Razvoj simulatora prijenosa paketa računalne mreže

Posavec, Mateo

SVEUČILIŠTE U ZAGREBU

FAKULTET PROMETNIH ZNANOSTI

Mateo Posavec

RAZVOJ SIMULATORA PRIJENOSA PAKETA RAČUNALNE MREŽE

ZAVRŠNI RAD

Zagreb, 2018.

SVEUČILIŠTE U ZAGREBU
FAKULTET PROMETNIH ZNANOSTI
ODBOR ZA ZAVRŠNI RAD

Zagreb, 5. travnja 2018.

Zavod: **Zavod za informacijsko komunikacijski promet**
Predmet: **Računalne mreže**

ZAVRŠNI ZADATAK br. 4859

Pristupnik: **Mateo Posavec (0036470394)**
Studij: Promet
Smjer: Informacijsko-komunikacijski promet

Zadatak: **Razvoj simulatora prijenosa paketa računalne mreže**

Opis zadatka:

U završnom radu opisat će se karakteristike TCP/IP složaja. Objasniti značaj mrežnih simulatora i odabir programskog jezika C#. Dizajnirat će se simulator prijenosa paketa računalne mreže. Testirat će se funkcionalnost izrađenog simulatora prijenosa paketa računalne mreže. Navest će se primjeri simulacije u dizajniranom i izrađenom simulatoru.

Mentor:

doc. dr. sc. Ivan Grgurević

Predsjednik povjerenstva za završni ispit:

Sveučilište u Zagrebu

Fakultet prometnih znanosti

ZAVRŠNI RAD

RAZVOJ SIMULATORA PRIJENOSA PAKETA RAČUNALNE MREŽE

DEVELOPMENT OF COMPUTER NETWORK PACKET TRANSMISSION SIMULATOR

Mentor: doc. dr. sc. Ivan Grgurević

Student: Mateo Posavec, 0036470394

Zagreb, rujan, 2018.

SAŽETAK

Završni rad zamišljen je i izrađen u dva povezana dijela. Jedan dio je tekstualni koji osim što izlaže znanja iz područja računalnih mreža, služi i kao dokumentacija drugom dijelu rada, razvijenom simulatoru za prijenos paketa računalne mreže. Ta dva dijela zajedno tvore cjeloviti završni rad. Simulator je baziran na TCP/IP modelu mreže, koji je oslonac rada najveće svjetske mreže, Interneta. Rasprostranjenost Interneta i njegovo svakodnevno korištenje od strane više milijardi korisnika, razlog su odabiru ove teme i stvaranju simulatora koji ima za cilj prosječnom korisniku približiti način rada računalnih mreža s razine paketa, odnosno podatkovnih jedinica koje je prenose mrežom. Simulator je zamišljen kao programski alat kojeg je moguće koristiti pri edukaciji različitih skupina korisnika. Kao programski jezik, za razvoj simulatora, odabran je C# zbog jednostavnosti rada s grafičkim elementima i velike popularnosti i zastupljenosti na tržištu.

Ključne riječi: simulator, TCP/IP model, C# programski jezik, edukacija

SUMMARY

This thesis is designed in two parts. One of them is the text document that conveys knowledge from the field of computer networking, and in the same time serves as a documentation for the other part of this thesis, the developed computer network packet transmission simulator. These two integral parts create a complete bachelor thesis. The simulator is based on TCP/IP network model, which is the backbone of the world's biggest network, the Internet. The sheer reach of the Internet, and it's daily use from billions of users were the main motive for choosing this perticular theme, and the creation of this simulator that has, as it's main goal, the purpose of acquainting an average user to the way computer networks function from the perspective of a packet, or a data unit that is sent through the network. The simulator is designed as a software that can be used as an educational tool for teaching a variety of different users (regarding their previous knowledge). The programing language chosen and used for the development of the simulator is C#. It is used besause of it's simplicity while working with graphical elements, and because of it's popularity and it's market presence.

Key Words: simualator, TCP/IP model, C# programming language, education

SADRŽAJ

1.UVOD

Svakodnevna aktivnost velikog broja ljudi je pristupanje Internetu, bilo zbog potrebe za pronalaskom odgovora na traženo pitanje, željom da se proširi znanje o nekoj temi ili komunicira s određenom osobom. Naizgled različiti ciljevi u stvarnosti su implementirani na gotovo isti način, a to je prijenos informacije, odnosno podataka putem mreže. Potpuno razumijevanje takvog procesa iziskuje razumijevanje mnogih grana znanosti: matematike, fizike, elektrotehnike, nauke o materijalima, računalne znanosti, teorije informacija i mnogih drugih.

Prethodno navedeno može se činiti zastrašujuće i nedostižno, te se iz tog razloga mnogi korisnici Interneta ne žele upustiti u shvaćanje načina na koji on radi. Potpuno znanje iz tog područja je uistinu opsežno, no to ne znači da nije potrebno shvatiti barem neke od osnovnih funkcionalnosti mreža, a time i Interneta. Upravo je to cilj, odnosno motiv stvaranja ovog simulatora i prateće dokumentacije. Pokušalo se na što jednostavniji način predstaviti prijenos podataka mrežom, uz primjenu pojednostavljenih prikaza stvarnih uređaja i medija za prijenos, dakle prikaz na dosta visokoj razini apstrakcije. Zbog toga se rad može koristiti pri učenju,odnosno pri edukaciji kako samog sebe, tako i drugih. Prema tome, cilj završnog rada je razvoj simulatora prijenosa paketa računalne mreže korištenjem programske podrške C#. Svrha završnog rada je prikazati prijenos podataka računalnom mrežom za potrebe boljeg razumijevanja načina prijenosa.

Završni rad sastoji se od dokumentacijskog (tekst završnog rada) i praktičnog dijela (izrađen simulator). Dokumentacijski dio završnog rada podijeljen je u osam poglavlja/teza:

1.Uvod

2. Karakteristike TCP/IP složaja

3. Značaj mrežnih simulatora

4. Opis programskog jezika C#

5. Dizajn simulatora prijenosa paketa računalne mreže

6. Funkcionalnost simulatora prijenosa paketa računalne mreže

7. Primjeri simulacije u dizajniranom i izrađenom simulatoru

8. Zaključak

Uvodno poglavlje daje osnovnu sliku o radu te definira cilj i strukturu rada.

Unutar drugog poglavlja predstavit će se značajke TCP/IP složaja, no ne u detalje, nego onoliko koliko je potrebno za razumijevanje rada simulatora. Objasnit će se protokoli korišteni na pojedinim slojevima s naglaskom na one koji su implementirani unutar samog simulatora.

Treće poglavlje dati će kratak opis pojma simulatora i simulacije, te će se nabrojati glavne funkcionalnosti simulatora danas u upotrebi.

U četvrtom poglavlju ukratko će se opisati korišteni programski jezik te će se navesti najnovije verzije kako programskog jezika, tako i razvojnog okruženja u trenutku pisanja ovog rada.

Peto poglavlje sadrži opis simulatora s razine programskog jezika. Nabrojani su korišteni elementi, objašnjene novostvorene metode i opisan je unutarnji rad simulatora, dakle ono što korisnik ne vidi.

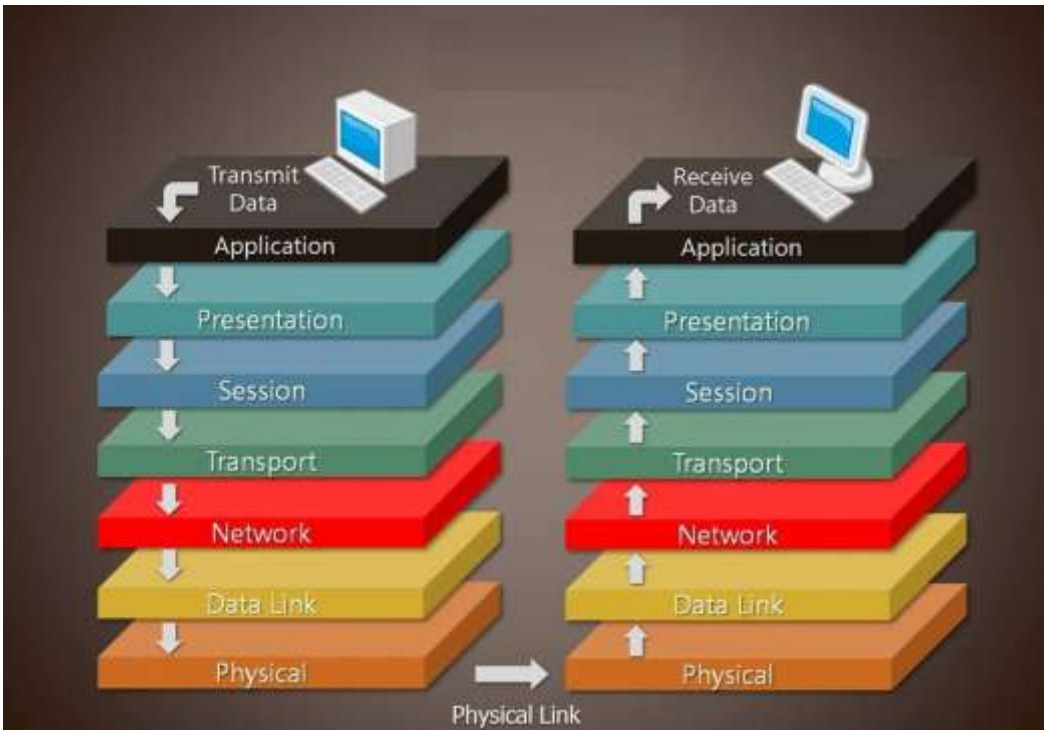
U šestom poglavlju opisane su najvažnije funkcije simulatora te su objašnjeni dijelovi prozora koje korisnik može mijenjati u svrhu promijene trenutno pokrenute simulacije.

Sedmo poglavlje predstavlja primjere korištenja simulatora, objašnjene tekstom i popraćene slikama.

