifikacijski broj veze to omogućava da veza ostane uspostavljena čak i ako dođe do promjene mreže na nekom od spojenih uređaja veza se ne prekida i ne mora biti ponovno uspostavljena.

2.3.2. UDP-Lite (Lightweight User Datagram Protocol)

UDP Lite je modifikacija standardnog UDP protokola koje dodaje mogućnost djelomično pouzdanog slanja podataka. Poput UDP-a, UDP Lite nije konekcijski orijentiran i nema mehanizme kontrole zagušenja. UDP Lite, za razliku od standardnog UDP-a, dopušta isporuku djelomično oštećenih datagrama umjesto da ih potpuno odbacuje.

UDP Lite sadrži element zaglavlja koje dozvoljava aplikacijama da označe koji dio korisnih podataka zahtijeva zaštitu integriteta, a koji dijelovi ne zahtijevaju. Primarno se koristi kod slanja podataka osjetljivih na kašnjenje gdje je prihvatljiviji djelomičan gubitak podataka umjesto potpunog odbacivanja.

3. Usporedba protokola

U ovom poglavlju ćemo usporediti četiri prethodno opisana protokola. To ćemo učiniti u dva dijela, u prvom dijelu usporedit ćemo protokole TCP i SCTP zbog toga što su oba protokola pouzdana i imaju slično područje primjene. U drugom dijelu usporedit ćemo UDP i DCCP zbog njihove nepouzdanosti i činjenice da oba protokola prioritiziraju brzinu nauštrb pouzdanosti, te kao i kod prethodne usporedbe, imaju slično područje primjene.

3.1. TCP i SCTP

Značajka	TCP	SCTP
Pouzdanost	Pouzdan prijenos	Pouzdan prijenos
Način spajanja	Konekcijski orijentiran	Konekcijski orijentiran
Način prijenosa	"byte-oriented"	"message-oriented"
Kontrola zagušenja	Da, klizeći prozor i ACK	Da, klizeći prozor i ACK
Multistreaming	Ne	Da, više nezavisnih tokova
Multi-homing	Ne	Da, više IP adresa na svakom kraju
Redoslijed isporuke	Zajamčen	Zajamčen(za svaki tok)
Sigurnosni mehanizmi	Nema ugrađene, oslanja se na SSL/TLS	Ugrađena zaštita integriteta i autentikacija
OS podrška	Široko podržan	Ograničeno, potrebni posebni moduli

Tablica 3.1

Oba protokola pružaju pouzdan prijenos podataka u pravilnom redoslijedu, koristeći slične, ali ne i identične mehanizme. TCP i SCTP koriste potvrde primljenih paketa i njihovo ponovno slanje ako su izgubljeni, SCTP ima mogućnost slanja podataka preko više tokova što također pomaže u pouzdanosti. Ako dođe do problema s podatcima u jednom toku, to neće utjecati na ostale tokove.

TCP i SCTP su konekcijski orijentirani protokoli koji uspostavljaju vezu prije slanja podataka. TCP proces uspostavljanja veze obavlja u 3 koraka, dok SCTP taj proces obavlja u 4 koraka.

Ključna razlika ova dva protokola je oblik u kojem jedan i drugi šalju podatke. TCP je tzv.“byte-oriented“ protokol što znači da se skup podataka koji se treba poslati šalje u kontinuiranom toku, bajt po bajt. Strana koja prima podatke mora odlučiti kada je poruka gotova. S druge strane SCTP je tzv. „message-oriented“ protokol, što znači da se skup podataka šalje razdvojen u diskretne dijelove koji imaju jasno definiran početak i kraj. Skup podataka šalje se poruku po poruku.

Protokoli koriste vrlo slične mehanizme kontrole zagušenja i oba osiguravaju da je prijenos podataka pouzdan čak i kad dođe do zagušenja u mreži.

Jedna od najvažnijih razlika između TCP-a i SCTP-a je to što SCTP ima mogućnost „multistreaminga“, tj. slanja podataka kroz više nezavisnih tokova. Ono što razlikuje „multistreaming“ i jednostavno korištenje više TCP veza je činjenica da „multistreaming“ omogućuje više tokova podataka kroz jedna vrata, tj. nije potrebno zauzeti više od jednih vrata za prijenos podataka, dok svaka TCP veza mora imati posebna vrata.