Osmo poglavlje je zaključak u kojem se daje osvrt na cjelokupni završni rad.

2. KARAKTERISTIKE TCP/IP SLOŽAJA

Pri govoru o bilo kakvoj računalnoj mreži, što je slučaj u gotovo svoj literaturi vezanoj za to područje, započinje se s opisom OSI (engl. *Open System Interconnection*) modela, kojeg se smatra, ali i zove, referentnim modelom. Radi se o modelu koji nema primjenu u stvarnom svijetu, odnosno niti jedna mrežna arhitektura nije dizajnirana po njemu, ali većina mreža korištenih u prošlosti, pa tako i one danas u upotrebi, koristile su OSI model kao svoj nacrt, odnosno plan. Slika 1. prikazuje slojeve OSI modela kojih je ukupno sedam.



Slika 1. OSI referentni model [1]

Osim samog naziva slojeva važno je primijetiti kako je kod komunikacije između dva uređaja potrebno prijeći kroz sve slojeve na obje strane i uz to je potrebno prolaziti kroz njih određenim redoslijedom koji je na slici 1. označen bijelom strelicom. Dakle na razinu N je moguće doći isključivo s razine N-1 ili N+1, ovisno o smjeru kretanja komunikacije, odnosno radi li se o predajnoj ili prijemnoj strani.

Nazivi slojeva koje je moguće pronaći u domaćoj literaturi su sljedeći:

- Aplikacijski sloj –sloj 7
- Prezentacijski sloj – sloj 6
- Sesijski sloj - sloj 5
- Transportni sloj – sloj 4
- Mrežni sloj – sloj 3
- Podatkovni sloj – sloj 2
- Fizički sloj – sloj 1

Aplikacijski sloj najbliži je korisniku i posebnost mu je što nema sloja iznad sebe, dakle on je sloj koji stvara informaciju[1], odnosno obrađuje informaciju[2]. Zbog imena ovog sloja često se neispravno zaključuje kako u ovaj sloj spadaju aplikacije poput pretraživača interneta i sl. To nije točno, jer takve aplikacije prelaze okvire samog OSI modela. Ovaj sloj uspostavlja komunikaciju s korisnikom i s operativnim sustavom računala i služi provjeri identiteta uređaja koji komuniciraju kao i provjeri raspoloživog kapaciteta i sl.

Prezentacijski sloj olakšava komunikaciju aplikacijskih slojeva na prijamnoj i predajnoj strani ukoliko postoji razlika u „jeziku“ tih dviju strana, odnosno može se reći kako ovaj sloj služi kao prevodilac.

Sesijski sloj upravlja sesijom, odnosno konekcijom između dva krajnja računala. On ju uspostavlja, njome koordinira i raskida.

Transportni sloj pruža, odnosno obavlja, nekoliko različitih funkcija: omogućava raspodjelu podatke u sekvence različitih dimenzija, osigurava ispravnost podataka, održava kvalitetu komunikacije, osigurava siguran i pouzdan prijenos podataka preko mreže, obavještava drugu stranu što je do sada primljeno i slično.

[1] Ne prima informaciju od sloja iznad sebe na predajnoj strani.
[2] Ne šalje informaciju sloju iznad sebe na prijemnoj strani.

Mrežni sloj omogućava prijenos podataka, veoma često nazivanih paketi (barem na ovom sloju), kroz mrežu od predajne do prijemne strane, preko čvorova koji se nalaze na tom putu. Omogućava i razdjeljivanje paketa u manje pakete postupkom koji se zove fragmentacija i ponovno ih sastavlja pri prijemu. Najvažnija zadaća sloja, od do sada nabrojanih, je usmjeravanje paketa kroz mrežu uz pomoć adrese odredišta koju bi trebalo posjedovati svako računalo u mreži.

Podatkovni sloj, za razliku od mrežnog sloja[3], omogućuje prijenos podataka između dva čvora u mreži, preciznije rečeno između dva susjedna čvora u mreži, te provjerava ispravnost poslanih podataka.

Fizički sloj zaslužan je za prijenos podataka preko prijenosnog medija, bio on zrak, metal ili svjetlovod. Vodi se briga o voltaži, broju pinova, vrsti medija, odnosno definiraju se sva električna i fizička svojstva mrežnih uređaja.

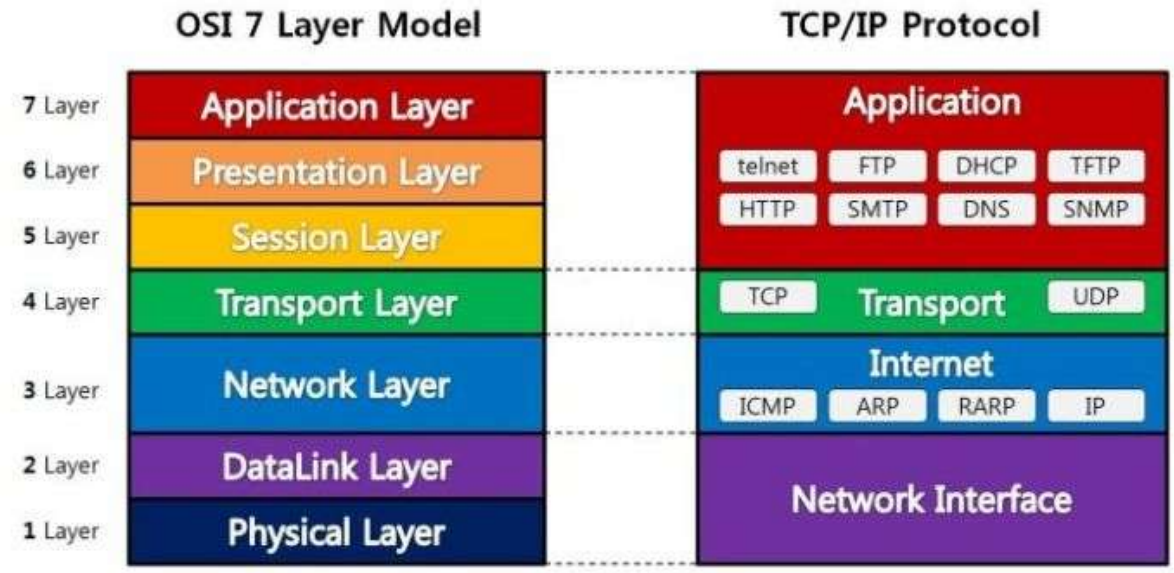
Svi slojevi OSI modela do sada opisani imaju više karakteristika nego što je navedeno, no objašnjene su najvažnije funkcije svih slojeva koje se pojavljuju u drugom modela mrežne arhitekture, tj. one funkcije koje se koriste u TCP/IP modelu.

TCP/IP (engl. *Transmission Control Protocol/Internet Protocol*) je naziv modela koji većina današnjih uređaja povezana na globalnu mrežu Internet koriste. Moguće je u domaćoj literaturi naći i naziv „složaj“ umjesto „model“, budući se radi o arhitekturi koja ima svoju primjenu u stvarnom svijetu. Ime ovog složaja dolazi od dva glavna protokola koja su njegov sastavni dio, a to su TCP i IP. Iako postoji još jedan veoma često korišten protokol (umjesto TCP-a) UDP (engl. *User Datagram Protocol*) ime složaja, pri njegovom korištenju, se ne mijenja.

Ne ulazeći duboko u povijest i razvoj ovog modela, valja napomenuti kako je američka obrambena agencija DARPA (engl. *Defence Advanced Research Projects Agency*) tokom 70-ih godina prošlog stoljeća razvijala mrežu naziva ARPANET, koja je kasnije prerasla u ono što danas zovemo Internetom. Uz razvoj ARPANETA razvijao se i TCP/IP složaj, te je to i razlog današnjeg gotovo isključivog korištenja tog složaja kod pristupa Internet-u.

[3] Mrežni sloj se brine za poveznicu (engl. *connection*) između dva krajnja uređaja u mreži, te ostvaruje prijenos podataka između njih.

TCP/IP slojevi prikazani su na slici 2. usporedno s OSI slojevima.



Slika 2. TCP/IP i OSI modeli [2]

Na slici se vidi kako TCP/IP složaj ima manje slojeva naspram OSI modela, ali valja napomenuti kako se unutar literature iz ovog područja ponekad može pronaći i TCP/IP složaj s pet (u odnosu na četiri) sloja, gdje je zadnji sloj podijeljen na dva. Nazivi slojeva koji se mogu pronaći u domaćoj literaturi su sljedeći:

- Aplikacijski sloj – sloj 4

- Transportni sloj – sloj 3

- Internet sloj – sloj 2

- Sloj mrežnog pristupa – sloj 1

Osim naziva slojeva moguće je vidjeti (na slici 2.) i nazive (skraćenice) najčešće korištenih protokola vezanih za pojedine slojeve, od kojih će neki biti opširnije opisani unutar ovog rada.