Još jedna razlika između ova dva protokola je mogućnost „multi hominga“ SCTP protokola. Ta mogućnost dozvoljava SCTP vezi da na svakoj strani ima više IP adresa od kojih se jedna aktivno koristi, a ostale djeluju kao rezervne adrese koje se mogu koristiti u slučaju da aktivna adresa više ne funkcionira. Prebacivanje s jedne adrese na drugu može se dogoditi tijekom slanja podataka bez gubitka pouzdanosti.

Kao što smo spomenuli kod značajke pouzdanosti, TCP i SCTP osiguravaju da su svi podatci isporučeni u točnom redoslijedu. To rade vrlo sličnim mehanizmima, međutim SCTP ima prednost u tome što garantira redoslijed isporuke za svaki pojedinačni tok podataka unutar jedne veze.

TCP nema sigurnosnih mehanizama. Sigurnost se najčešće ostvaruje korištenjem SSL/TLS protokola. SCTP ima ugrađene sigurnosne mehanizme kao što su zaštita integriteta podataka i autentifikacija poruka.

TCP ima daleko bolju podršku u različitim operacijskim sustavima, nisu potrebni dodatni

moduli ili naknadne implementacije za njegovo funkcioniranje. S druge strane, SCTP je podržan u manje sustava i često su potrebni vanjski moduli za njegovo ispravno funkcioniranje.

Zaključno, odabir između ova dva protokola ovisi o potrebama korisnika. SCTP nudi većinu mogućnosti TCP-a te ih dodatno proširuje, ali nedostaje mu široke podrške koju uživa TCP. SCTP je pogodan za specijalističke sustave i slučajeve gdje su pouzdanost i sigurnost prvi na listi prioriteta, poput financijskih sustava. TCP s druge strane je pogodniji za općenitu primjenu zbog svoje široke razine podržanosti.

3.2. UDP i DCCP

Značajka	UDP	DCCP
Pouzdanost	Nepouzdan prijenos	Nepouzdan prijenos
Način konekcije	Beskonekcijski	Beskonekcijski
Kontrola zagušenja	Nema ugrađenu kontrolu zagušenja	Ima ugrađene mehanizme kontrole zagušenja
Redoslijed isporuke	Nije zajamčen	Nije zajamčen
Detekcija dupliciranih paketa	Nema	Ima podršku za detekciju dupliciranih paketa
Potvrda primitka	Nema ugrađen mehanizam za potvrdu	Ima opcionalni mehanizam za potvrdu primitka
Podrška u OS-ovima i uređajima	Široko podržan	Ograničena podrška, zahtijeva kernel module
Pogodno za real-time	Da, zbog niske latencije	Da, zbog niske latencije i kontrole zagušenja

Tablica 3.2

UDP i DCCP su oba nepouzdani protokoli koji ne garantiraju dostavu svih podataka na odredišnu adresu, već se fokusiraju na brzinu prijenosa i pravovremenost pristiglih podataka.

UDP je beskonekcijski protokol u punom smislu riječi, nema potrebu uspostavljati vezu prije slanja podataka i ne postoji nikakav proces gašenja te iste veze nakon što su podatci poslani.

S druge strane, DCCP, iako nije strogo konekcijski orijentiran protokol, ima proces uspostavljanja veze, tzv. „3-way handshake“. DCCP implementira ovaj proces zato što mu to omogućuje da ima ugrađenu kontrolu zagušenja, tijekom procesa uspostavljanja veze razmjenjuje se parametri koji su potrebni za efikasno kontroliranje zagušenja mreže.

UDP nema ugrađenu kontrolu zagušenja. Glavna prednost DCCP protokola u odnosu na UDP je upravo ugrađena kontrola zagušenja s pomoću koje može nadoknaditi pakete koji su oštećeni ili izgubljeni zbog mrežnog zagušenja.

Niti jedan od ova dva protokola ne garantira pravilan redoslijed isporuke podataka, međutim DCCP ima određene mehanizme koji pomažu u otkrivanju gubitka podataka i sprječavanju zagušenja.

Kod UDP protokola ne postoji mehanizam za otkrivanje dupliciranih paketa, dok kod DCCP-a taj mehanizam postoji. Aplikacije koje koriste UDP moraju same implementirati način na koji će se brinuti o dupliciranim paketima, ali to nije potrebno ako se koristi DCCP.

UDP nema mogućnost slanja potvrde primitka podataka, već jednostavno šalje zadani skup podataka bez praćenja uspješnosti. DCCP ima tu mogućnost, ali je ona opcionalna.

UDP je široko korišten protokol. DCCP s druge strane je vrlo specijaliziran protokol koji nije podržan na nekim od važnijih operacijskih sustava. DCCP često zahtijeva dodatne module kako bi ispravno funkcionirao što nije slučaj s UDP-om.

Oba protokola su vrlo korisna u real-time primjenama gdje je brzi i pravovremeni prijenos podataka prioritet, te se mali broj izgubljenih podataka može istrpjeti. DCCP ima prednost u ovoj značajki zbog ugrađene kontrole zagušenja koju UDP ne posjeduje.

Primjena UDP je vrlo široka zbog njegove jednostavnosti i brzine, iako DCCP nudi proširene mogućnosti u odnosu na UDP. Problem DCCP-a je vrlo niska razina podržanosti s kojom se malo korisnika želi suočiti, DCCP je podržan na sustavima kao što su Linux i FreeBSD, ali ga ne podržavaju MacOS i Windows koji su u širokoj upotrebi. DCCP pati najviše od manjka redovitog održavanja i razvoja zbog čega ga Linux planira ukinuti s liste podržanih protokola 2025. godine.

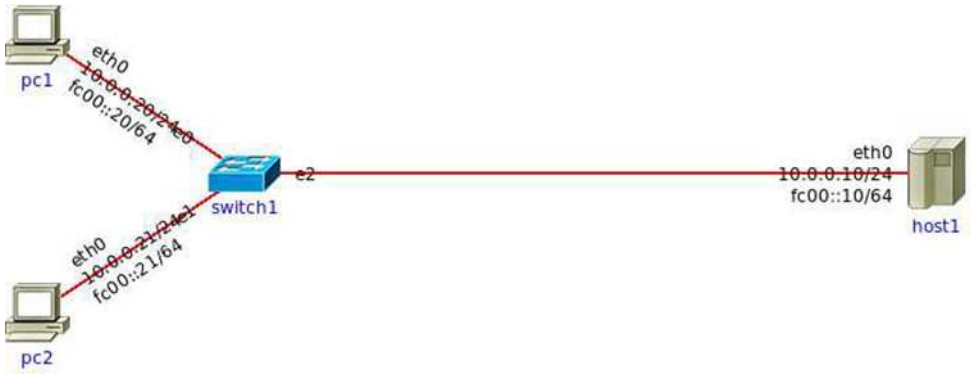
4. Mrežni simulator IMUNES

IMUNES je mrežni simulator, razvijen na Fakultetu elektrotehnike i računarstva u Zagrebu. IMUNES radi na operacijskom sustavu FreeBSD i Linux. Služi za emulaciju i analizu mrežnih topologija. Koristit ćemo IMUNES kako bi demonstrirali rad alternativnih protokola i kako bi ih usporedili s već ustaljenim TCP i UDP protokolima.

4.1. Parametri testiranja

Koristimo IMUNES na virtualnom stroju koji radi na FreeBSD operacijskom sustavu(verzija 12.3). Koristimo alat VirtualBox za pokretanje i podešavanje virtualnog stroja.

Naša testna mreža sastoji se od 2 klijenta (pc1, pc2), mrežnog komutatora i poslužitelja (host1) kao što možete vidjeti na Slici 4.1.



Slika 4.1

U sljedećim slučajevima testirat ćemo slanje podataka i poruka s klijenta na poslužitelj pomoću TCP, UDP i SCTP protokola i usporedit ćemo njihove značajke i performanse. Za testiranje je korišten većinom alat Nmap ncat i netperf koji imaju opcije korištenja TCP, UDP i SCTP protokola.