Aplikacijski sloj, odnosno sloj 4, najviši je sloj TCP/IP složaja i kao što je vidljivo na slici 2. on „pokriva“ područje gornja tri sloja OSI modela. Sve prethodno navedene funkcije za gornja tri sloja OSI modela implementirani su u ovom sloju, te se neće ponovno navoditi. Postoji poveći broj protokola korištenih na ovom sloju, a

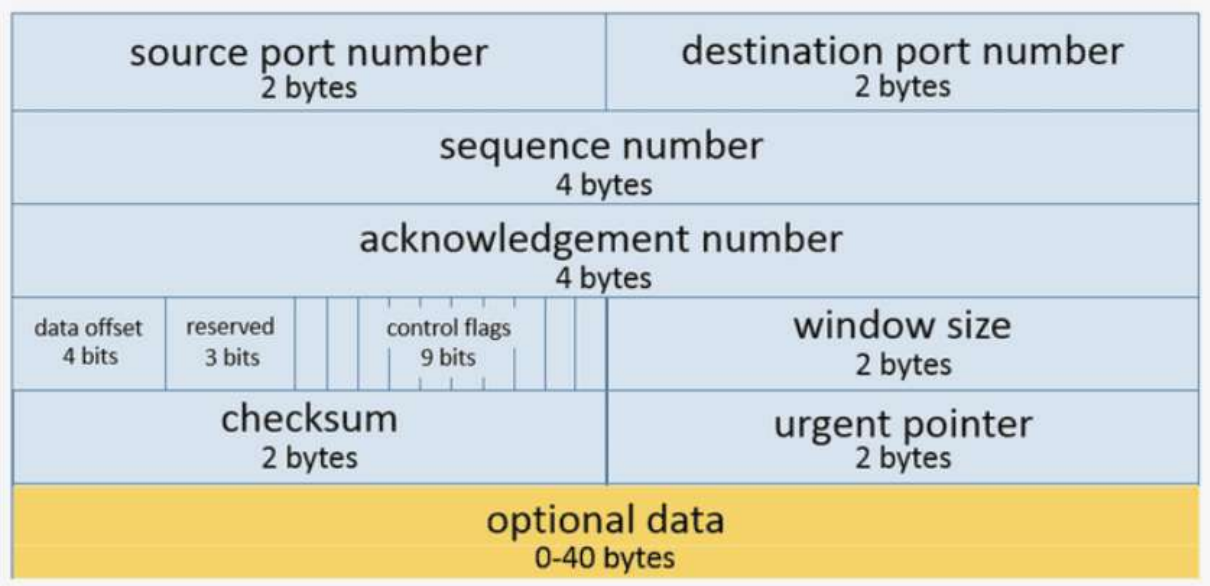
neki su i navedeni na slici 2., a to su DNS (engl. *Domain Name System*), HTTP (engl. *HyperText Transfer Protocol*), DHCP (engl. *Dynamic Host Configuration Protocol*), FTP (engl. *File Transfer Protocol*), SMTP (engl. *Simple Mail Transfer Protocol*). Od protokola koji nisu navedeni na slici valja spomenuti HTTPS (engl. *HyperText Transfer Protocol Secure*), POP3 (eng. *Post Office Protocol ver.3*) i IMAP (engl. *Internet Message Access Protocol*). DNS i DHCP su protokoli koji olakšavaju korištenje mreže prosječnom korisniku, budući korisnik ne mora pamtiti adrese računala (servera) na kojima se nalazi sadržaj kojemu namjerava pristupiti nego može upisati „ime“ stranice koje je mnogo lakše zapamtiti (DNS), i uz to se ne mora brinuti za koliziju adrese svoga računala s ostalim računalima u mreži nego to prepušta samom računalu (DHCP). SMTP, POP3 i IMAP su protokoli koji omogućuju slanje i primanje elektroničke pošte, SMTP za slanje, a POP3 i IMAP za primanje poruka. FTP je protokol korišten za prijenos sadržaja preko Interneta, no danas je u velikoj mjeri zamijenjen HTTP i HTTPS-om. HTTP je protokol originalno zamišljen za predstavljanje sadržaja na Internet stranicama, u vidu teksta, no tokom razvoja omogućen je prikaz slika, multimedije, pa i prijenos velikih datoteka različitog tipa (za što se nekada koristio FTP). HTTPS je nastavak na HTTP uz velike promjene u vidu sigurnosti i tajnosti podataka.

Transportni sloj, sloj 3, nazivan i prijenosni sloj omogućava, odnosno osigurava prijenos cjelokupne informacije od predajne do prijemne strane. Omogućava pouzdani i nepouzdani prijenos, ovisno o protokolu koji je korišten. Ukoliko se koristi TCP protokol omogućava se pouzdani prijenos, a ako se koristi UDP protokol nepouzdani prijenos. Ta dva protokola su najvažniji protokoli ovog sloja. Ukoliko je korišten UDP nema potrebe za retransmisijom neispravno primljenih podataka[4] ili podataka koji uopće nisu primljeni, budući se ovaj protokol koristi u slučajevima kada je informacija koja se šalje vremenski kritična, odnosno gubi na važnosti ukoliko dođe do kašnjenja. Dakle bolje je primiti informaciju s neispravnostima, nego ju primiti s kašnjenjem. Za razliku od UDP-a TCP se koristi kada je ispravnost informacije od kritične važnosti i kada nije problem kašnjenje sadržaja. Zbog toga je TCP i češće korišten protokol kod svakodnevnog pretraživanja Interneta, gdje je korisnik navikao pričekati određeno vrijeme dok se stranica ne „učita“. Točnost, odnosno ispravnost primljene informacije provjerava se

[4] Podatci izmijenjeni uslijed djelovanja smetnji na prijenosnom putu.

matematičkim radnjama nad datagramom (naziv paketa na sloj 3) i ukoliko se pokaže neispravnim, odbacuje se i traži retransmisija. Ako je primljeni datagram ispravan onda se njegov informacijski sadržaj pohranjuje u računalo, a predajnoj strani se šalje potvrda o uspješnom prijemu informacije tzv. ACK (engl. Acknowledgment) paketom. Ovisno o implementaciji moguće je potvrdu prijema tražiti (odnosno slati, ovisno s koje se strane promatra) nakon svakog poslanog datagrama ili nakon nekoliko poslanih datagrama.

Na slici 3. prikazan je izgled zaglavlja TCP protokola, odnosno polja koja se dodaju na postojeće podatke došle s aplikacijskog sloja.



Slika 3. TCP zaglavlje [3]

Prva dva polja predstavljaju izvorišni i odredišni *port*. Radi se o broju veličine 2 okteta, dakle broju u rasponu od 0 do 2^{16}-1. Upravo preko broja porta računala znaju o kakvoj se konekciji radi i ukoliko ista dva računala imaju više istovremeno ostvarenih konekcija onda ih je moguće razlikovati po broju *port*-a (svaka nova konekcija na predajnom računalu dobiva nezavisni broj *port*-a). Poslužiteljska strana (engl. *server*) ima posebno određene port-ove koje registrira i održava organizacija IANA (engl. *Internet Assigned Numbers Authority).* Najpoznatiji su *port* 25 za SMTP, *port* 53 za DNS, *port* 80 za HTTP protokol, *port* 110 za POP3, *port* 443 za HTTPS.

Port-ovi 0 pa do 1023 poznati su pod nazivom *Well-Known Ports* (Dobro poznati *port*-ovi). Od 1024 pa do 49151 nalaze se *port*-ovi koje je moguće registrirati preko organizacije IANA. Od 49152, pa sve do 65535 su *port*-ovi koji se koriste u privatne svrhe, odnosno dinamički *port*-ovi. Sljedeća dva polja su *sequence* i *acknowledgment number*. To su polja pomoću kojih se drugom računalu u komunikaciji dostavljaju informacije vezane za podatke koje se šalju i one koji su do tada zaprimljeni. Oba polja su veličine 4 okteta odnosno 32 bita, dakle imaju raspon od 0 do 2^{32}-1, tj. od 0b (*bit*-a) do 4GB (*gigabit*-a). Unutar *Sequence* polja upisuje se redni broj kojeg ima prvi oktet podataka unutar datagrama u odnosu na datoteku koja se šalje ili u odnosu na do tada prenesene podatke. Ukoliko dođe do prijenosa dva ista datagrama prijemna strana će prema tom podatku znati da drugi datagram može odbaciti. Polje *Acknowledgment number* predstavlja redni broj sljedećeg okteta koje je računalo spremno primiti u novom datagramu, dakle to je broj kojim se potvrđuje prijem paketa, kako bi se komunikacija mogla nastaviti i kako predajna strana ne bi ponovno slala već poslani paket. Polje *Data Offset* često se naziva i *header length* (duljina zaglavlja), a predstavlja duljinu zaglavlja TCP datagrama, ali ne u oktetnoj vrijednosti nego u 4-oktetnoj vrijednosti. Dakle, za osnovno zaglavlje (bez *Options* polja) taj broj je jednak pet. Sljedeće polje *Reserved*, kako mu i samo ime govori, rezervirano je za posebne svrhe koje nisu vidjele veliku upotrebu u svakodnevici, te je standardno postavljeno na 0. Sljedeće polje je polje zastavica, njih ukupno devet od kojih su najvažnije zastavica ACK (5. zastavica) koja ukazuje da je polje *acknowledgment number* tog paketa relevantno, zastavica SYN (8. zastavica) koja je postavljena samo kod slanja prvoga paketa u komunikaciji kako bi se prijemnoj stani dalo do znanja da se želi uspostaviti komunikacija. Zadnja zastavica (9. zastavica) je zastavica FIN koja se postavlja samo kod zadnjeg paketa. *Windows size* je polje koje definira najveći mogući broj okteta memorije koja računalo ima slobodno za danu konekciju i često će se mijenjati sa svakim paketom, pogotovo kod osobnih računala koji imaju dosta ograničen TCP memorijski spremnik. *Checksum* je polje koje štiti datagram, odnosno omogućava provjeru ispravnosti datagrama. Štiti se zaglavlje datagrama, podatci unutar njega, ali i zaglavlje nižeg sloja, točnije polja *Source* IP *Address*, *Destination* IP *Address* i *protocol number* (sva tri polja će biti objašnjena kasnije u radu). Kod izračuna tog broja svi se štićeni podatci razdjeljuju u 16 bitne vrijednosti (2 okteta), te se, započevši od prvih 16 bita, nadodaju novonastalom broju (iz prethodnog koraka) koristeći logičku operaciju „isključivo ILI“. U konačnici se tako nastao broj