Iskoristit ćemo još jedan isječak koda [11] kako bi testirali DCCP u IMUNES u nekim mogućim parametrima. Za „kompajliranje“ ovog koda potreban nam je alat zig koji ćemo instalirati naredbama:

- `wget https://mrepro.tel.fer.hr/instaliraj_zig.sh`
- `sudo bash instaliraj_zig.sh`

Sada možemo „kompajlirati“ kod:

- `zig cc -target aarch64-linux-gnu ser.c -o dccp_server`
- `zig cc -target aarch64-linux-gnu cl.c -o dccp_client`

Naše skripte sada možemo poslati na poslužitelj host1 i klijent pc1:

- `hcp dccp_server host1`
- `hcp dccp_client pc1`

Sljedeće postupke ćemo objasniti u poglavlju koje se bavi testiranjem.

4.2. Testiranje

4.2.1. Vrijeme potrebno za uspostavljanje veze

Koristili smo alat Nmap ncat kojim smo uspostavili vezu od klijenta pc1 do poslužitelja host1. S pomoću alata Wireshark pratili smo koliko je vremena bilo potrebno da se veza uspostavi za svaki od testiranih protokola. Za svaki protokol test je napravljen 10 puta i izračunat je prosjek vremena potrebnog za uspostavu veze.

Za DCCP je mjerenje provedeno na identičan način, a testiranje smo obavili ovako:

- Spajamo se na poslužitelj host1 naredbom:
  - `himage host1`
- Pokrećemo naš server na poslužitelju host1:
  - `./dccp_server`
- Spajamo se na klijent pc1, pokrećemo ga i šaljemo poruku na vrata 1200 na našem poslužitelju host1:
  - `himage pc1`
  - `./dccp_client 10.0.0.10 42 1200 test`

Testiranje ostalih protokola pomoću alata Nmap ncat obavljeno je na sljedeći način:

- Poslužitelj host1 postavili smo da sluša na vratima 1200:
  - `ncat -l 10.0.0.10 1200`
  - `ncat -l 10.0.0.10 1200 –sctp`
- Na klijentu pc1 smo uspostavili vezu s poslužiteljem na izabrana vrata:
  - `ncat -l 10.0.0.10 1200`
  - `ncat -l 10.0.0.10 1200 –sctp`

TCP	SCTP	DCCP
0.023 ms	0.092 ms	0.072 ms

Tablica 4.1

4.2.2. Vrijeme potrebno za transfer datoteke

Kao i u prethodnom testu koristimo alat Nmap ncat i pratimo vrijeme slanja s pomoću Wireshark alata. Poslali smo od klijenta pc1 do poslužitelja host1 tri različite datoteke, prva ima 10 MiB, druga 100 MiB, a treća 500 MiB. Radi se o tekstualnim datotekama generiranim korištenjem naredbe dd koju ćemo detaljno raspisati dalje u tekstu. Vrijeme je mjereno nakon što je veza već uspostavljena. Svaki prijenos je izmjeren 10 puta i izračunat je prosjek.

- Na poslužitelju host1 postavili smo vrata na 1200 i napisali ime datoteke u koju će biti zapisana dolazna datoteka:
  - `ncat -l 10.0.0.10 1200 > outputfile`
  - `ncat -l 10.0.0.10 1200 > outputfile --udp`
  - `ncat -l 10.0.0.10 1200 > outputfile --sctp`
- Na klijentu pc1 prvo smo generirali datoteke potrebne za naše testiranje:
  - `dd if=/dev/zero of=inputfile bs=1048576 count=10`
  - `dd if=/dev/zero of=inputfile bs=1048576 count=100`
  - `dd if=/dev/zero of=inputfile bs=1048576 count=500`
- Na klijentu pc1 smo onda poslali datoteku na poslužitelj:
  - `ncat -send-only 10.0.0.10 1200 < inputfile`
  - `ncat -send-only 10.0.0.10 1200 < inputfile -udp`
  - `ncat -send-only 10.0.0.10 1200 < inputfile -sctp`

Veličina datoteke	TCP	UDP	SCTP
10 MiB	0.237 s	0.153 s	0.587 s
100 MiB	4.226 s	3.367 s	3.515 s
500 MiB	16.772 s	17.272 s	21.151 s

Tablica 4.2

Pri veličini datoteke od 10 MiB rezultati su očekivani, UDP je nepouzdan, ali brz protokol. Pri većim veličinama TCP je čak prestigao UDP, a SCTP malo zaostaje.

4.2.3. Pouzdanost

Kod izvođenja prethodnog testa smo također mjerili pouzdanost pojedinog protokola.

Izgubljene pakete mjerili smo alatom Wireshark. Koristili smo dvije instance Wireshark-a, jednu pokrenutu na klijentu, a drugu na poslužitelju. Nakon slanja podataka usporedili smo koliko je paketa poslano, a koliko primljeno.

Veličina datoteke	TCP	UDP	SCTP
10 MiB	0%	0%	0%
100 MiB	0%	4.5%	0%
500 MiB	0%	4.8%	0%

Tablica 4.3

TCP i SCTP su pouzdani protokoli koji sprječavaju gubljenje pakete, kao što vidimo i u testu. UDP očekivano ima neke gubitke zato što nije pouzdan protokol te ako dođe do izgubljenih paketa, UDP ih ne može nadoknaditi. Kao što smo spomenuli, UDP je nepouzdan protokol koji podatke šalje bez provjera ili povratnih informacija o njihovom dolasku do odredišne adrese.

4.2.4. Korištenje računalnih resursa

Tijekom izvođenja testa slanja datoteke mjerili smo opterećenost procesora klijenta i poslužitelja za svaki od testiranih protokola. Koristili smo naredbu : top, i pratili koliko proces ncat opterećuje procesor na našem klijentu i na našem poslužitelju tijekom slanja datoteke. Tijekom svakog slanja datoteke rezultati su zapisani svakih 5 sekundi i izračunat je prosjek. Mjerenje je izvršeno samo kod slanja datoteke od 500 MiB kako bi imali dovoljno vremena za konzistentno mjerenje

	TCP	UDP	SCTP
Klijent	88%	64%	82%

Poslužitelj	45%	16%	38%

Tablica 4.4

Po našim rezultatima možemo vidjeti da TCP i SCTP koriste procesor naših testnih čvorova značajno više nego UDP. Razlog je u tome što je UDP u usporedbi s TCP-om i SCTP-om vrlo jednostavan protokol bez mnogo ugrađenih provjera.

4.2.5. Kašnjenje (Latency)

Koristimo alat Nmap ncat kako bi provjerili kašnjenje u našoj mreži. Šaljemo zahtjev od klijenta pc1 do poslužitelja host1 i mjerimo vrijeme potrebno za odaziv. U ispisu naredbe uzimamo „real“ vrijednost. Svaki test je proveden 10 puta i izračunat je prosječno vrijeme kašnjenja.

Za testiranje DCCP-a koristili smo isti proces kao i kad smo mjerili vrijeme uspostavljanja veze i mjerili smo kašnjenje pomoću alata Wireshark gdje smo gledali koliko je potrebno potvrdi poslužitelju host1 da potvrdi primanje poruke od strane klijenta pc1.

Za ostale protokole testirali smo na sljedeći način:

- Na poslužitelju host1 vrata 1200 su postavljena da slušaju:
  - `ncat -l 10.0.0.10 1200`
  - `ncat -l 10.0.0.10 1200 --udp`
  - `ncat -l 10.0.0.10 1200 --sctp`
- Na klijentu pc1 poslali smo naredbu za mjerenje kašnjenja:
  - `time (echo "test" | ncat 10.0.0.10 1200`
  - `time (echo "test" | ncat 10.0.0.10 1200 --udp`
  - `time (echo "test" | ncat 10.0.0.10 1200 -sctp`
- U terminalu našeg klijenta onda vidimo 3 vrijednosti(real, user i sys), u našem slučaju koristimo brojke real vrijednosti

TCP	UDP	SCTP	DCCP
27 ms	24 ms	27 ms	20 ms

Tablica 4.5

Razlike nisu velike, ali postoje. Očito je da će UDP, kao i DCCP, imati najmanje kašnjenje zbog svog dizajna, ne mora vršiti nikakve dodatne provjere već jednostavno šalje podatke koji su mu zadani. Zato se UDP i koristi npr. u slučaju video igara povezanih na internet.