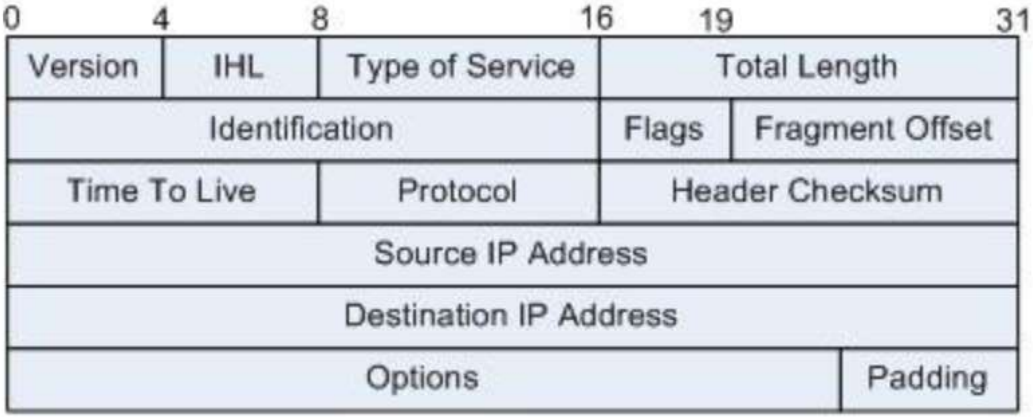
komplementira (jedinični komplement) i zapisuje u polje *Checksum*. Valja napomenuti kako je kod toga izračuna na predajnoj strani u polje *Checksum* prvotno zapisana vrijednost 0. Na predajnoj strani se ponovno ponavlja identičan postupak i tako izračunat broj treba iznositi 0. Ukoliko iznosi, datagram je ispravan. *Urgent pointer* je polje koje se gleda ukoliko je zastavica URG postavljena, no neće se detaljnije opisivati u ovom radu. Polje *Options* je polje koje ne mora postojati u TCP datagram, ali može ovisno o potrebama i također se neće detaljnije opisivati. Valja napomenuti kako njegova duljina mora biti djeljiva s 32 (ukoliko se promatra duljinu u bitovima) i da mu je najveća moguća duljina 40 okteta (320 bita). Nakon svih prethodno objašnjenih polja dolaze podatci koje je sloju tri dostavio sloj četiri.

Internet sloj, sloj 2, često nazivan mrežni sloj, kao i transportni ima ulogu prijenosa od kraja do kraja, ali u slučaju ovoga sloja taj prijenos nije vezan za cjelokupnu informaciju nego jedan paket. Najvažnija zadaća ovoga sloja je usmjeravanje (engl. *routing*). To je proces pronalaženja puta od predajne do prijemne strane kroz mrežu koja se nalazi između njih. Ti putovi nisu fiksni i s vremenom se mijenjaju, ovisno o stanju u mreži ili njenim pojedinim dijelovima. Uređaji koji rade na ovom sloju nazivaju se usmjernici (engl. *router*). Kada paket dođe do jednog usmjernika donosi se odluka o sljedećoj destinaciji njegove rute. Ta odluka može biti odabrana po više kriterija kao što su: broj skokova, zagušenje na linku (poveznici između dva uređaja u mreži), raspoloživi kapacitet, vjerojatnost pogrešaka i dr. Sukladno kriteriju kojeg usmjernik koristi postoje i različiti protokoli za stvaranje „slike“ mreže, a najpoznatiji su RIP (engl. *Routing Information Protocol*), OSPF(engl. *Open Shortest Path First*), BGP (engl. *Border Gateway Protocol*) i dr. Kod protokola usmjeravanja postoji i podjela na vanjske i unutarnje[5], gdje se vanjski koriste za usmjeravanje između autonomnih sustava, a unutarnji unutar njih. Autonomni sustav je dio mreže koji je pod jurisdikcijom neke veće organizacije, države ili njenog predstavnika. Budući protokoli umjeravanja nisu veoma bitni za ovaj rad dalje se neće objašnjavati.

Najvažniji protokol ovog sloja je IP koji je ujedno i sastavni dio imena TCP/IP složaja. Radi se o protokolu koji omogućava adresiranje svakog uređaja spojenog na Internet, uz pomoć IP adrese, koja treba biti jedinstvena za svakog pojedinog

[5] BGP je vanjski protokol, dok su RIP i OSPF unutarnji protokoli.

korisnika. Važna karakteristika ovog protokola jest to da on ne osigurava niti provjerava dali je paket uspješno došao na prijemnu stranu, te ga se zato naziva nepouzdani protokol. U današnje vrijeme u upotrebi su dvije različite verzije IP protokola, a to su verzija 4 IPv4 i verzija 6 IPv6, uz naglasak da IPv4 i dalje prevladava u stvarnom svijetu. Nastoji se u budućnosti potpuno prijeći na IPv6 budući postoji ograničenje verzije 4 vezano za broj adresa koje je moguće dodijeliti uređajima. Druge razlike su odbacivanje nepotrebnih polja u IPv4 zaglavlju, odnosno pojednostavljenje samog zaglavlja. Broj uređaja koje je moguće adresirati s IPv4 iznosi 2^{32}, dok je kod IPv6 taj broj 2^{128}. Zbog određenih restrikcija vezanih uz raspodjelu IP adresa i rezervaciju prve i zadnje adrese unutar jedne mreže ili podmreže broj raspoloživih adresa je u stvarnosti manji od 2^{32}, ali se okvirno može reći kako je moguće adresirati oko 4 milijarde različitih uređaja, što predstavlja problem budući je stanovništvo svijeta odavno preraslo taj broj. Verzija 6 rješava taj problem jer broj adresa prelazi ne samo broj stanovništva Zemlje, nego i broj atoma na njenoj površini. Detaljniji opis stvaranja mreža, podmreža, dodjeljivanje IP adresa, kao i objašnjenje zaglavlja IPv6 protokola prelaze okvire ovoga rada, te se neće detaljnije obrađivati. Zaglavlje IPv4 protokola vezano je za ovaj rad te je prikazano na slici 4.



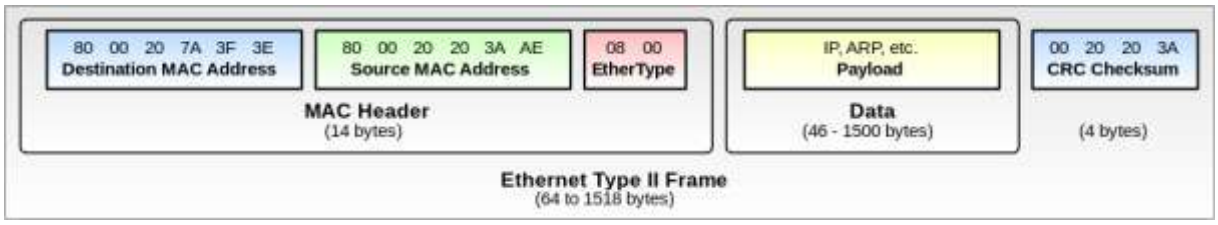
Slika 4. IP zaglavlje [3]

Polje Version označava o kojoj se verziji IP protokola radi, što je četiri u slučaju IPv4 i to je konstantna vrijednost za sve pakete poslane korištenjem IPv4 protokola (polje ima vrijednost šest kod IPv6). U polje IHL (engl. *Internet Header*

Length) je upisan ukupni broj 32 bitnih riječi (četiri okteta) od kojih je zaglavlje IP paketa sastavljeno. Standardna vrijednost je pet, a može se i povećati ukoliko se koristi polje *Options*. *Type of Service* je polje koje može sadržavati broj koji predstavlja posebnu uslugu koja se prenosi unutar tog paketa, kako bi se omogućilo drugačije postupanje s takvim paketima, npr. stavljanje na početak reda čekanja za izlazak iz usmjernika i sl. Ovo polje se rijetko koristi i najčešće iznosi 0. *Total Length* predstavlja ukupnu duljinu paketa u oktetima, dakle IP zaglavlje s podatcima (u podatke su uključena zaglavlja viših slojeva). Slijedeća tri polja trebala bi se koristiti u slučaju fragmentacije paketa. Do fragmentacije paketa dolazi ukoliko se na sloju ispod koriste različite tehnologije koje ne moraju imati istu dopuštenu maksimalnu duljinu okvira, te se takav paket mora fragmentirati (rastaviti) na dva ili više manjih. Ukoliko do toga dođe polje *Indetification* svih tih fragmenata mora imati istu vrijednost i prema preporuci se ne bi trebalo koristiti ni za što drugo, iako se u praksi veoma često koristi za označavanje slijednih paketa. Polje *Flags* ima 3 zastavice. Prva zastavica je rezervirana i uvijek je 0. Druga zastavica je *Don't Fragment*, odnosno Ne Fragmentiraj. Ukoliko je ona postavljena paket se ne smije fragmentirati, pa tako u slučaju kada ga fizički nije moguće dalje proslijediti bez fragmentiranja paket se odbacuje. Treća zastavica je *More Fragments* i označava postojanje još fragmenata osim zadnje primljenog, dakle svi fragmenti imaju tu zastavicu postavljenu osim zadnjeg. Zadnje polje vezano za fragmentiranje je *Fragment Offset*. On mjeri podatke u jedinicama od 8 okteta, odnosno specificira početni oktet fragmenta u odnosu na njegovu poziciju u ne fragmentiranom IP paketu. Time To Live je polje koje određuje koliko „skokova“ (prebacivanje paketa iz jednog usmjernika u drugi) paket još smije napraviti. Nakon što paket stigne u usmjernik vrijednost polja *Time To Live* se umanjuje za jedan i ako postane nula paket se odbacuje. Jedino se u tom slučaju generira povratna poruka koja obavještava pošiljatelja o odbacivanju paketa (barem na ovom sloju). Ovo polje je veoma korisno budući onemogućuje beskonačno kruženje paketa. Zadana vrijednost je 64. Polje *Protocol* specificira koji se protokol koristi na sloju 3. Ako se koristi TCP onda je vrijednost polja 6, a ako je UDP onda je vrijednost 17. Polje *Header Checksum*, kako mu i ime govori, je zaštitni broj za provjeru ispravnosti zaglavlja, ali ne i cijelog paketa (što je čest slučaj na drugim slojevima). Računanje je istovjetno onome kod polja *Checksum* TCP zaglavlja. *Source IP Adress* je polje koje sadrži IP adresu izvorišta informacije, dok je *Destination IP Adress* polje koje sadrži IP adresu odredišta informacije. Oba polja su

32 bitne vrijednosti. *Options* je polje koje se ne mora koristiti i često se i ne koristi u svakodnevnoj primjeni. Pomoću njega su omogućene dodatne funkcionalnosti poput zapisivanja vremena ili prijeđene rute. Polje *Padding* služi kako bi, ukoliko postoji, poravnao polje *Options* na vrijednost djeljivu s 32.