4.2.6. Propusnost

Koristeći alata netperf testirali smo i propusnost protokola. Rezultati su u Mbit/s.

- Na poslužitelju pokrenuli smo server koji sluša na vratima 1200 naredbom:
  - `netserver -p 1200`
- Na klijentu smo pokrenuli test naredbom:
  - `netperf -H 10.0.0.10 -p 1200 -l 30 -- -m 1024`
  - `netperf -H 10.0.0.10 -p 1200 -l 30 -t UDP_STREAM -- -m 1024`
  - `netperf -H 10.0.0.10 -p 1200 -l 30 -t SCTP_STREAM -- -m 1024`

	TCP	UDP	SCTP
Propusnost	1422.4 Mbit/s	2236.1 Mbit/s	1374.7 Mbit/s

Tablica 4.6

Kao što smo mogli vidjeti i sa testom slanja datoteke, UDP je najbrži protokol zahvaljujući svojoj jednostavnosti.

4.2.7. Testovi pod uvjetima zagušenja mreže

Neke od testova izveli smo ponovo dok smo simulirali zagušenje naše testne mreže. Zagušenje smo simulirali na sljedeći način:

- Na poveznici između komutatora i poslužitelja dodano je kašnjenje od 10 mikro sekundi
- Na poslužitelju su pokrenute dvije inačice servera, na vratima 1200 i 1300:
  - `netserver -p 1200`
  - `netserver -p 1300`
- Sa klijenta pc2 pokrenuto je slanje poruka koje je trajalo dok su se izvodili svi testovi:
  - `netperf -H 10.0.0.10 -p 1300 -l 60 -- -m 1024`

Kašnjenje	TCP	UDP	SCTP
Vrijeme	62 ms	55ms	67 ms

Tablica 4.7

Transfer datoteke	TCP	UDP	SCTP
10 MiB	0.823 s	0.433 s	1.382 s
100 MiB	9.188 s	5.261 s	9.765 s
500 MiB	35.575 s	26.744 s	38.966 s

Tablica 4.8

Pouzdanost	TCP	UDP	SCTP
10 MiB	0%	0.2%	0%
100 MiB	0%	10.6%	0%
500 MiB	0%	14.5%	0%

Tablica 4.9

Na rezultatima vidimo da je kod svih testiranih protokola došlo do usporavanja u brzini prijenosa podataka i povećanje vrijeme kašnjenja, ali kod UDP je ono najmanje. Kod TCP-a se vrijeme potrebno za transfer datoteke od 500 MiB povećalo 2.12 puta, kod SCTP-a 1.94 puta , a kod UDP-a 1.54 puta što je značajno manje nego kod ostala dva protokola. Ono gdje se UDP protokol muči je pouzdanost u prijenosu podataka. Kod TCP-a i SCTP-a nema gubitka paketa unatoč zagušenju mreže, dok kod UDP imamo gubitak paketa kod svih veličina datoteke koja se prenosila, i značajno povećanje gubitka paketa u odnosu na mrežu koja nije zagušena. Ovo se događa zato što UDP nema mehanizama pomoću kojih se može nositi s zagušenjem mreže.

Zaključak

U ovom radu istražili smo protokole transportnog sloja, fokusirajući se na alternativne protokole kao što su SCTP i DCCP. Analizom smo pokazali da svaki alternativni protokol ima svojih prednosti, ali isto tako i nedostataka. Navedeni protokoli imaju svoju primjenu, neki širu, a neki užu, ali u ovom trenutku se ne čini kao da će ubrzo moći zamijeniti ustaljene i dobro podržane protokole transportnog sloja kao što su TCP i UDP.

TCP je protokol koji je razvijen davno, bar u smislu razvoja internetski protokola, i samim time je dobro istražen i optimiziran. Uživa široku podršku na raznim operacijskim sustavima te je jednostavan za implementaciju. Ima detaljnu dokumentaciju i godine iskustva milijuna korisnika. Nudi dovoljnu razinu pouzdanosti i brzine za većinu korisnika koji trebaju pouzdan protokol. Slično vrijedi i za protokol UDP, jednostavan i dobro podržan protokol koji zadovoljava potrebe većine korisnika kojima je potreban brz i jednostavan protokol nauštrb pouzdanosti.