Sloj mrežnog pristupa, sloj 1, kako je već rečeno katkada može biti podijeljen u dva sloja fizički i podatkovni, ali je ipak u široj literaturi češće spojen u jedan sloj. Na njenu su definirani standardi za spajanje računala s mrežom, razna kodiranja i sl., no budući to nije od velike važnosti za ovaj rad neće se objašnjavati. Jedan veoma važan protokol ove razine je *Ethernet*, koji je nedvojbeno najčešći protokol korišten ponajviše u LAN i WAN mrežama. Koristi se od samih početaka razvitka Interneta i do danas je očuvan u gotovo neizmijenjenom stanju. Slika 5. predstavlja izgled *Ethernet* okvira.



Slika 5. *Ethernet* okvir [4]

Prvo polje *Destination MAC Address* je odredišna MAC (engl. *Media Access Control*) adresa koja je jedinstvena za svaki uređaj koji ima mogućnost spajanja na mrežu[6]. Drugo polje je *Source MAC Adress*, odnosno MAC adresa izvora podataka. Oba polja su duljine 48 bita. Polje *Ether Type* označava protokol više razine koji je u današnje vrijeme uglavnom IPv4 ili IPv6, no u prošlosti je postojalo više protokola čije su se vrijednosti mogle naći u tom polju. Ukoliko se koristi IPv4 onda se u tom polju nalazi heksadekadska vrijednost 0x0800, a 0x86DD ako se koristi IPv6.*Payload* je polje u kojemu se nalaze podatci i zaglavlja viših slojeva. *CRC Checksum* , često se naziva i FCS (engl. *Frame Check Sequence*), je polje za zaštitu od pogrešaka. Pri stvaranju okvira se uz pomoć CRC (engl. *Cyclic Redundancy Check*) načina

[6] Jedan uređaj može imati više MAC adresa ukoliko ima više pristupnih utora ili mrežnih kartica.

kodiranja stvaraju 32 zaštitan bita koji se dodaju na kraj okvira. Isti postupak se obavlja kod primitka okvira na svakom usmjerivaču i ukoliko se ne dobije rezultat 0 znači da je došlo do pogreške te se okvir odbacuje. Ovo polje će biti opisano detaljnije u nastavku rada.

3. ZNAČAJ MREŽNIH SIMULATORA

Simulacija se može definirati kao imitacija izvođenja određenih procesa ili sustava u stvarnome svijetu. Oxfordski rječnik kao jednu od definicija riječi simulacija navodi sljedeće: „*The production of a computer model of something, especially for the purpose of study*“[5]. Grubi prijevod glasio bi: „Stvaranje računalnog modela nečega, pogotovo u svrhu učenja“. Isti rječnik za termin simulator daje sljedeću definiciju: „*A program enabling a computer to execute programs written for a different operating system*“[5]. Prijevod: „Program koji računalu omogućava izvršavanje programa napisanog za drugi operativni sustav“. Još jedna definicija glasi: „*A machine designed to provide a realistic imitation of the controls and operation of a vehicle, aircraft, or other complex system, used for training purposes*“[5]. Prijevod: „Uređaj dizajniran kako bi omogućio realističnu imitaciju kontrola i operacija nekog vozila, letjelice ili drugog složenog sustava, korišten za svrhe treniranja “. Simulator koji je sastavni dio ovog rada može se definirati uz pomoć složene definicije svih do sada gore navedenih:

Simulator[7] koji je kreiran kao dio ovog rada ima za funkciju imitirati mrežni prijenos paketa, odnosno *Ethernet* okvira, uz primjenu IP protokola i TCP protokola, dakle treba omogućiti pouzdani prijenos podataka, preko umjetno stvorenog, nepouzdanog prijenosnog puta.

Postoji mnogo primjena simulatora kao npr. optimizacija performansi nekog sustava bez da se preinake odrađuju na samom sustavu, sigurnosno testiranje sustava koji se planira izgraditi, provjeravanje i potvrđivanje predloženih modela, treniranje ljudi (ali i uređaja) kako bi bili spremni raditi sa stvarnim sustavom i dr. Moguće je pronaći mrežne simulatore za sve prethodno navedene svrhe, što i potvrđuje važnu ulogu mreža i informacijske povezanosti koja je prisutna u današnjem svijetu. Jedna važna svrha još nije navedena, a to je korištenje simulatora u svrhu edukacije i obrazovanja. Ova svrha navedena je kao zadnja upravo zato što je to svrha s kojom je stvoren ovaj simulator, dakle pokušati korisniku simulatora objasniti stvaranje i prosljeđivanje paketa po mreži koju nazivamo Internet uz slikovnu interpretaciju stvarnih događaja.

[7] Simulator u slučaju ovog rada predstavlja programsko rješenje, a ne zaseban uređaj.

Kreirani simulator nije jedinstven u niti jednom pogledu, sve njegove funkcionalnosti moguće je pronaći na već postojećim simulatorima dostupnim, veoma često putem Interneta, uz plaćanje ili besplatno (ovisi o simulatoru i njegovim mogućnostima).

Najpoznatiji primjer je *Cicso Packet Tracer*. Radi se o simulatoru razvijenom u tvrtci *Cisco Systems* koja je vodeća tvrtka na području mreža i mrežnih tehnologija. Upravo zbog toga je i njihova programska izvedba, u vidu simulatora, standard kod proučavanja i educiranja na mnogim visokim učilištima i tečajevima iz područja računalnih mreža. Valja napomenuti kako mnogi tečajevi i međunarodno priznati certifikati također nose ime *Cisco*.

Wireshark, iako nije simulator, veoma je važan program također korišten i upotrebljavan u većini poslova, ali i edukacija vezanih za rad na računalnoj mreži. Radi se o programu za analiziranje podatkovnog prometa odaslanog i primljenog na računalu na kojem je isti instaliran. Omogućuje prikaz pojedinih paketa te daje dodatna objašnjenja polja unutar njih.

IMUNES je za razliku od gore navedenih manje poznat i slabije korišten simulator. Preciznije rečeno radi se o simulatoru i emulatoru, dakle može preuzimati i prenositi podatke prema stvarnom svijetu. Radi se o programu razvijenom na Fakultetu elektrotehnike i računalstva u Zagrebu, uz potporu Ministarstva znanosti i drugih kako tuzemnih tako i inozemnih institucija. Program omogućava stvaranje mreže i njen vizualni prikaz kao kod *Cisco Packet Tracer*-a, ali posjeduje svoju implementaciju programa *Wireshark* uz pomoć koje se prikazuju predani i primljeni paketi.

4. OPIS PROGRAMSKOG JEZIKA C#

C# jedan je od popularnijih programskih jezika današnjice. Prema istraživanju provedenom od strane internetske stranice *StackOverflow,* C# je treći trenutno najčešće korišteni programski jezik[6]. Radi se o modernom objektno orijentiranom programskom jeziku koji je razvijen 2000.godine[8] kako bi postao konkurencija tada veoma popularnom programskom jeziku *Java* [9] . Naime, tvrtka *Sun* (kasnije preimenovana u *Oracle*) nije dozvoljavala *Microsoftu* nikakve preinake nad svojim programskim jezikom te je to bio glavni poticaj razvoja C# jezika. Upravo iz tog razloga ova dva jezika dijele mnogo sličnosti i često se upotrebljavaju za iste vrste zadataka[7]. Mnogo obrazovnih ustanova organizira predavanja i predmete vezane za razvoj programa i aplikacija uz pomoć C# jezika. Osim toga postoji i mnogo *online* tečajeva koji podučavaju programiranje s ovim jezikom (plaćeni i besplatni). Veliku upotrebu ima pogotovo u razvoju video igara, budući je programska podrška za razvoj igara (engl. *game engine*) *Unity* napravljena za rad sa njim.

Za vrijeme pisanja ovog rada najnovija verzija jezika je 7.3[8]. Dostupna je unutar razvojne okoline *Visual Studio 2017 version 15.7*, što je ujedno i razvojna okolina korištena za razvoj simulatora vezanog za ovaj rad. Programi pisani u jeziku C# (ali i nekim drugim jezicima) mogu se pokrenuti na uređajima koji podržavaju .NET Framework razvojnu platformu koja je integralni dio svih *Windows* operativnih sustava. Detaljniji opis prevođenja jezika, kompiliranja i ostalih radnji koje obavlja .NET Framework prelaze okvire ovog rada. Valja napomenuti kako je za vrijeme pisanja ovog rada trenutno dostupna verzija .NET Framework 4.7.2[8].

Uz pomoć C#-a, kao i s većinom drugih programskih jezika, moguće je stvoriti gotovo svaku vrstu programa i aplikacije. Naravno u određenim slučajevima prihvatljivije je korištenje programskih jezika nešto oskudnijih značajki poput programskog jezika C ili C++. Kod takvih slučajeva često se zahtijeva veća brzina izvođenja[10] koju , ukoliko je program stvoren u C#, isti ne može ostvariti. Primjeri

[8] Tvrtka *Microsoft* razvila je programski jezik C#.
[9] Programski jezik Java i danas je jedan od popularnijih jezika, razvijen unutar tvrtke *Oracle.*
[10] Mnoge značajke koje pruža C#, iako olakšavaju korištenje samog jezika često usporavaju izvođenje programa koji su u njemu pisani. Način na koji C# kreira gotove programe nije moguće implementirati kod programa koji se izvode „*Close To Metal*“, odnosno veoma bližu ili neposredno na sklopovlju samog računala.

takvih programa su operacijski sustavi i *driver*-i[11]. Za stvaranje takvih programa moguće je korištenje i „programskih jezika niske razine“ (engl. *low-level programming language*). To su programski jezici koji nemaju veliku razliku naspram instrukcijskog seta samog računala, te su često vezani za određenu arhitekturu računala. Primjer su tzv. asemblerski jezici (engl. *assembly language*).

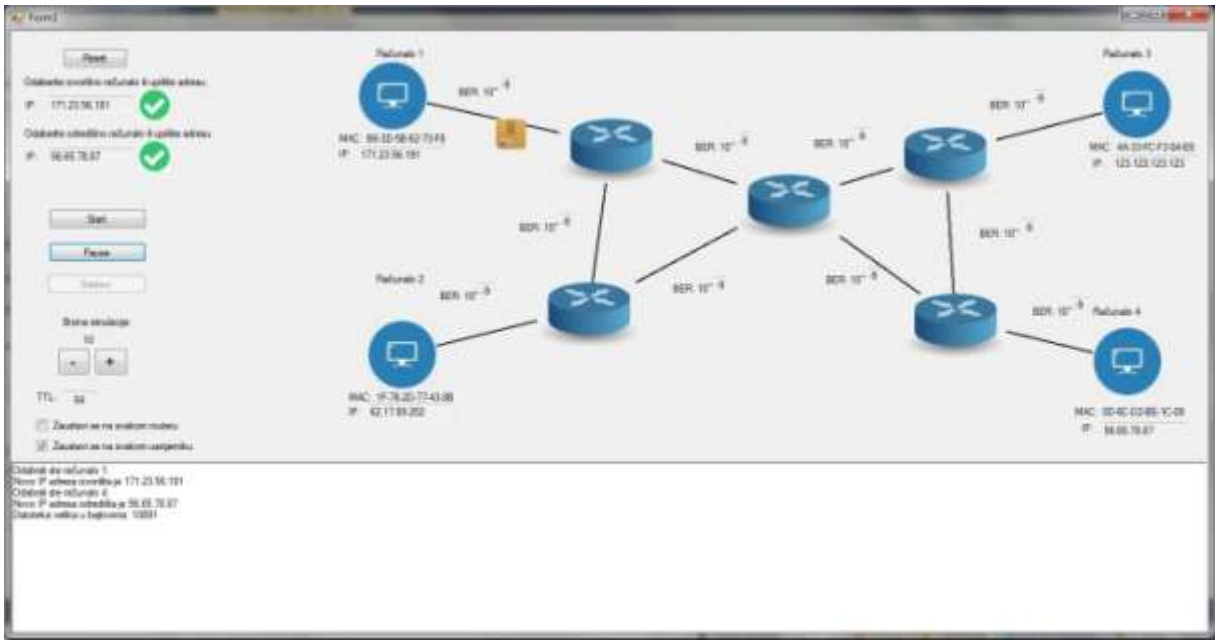
C# se pogotovo često koristi kod stvaranja *Windows desktop* aplikacija, upravo zato što je, uz primjenu *Visual Studio* okruženja, njihovo stvaranje uz C# veoma pojednostavljeno i krajnje intuitivno. Uzme li se u obzir zastupljenost operativnog sustava *Windows* na svjetskom tržištu (gledajući samo sektor osobnih računala) od otprilike 88 % [9] (u vrijeme pisanja ovog rada) može se lako zaključiti zašto je upravo C# treći najčešće korišten programski jezik.

Dokumentaciju o korištenju programskog jezika C# i razvojnog okruženja *Visual Studio 2017* moguće je pronaći na [8]. Pri navođenju pojmova i kod detaljnijih objašnjenja određenih dijelova unutar ovog rada smatra se da čitatelj posjeduje potrebno znanje za razvoj programa unutar spomenutog okruženja. Valja napomenuti kako je simulator stvoren kao *Windows Forms Application*, te da nisu upotrebljavane naprednije knjižnice (engl. *library*) metoda, već one najosnovnije koje se korisniku nude pri kreiranju novog *Windows Form Application* programa uz poneku iznimku, kao što je *System.IO.*

[11] Programi koji omogućavaju sklopovlju komunikaciju sa operativnim sustavom i obrnuto.

5. DIZAJN SIMULATORA PRIJENOSA PAKETA RAČUNALNE MREŽE

U prethodnom poglavlju navedene su neke od osnovnih značajki programskog jezika C# iz razloga što je upravo on korišten za izradu simulatora koji je sastavni i ključan dio ovog završnog rada. Simulator je stvoren u programskom okruženju *Visual Studio 2017*. Slika 6. prikazuje simulator tijekom rada, odnosno prijenosa „paketa“ po mreži.



Slika 6. Simulator prijenosa paketa računalne mreže

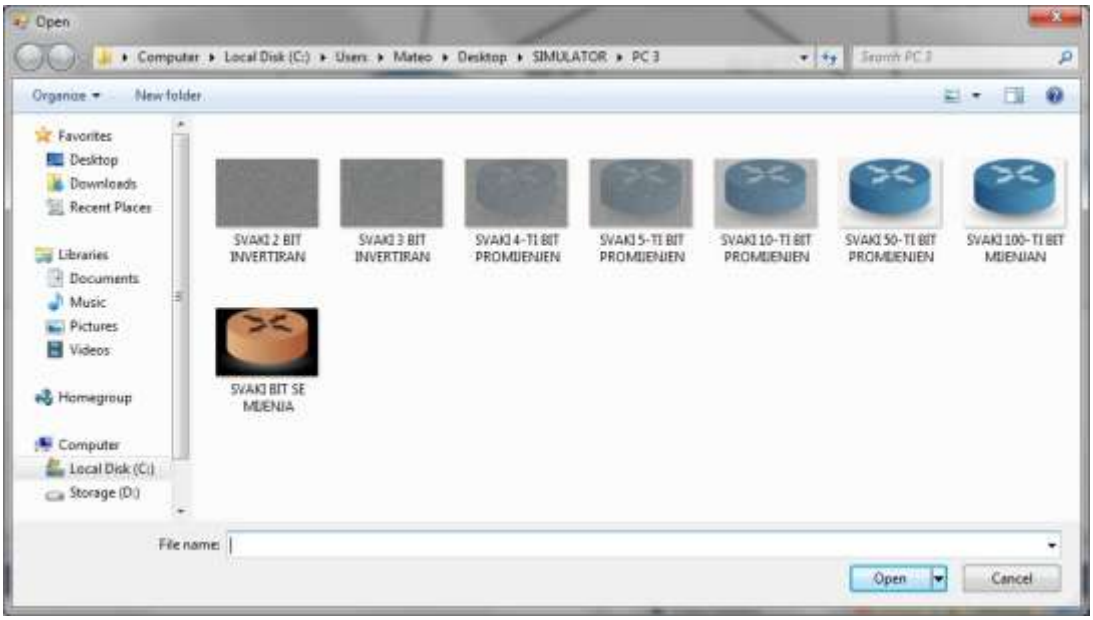
Lijevi dio slike, odnosno lijevi dio simulatora zamišljen je kao upravljački dio, gdje se korisniku nude opcije kojima može upravljati simulacijom, dok su sredina i desni dio simulatora ostavljeni za prikaz same simulacije. Donji dio simulatora je element često korišten u radu s *Form*-ama, a to je *RichTextBox*. Unutar njega se ispisuju sve važne napomene i događaji tijekom simulacije i pri završetku simulacije ispisuju se statistički podatci[12]. Radi lakšeg promatranja simulacije omogućeno je mijenjanje veličine prozora, pri čemu novonastali prostor zauzima prikaz simulacije dok svi ostali elementi ostaju nepromijenjeni. U nastavku rada objasnit će se važniji

[12]Broj pogrešaka nastalih u simulaciji, raspodjela po vrsti pogrešaka, odnos ispravnih naspram neispravno poslanih paketa.

elementi simulatora, a sama funkcionalnost i razlozi korištenja biti će objašnjeni u narednom poglavlju.

5.1 UČITAVANJE DATOTEKE

Ne postoji određeni tip datoteke koju simulator učitava, taj odabir ostavljen je korisniku na izbor. Kako bi korisnik započeo simulaciju mora odabrati datoteku. Kako bi mu se to olakšalo korištena je *OpenFileDialog* kontrola koja korisniku nudi grafički prikaz svih raspoloživih datoteka. Slika 7. prikazuje izgled otvorene kontrole.



Slika 7. *OpenFileDialog* kontrola

U ovom slučaju otvorena je mapa pod imenom „PC3“, koja se nalazi u mapi „SIMULATOR“ koja se u konačnici nalazi na radnoj površini računala. Taj put nije slučajno odabran već se postavlja pri otvaranju *OpenFileDialog* kontrole. Na slici 6. moguće je vidjeti kako ukupno u simulatoru postoje 4 računala. Zavisno o odabiru izvorišnog računala zadana (engl. *default*) putanja će se mijenjati na onu mapu koja je predodređena za odabrano računalo. Unutar tih mapa korisnik može postaviti datoteke po svom izboru. Zadavanje te putanje postiže se korištenjem *OpenFileDialog.InitialDirectory* postavke za kreiranu kontrolu. Kako bi se osiguralo

pravilno izvođenje programa ukoliko dođe do nekakve pogreške pri učitavanju koristi se *Try, Catch* naredbe unutar kojih je sve ostalo kreirano. Kada korisnik odabere datoteku ista se pomoću *File.ReadAllBytes()* metode učitava u polje byte-ova. Razlog ovakve izvedbe je što brže izvođenje simulatora, budući će cjelokupna datoteka biti direktno učitana u memoriju. Valja napomenuti kako zbog ovog pristupa postoje i ograničenja vezana za veličinu datoteke. Korisnik treba paziti na količinu slobodne radne memorije jer nije moguće učitati datoteku koja prelazi fizičku veličinu radne memorije korisnikovog računala. Savjetuje se učitavati datoteku koja je mnogo puta manja od dostupne radne memorije, budući će postojati više kopija navedenog polja unutar memorije, a i sama sporost simulatora ograničava veličinu datoteke. Kod datoteke od nekoliko stotina kB (engl. *kilobyte*) pri najvećoj brzini izvođenja simulacija traje više desetaka sekundi.