SCTP i DCCP s druge strane nude proširene mogućnosti u odnosu na TCP i UDP, ali su protokoli bez velike količine podrške i iskustva koji uživaju TCP i UDP. Oba protokola su našla svoje specifične upotrebe, SCTP kao vrlo siguran i stabilan protokol koji se koristi kada su primarni ciljevi pouzdanost i sigurnost, DCCP kao brzi protokol s ugrađenom mogućnošću kontrole zagušenja koji olakšava dizajn aplikacija bez potrebe za odvojenom implementacijom kontrole zagušenja.

Zaključno, možemo reći kako je SCTP više prihvaćen protokol od DCCP-a, te daljnji razvoj i istraživanje može doprinijeti poboljšanju mrežnih aplikacija. S druge strane, DCCP nikada nije prihvaćen u značajnu upotrebu i čini se da gubi ono malo podrške koje je imao kada je razvijen. DCCP sigurno nudi neke prednosti nad UDP protokolom te daljnji razvoj i istraživanje može osigurati širu primjenu nego trenutno.

Literatura

[1] Postel, J. Transmission Control Protocol. RFC 793. IETF, 1981.

[2] Stewart, R. Stream Control Transmission Protocol (SCTP). RFC 2960. IETF, 2000.

[3] Kohler, E., Handley, M., Floyd, S. Datagram Congestion Control Protocol (DCCP). RFC 4340. IETF, 2006.

[4] Postel, J. B. User Datagram Protocol. RFC 768. IETF, 1980.

[5] Larzon, L. Å., Degermark, M., Pink, S., Jonsson, L., Fairhurst, G. The Lightweight User Datagram Protocol (UDP-Lite). RFC 3828. IETF, 2004.

[6] Thomson, M., Pauly, T., Eddy, W., Weston, M. QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000. IETF, 2021.

[7] Metz, C. Host-to-Host Protocol for the ARPA Network. RFC 908. IETF, 1984.

[8] IMUNES: User Guide. Poveznica: https://imunes.net/guide/; pristupljeno 12. lipnja 2024.

[9] Ncat Users' Guide. Poveznica: https://nmap.org/ncat/guide/; pristupljeno 12. lipnja 2024.

[10] Github: zonque. Poveznica: https://gist.github.com/zonque/7361965a20b3b4cecf19; pristupljeno 13. lipnja 2024.

[11] Anmol Sarma. Poveznica: https://www.anmolsarma.in/post/dccp/; pristupljeno 3. rujna 2024.

[12] Stewart, R., Stream Control Transmission Protocol. RFC 4960. IETF, 2007.

[13] Cloudflare. Poveznica: https://www.cloudflare.com/learning/ddos/syn-flood-ddos-attack/; pristupljeno 25. kolovoza 2024.

Sažetak

Ovaj rad istražuje prednosti, nedostatke i razlike alternativnih protokola transportnog sloja kao što su DCCP, SCTP, QUIC, RDP i UDP-Lite u usporedbi s standardnim i dugo korištenim protokolima TCP i UDP. Rad se fokusira primarno na protokole SCTP i DCCP. Demonstriranje i testiranje obavljeno je u mrežnom simulatoru IMUNES. Korišteni su alati Wireshark i Nmap ncat. Rad analizira ove protokole i daje prijedlog kada i gdje su primjenjivi u ovisnosti o njihovim značajkama i performansama na testiranju.

Ključne riječi: TCP, UDP, SCTP, DCCP, QUIC, UDP- Lite

Summary

This paper explores the advantages, disadvantages, and differences of alternative transport layer protocols such as DCCP, SCTP, QUIC, RDP, and UDP-Lite compared to the standard and relatively old TCP and UDP protocols. The focus of the paper is primarily on SCTP and DCCP protocols. Demonstration and testing were conducted using the network simulator IMUNES. Tools that were used include Wireshark and Nmap ncat. The paper analyzes these protocols and provides recommendations on when and where they can be used based on their features and performance in testing.

Keywords: TCP, UDP, SCTP, DCCP, QUIC, UDP- Lite