5.2 STVARANJE PAKETA

Stvaranje_paketa() posebna je metoda unutar simulatora koja se poziva najmanje jednom, ukoliko je datoteka manja od onoga što standardni *Ethernet* okvir može prenijeti, dakle 1460 *byte*-a. Ukoliko je datoteka veća ista metoda će se pozivati dokle god se svi okteti ne prenesu. Paket unutar ovog simulatora napravljen je po uzoru na *Ethernet* okvir, s TCP i IP zaglavljima, a predstavljen je unutar kôda kao polje *byte*-ova, s ukupno 1518[13] elemenata. Okteti učitane datoteke prebacuju se u polje *byte*-ova Paket i tako se započinje stvaranje paketa unutar simulatora. Globalna varijabla pohranjuje indeks zadnje prebačenog okteta kako bi se postupak pravilno obavljao pri sljedećem pozivu metode. Osim podataka potrebno je stvoriti i pravilna zaglavlja paketa, što se također obavlja unutar ove metode. Svi okteti zaglavlja popunjavaju se po pravilima danim u poglavlju 2, a informacije se preuzimaju iz *TextBox* elemenata koji su vidljivi na slici 6. ili su fiksno zadane vezano za poziciju paketa u datom trenutku simulacije. Postoje polja kod kojih je potreban matematički izračun vrijednosti kao npr. kod polja *Checksum* i FCS (vidi poglavlje 2). Zbog jednostavnosti izvedbe *Checksum* se izvodi unutar metode Stvaranje-paketa() preko for petlje koja prolazi preko paketa i zbraja svakih 16 bitova. Ukoliko dođe do

[13] 1460 okteta podataka, 20 okteta TCP zaglavlja, 20 okteta IP zaglavlja, 18 okteta *Ethernet* zaglavlja.

preljeva, isti se miče sa sume i zbraja s preostalom sumom sve dok se više ne dogodi preljev. Dobiveni broj se komplementira (jedinični komplement) i pohranjuje u polje Paket. Valja napomenuti kako su okteti na čijem mjestu se nalazi *Checksum* postavljeni na 0 budući se to način generiranja tog polje u praksi. Polje FCS malo je kompleksnije od polja *Checksum* te se izvodi u posebnoj metodi. Unutar *long* varijable učitavaju se brojevi iz paketa (bit po bit) te se posmiču do trenutka kada je 33 bit jednak jedinici (indeks tog bita je 32). Nakon toga izvršava se operacija Isključivo ILI između broja i long varijabli i preddefiniranog polinoma koji se koristi kod kreiranja *Ethernet* okvira u praksi. Taj polinom je:

$$f(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + x^0 \quad (1)$$

Nakon toga postupak ubacivanja bitova, posmicanja i operacije Isključivo ILI se ponavlja dokle god postoje bitovi unutar paketa. Broj koji je ostao na kraju se vraća metodi Stvaranje_paketa(). Postoje i polja koja u stvarnom svijetu igraju važnu ulogu, no budući se ovdje radi o simulatoru nemaju neku svrhu, poput polja Windows *Size* (vidi poglavlje 2). Takva polja popunjavaju se uz pomoć metode *Random* sa slučajno odabranim brojem iz zadanog intervala.

U pravilu se paket mijenja kod svakog prolaska kroz usmjernik. Polja koja se mijenjaju su Time To Live, *Checksum* (unutar TCP i IP zaglavlja), *Destination MAC Address*, *Source MAC Address* i FCS. Budući sva ostala polja ostaju nepromijenjena, nema smisla pozivati metodu Stvaranje_paketa(), nego se kod prolaska paketa kroz usmjernike poziva metoda Update_paketa() koja mijenja navedena polja (po pravilima opisanima u poglavlju 2).

5.3 STVARANJE IZLAZNE DATOTEKE

Stvaranje_izlazne_datoteke() je metoda koja, kako joj i ime govori, stvara datoteku koja će se pohraniti u mapi odredišnog računala. Mapa u kojoj će se stvoriti datoteka određena je prije početka simulacije, nakon što korisnik odabere odredišno računalo i nije ju moguće mijenjati u datoj simulaciju. Ukoliko je paket stigao bez pogrešaka na odredišno računalo, pokreće se ova metoda. Unutar nje postoji lista kojoj se nadodaju elementi paketa, ali samo oni koji su vezani za podatke učitane

datoteke. Podatci zaglavlja svih slojeva se ne pohranjuju u nju. Pohrana se obavlja jednostavnom for petljom koja prolazi preko svih okteta polja *byte*-ova Paket u kojima se nalaze podatci i koristi se naredba *List.Add().* Nakon što se odradi pohrana podataka u listu ponovno se pokreće metoda Stvaranje-paketa(). Ukoliko se radi o zadnjem paketu koji je potrebno prenijeti u datoj simulaciji, tada se ne pokreće metoda Stvaranje_paketa(), već se uz pomoć funkcije *File.WriteAllBytes()* upisuje, odnosno stvara nova datoteka koja sadrži sve oktete koji su elementi liste. Naziv novostvorene datoteke jednak je nazivu datoteke koja je odabrana na početku.

5.4 SIMULACIJA POKRETA PAKETA

Paket, odnosno njegova slika fizički se pomiče između dva uređaja, što je moguće vidjeti na slici 6. To se ostvaruje pomoću posebne klase unutar C# jezika koja se zove *BackgroundWorker* (u daljnjem tekstu BW). Kako se ne bi opteretila glavna dretva stvara se sporedna dretva koja može obavljati određeni posao. U slučaju ovog simulatora BW računa sljedeću poziciju paketa. Ovisno o odredištu, izvorištu, trenutnom usmjerniku, odnosno sljedećem usmjerniku do kojega paket treba stići ovisit će sljedeća pozicija paketa. Nakon što se odredi prema kojem uređaju se paket treba kretati, ulazi se u metodu Nova_lokacija() (ulazi se iz dretve BW) gdje se određuju nova pozicija slike paketa. Moguće je osam različitih slučaja kretanja. Nakon odabira jednog od tih slučaja, izračunavaju se nove koordinate elementa *PictureBox* (koji sadrži sliku paketa) po X i Y osi, te se iste osvježavaju (engl. *Update*). Problem kod ovakvog pristupa jest nemogućnost promjene parametara elementa koji je stvoren u jednoj dretvi iz druge dretve. *PictureBox* je element stvoren u glavnoj dretvi, pa je potrebno korištenje Delegata (engl. *Delegate*) koji pojednostavljeno rečeno omogućuju slanje poruka i/ili naredbi između različitih dretvi. Prethodno opisani proces ponavlja se sve dok element *PictureBox* paketa ne stigne do *PictureBox*-a nekog od uređaja koje je moguće vidjeti na slici 6. Ukoliko je taj uređaj usmjernik pokreće se metoda Update_paketa() (vidi poglavlje 5.2), a ukoliko je uređaj upravo odredišno računalo pokreće se metoda Stvaranje_izlazne_datoteke() (vidi poglavlje 5.3).

5.5 GENERIRANJE POGREŠAKA

Simulacija se u potpunosti obavlja unutar računala, te se ne mogu dogoditi pogreške kao u stvarnom svijetu. Zbog toga je potrebno umjetno generirati pogreške. Na slici 6. može se vidjeti da pored svake poveznice između dva uređaja postoji oznaka. Te oznake označavaju razinu BER-a (engl. *Bit Error Rate/Ratio*) na svakoj pojedinoj poveznici, odnosno linku. U stvarnom svijetu pri prolazu okvira(paketa) kroz fizički medij dolazi do smetnji, interferencija i drugih negativnih učinaka koji u konačnici mogu izmijeniti signal koji bi umjesto 1 prikazivao 0 i obrnuto. Detaljan opis događaja na fizičkom mediju prelazi okvire ovog rada. Sličan proces je implementiran i u ovom simulatoru. Nakon što *PictureBox* paketa prijeđe link odnosno dođe do sljedećeg uređaja, sadržaj paketa se „provodi“ kroz metodu koja može izazvati promijene bitova[14] (ne nužno cijelog okteta). Ovisno o tome preko kojeg je linka paket prolazio ovisit će i vjerojatnost pogreške (već spomenuti BER).

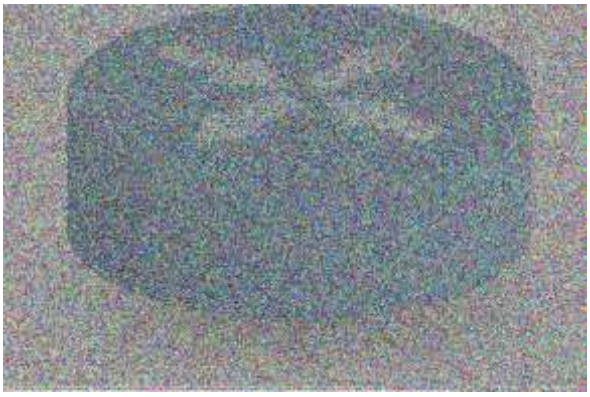
Postupak je sljedeći: metoda očitava vrijednost BER-a za zadani link te njemu recipročan broj pretvara u gornju granicu intervala iz kojeg se bira slučaja broj. Metodom *Random* odabire se broj iz intervala, te se uspoređuje s brojem koji je izabran kao traženi, u slučaju ovog simulatora to je broj 5. Ako se ispostavi da je slučajno odabran broj jednak broju 5 bit se mijenja iz 0 u 1 i obrnuto. Ovaj postupak se ponavlja za svaki oktet unutar polja *byte*-ova Paket, a na svakom oktetu se izvršava 8 puta[15], kako bi se uistinu omogućila izmjena svakog bita.

Slike 8., 9., 10., i 11. prikazuju što se dogodi sa slikom u .bmp (*Bitmap*) formatu ukoliko se provede kroz tu funkciju.

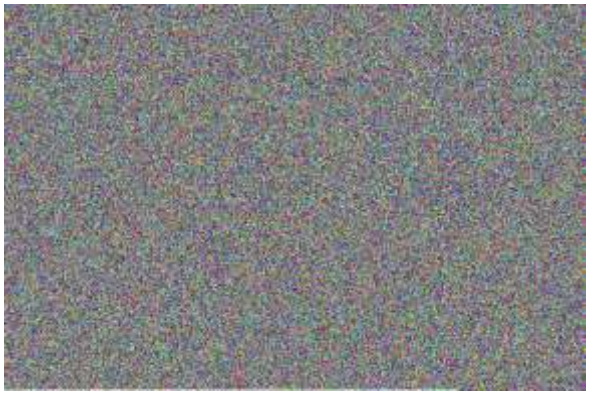
[14] Najmanja jedinica podataka koju je moguće stvoriti i njome upravljati unutar razvojne okoline *Visual Studio* je *byte*, odnosno oktet (8 bitova).

[15] BER označava vjerojatnost promjene bita, a ne okteta, te je potrebno trenutno obrađivani oktet testirati 8 puta kako bi se uistinu simulirala promjena bita koja se događa u realnim uvjetima. To se postiže posmakom vrijednosti 128 udesno za jedno mjesto u svakom koraku i ukoliko je potrebno generirati pogrešku koristi se operacija isključivo ILI između posmaknute vrijednosti i trenutnog okteta. Time je promjena odrađena upravo na onom bitu koji bi se promijenio u realnim uvjetima.

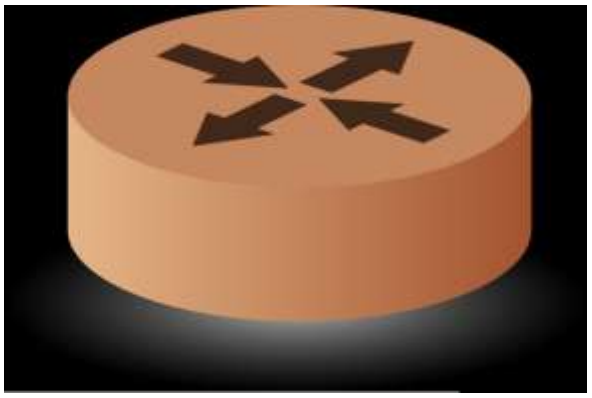
Slika 8. Svaki 100-ti bit promijenjen



Slika 9. Svaki 10-ti bit promijenjen



Slika 10. Svaki drugi bit promijenjen



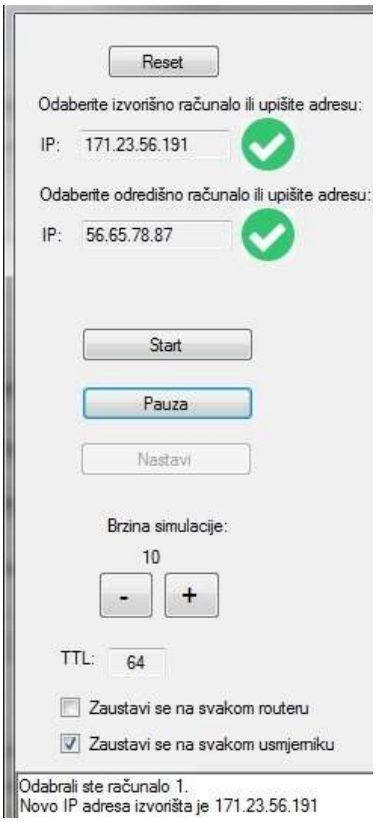
Slika 11. Svaki bit promijenjen

Valja napomenuti kako je .bmp format stvoren tako da svaki element slike (engl. *pixel*) predstavi kao 24-bitnu vrijednost, odnosno 8-bitnu vrijednost za svaku od tri glavne boje crvenu, zelenu i plavu. Slika 8. predstavlja promjenu na svakom 100-tom bitu i može se vidjeti kako je to rezultiralo samo šumom na slici, a slika je i dalje veoma čista. Ukoliko promijenimo svaki 10-ti bit (slika 9.) slika je dosta degradirana, ali je i dalje moguće vidjeti njezin originalni sadržaj. Po tome se da zaključiti kolika je ustvari redundancija .bmp formata. Slika 10. predstavlja potpuni šum i nije moguće ništa raspoznati iz slike, odnosno ona ne nosi nikakvu informaciju. Slika 11. je nastala promjenom svakog bita i moguće je jasno vidjeti originalnu sliku, ali u negativu, budući su sve tri komponente (boje) ustvari komplementirane.

5.6 OSTALO

Do sada su opisane neke od glavnih karakteristika u dizajnu ovog simulatora, no ima ih još mnogo. Budući nema smisla opisivati svaku pojedinu karakteristiku u detalje nekoliko njih će se skraćeno objasniti u ovom potpoglavlju.

Na slici 12. može se vidjeti lijeva strana simulatora, odnosno lijeva strana slike 6.



Slika 12. Upravljanje simulatorom

Simulator ima ukupno 5 tipki koje su vidljive na gornjoj slici. Reset, Start, Pauza, Nastavi, + i -. Također postoje i polja za unos vrijednosti i kućice za označavanje. Detaljniji opis svih funkcija koje pojedini elementi pružaju biti će opisane u sljedećem poglavlju. Pritiskom na bilo koju od gore navedenih tipki pokreće se događaj (engl. *Event*), gdje dolazi do uključivanja i isključivanja globalnih zastavica, postavljanja vrijednosti internih varijabli na tražene vrijednosti ili postavljanje svih na 0. Određene tipke onemogućuju (engl. *Disable*) i/ili omogućuju

(engl. *Enable*) ostale tipke i polja za unos. Također se uz pomoć tih tipki (osim tipki + i -) pokreće i zaustavlja BW. Svime navedenim omogućeno je upravljanje, odnosno zaustavljanje simulacije.

Dodatna mogućnost je i pregledavanje sadržaja paketa u bilo kojem trenutku. Prikaz se odvija u novom prozoru i moguće je pregledati izgled paketa u cijelosti ili po slojevima (prikazi će biti pokazani u sljedećem poglavlju). Simulator označava promijenjene bitove (ili oktete). To je postignuto slanjem prethodne vrijednosti polja *byte*-ova Paket i slanjem trenutne vrijednosti u novu *Form*-u (ona upravlja novim prozorom), koja zatim ispisuje vrijednost oba poslana polja *byte*-ova te pronalazi razlike. Razlike se pronalaze prolaskom For petlje preko sadržaja oba polja i upisivanjem indeksa mjesta gdje se oni razlikuju u listu (engl. *List*) pogrešaka. Indeksi unutar te liste se koriste za daljnje obilježavanje pogrešaka i razlika.

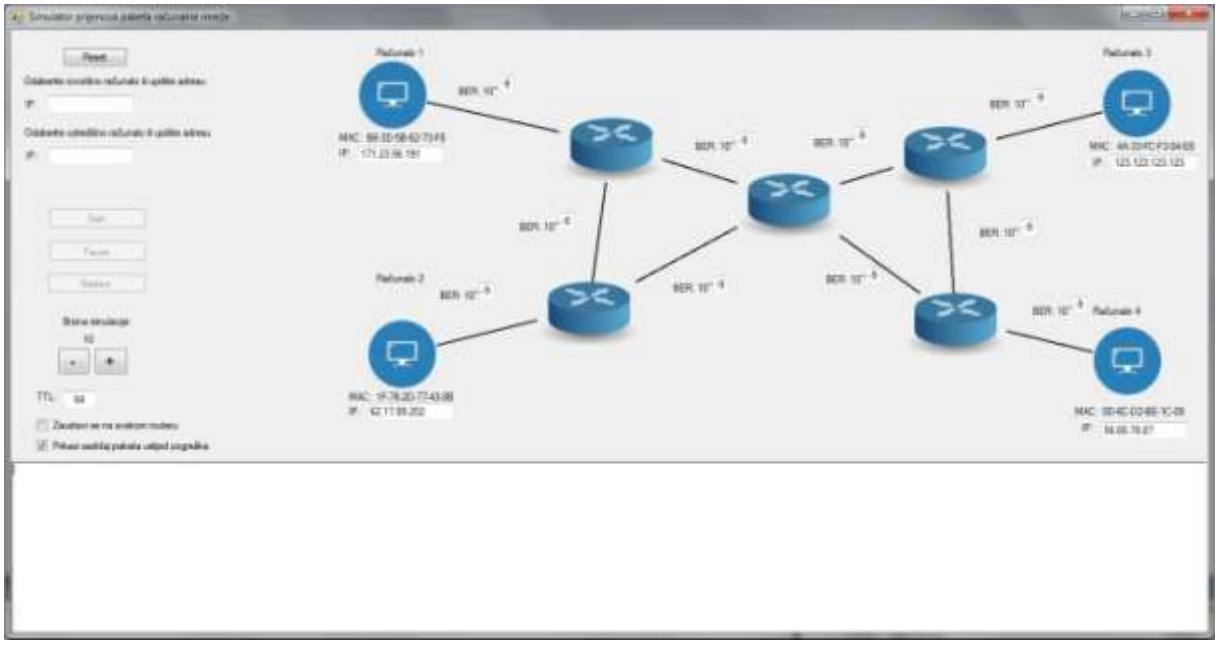
Simulator također prikazuje generiranje ACK paketa (vidi poglavlje 2). Postoje razlike u stvaranju ACK paketa, budući on nema podataka unutar sebe, a kako njegovo stvaranje ne bi utjecalo na mnoge varijable i zastavice koje se koriste za ispisivanje rezultata na kraju simulacije, njegovo kreiranje obavlja se u posebnoj metodi ACK_stvaranje_paketa() koja je veoma slična metodi opisanoj u poglavlju 5.2. Još jedna razlika je slika takvog paketa. Ustvari, radi se o istom *PictureBox*-u kao i za prethodni paket samo se pri generiranju ACK paketa slika unutar tog *PictureBox*-a mijenja i nije više smeđe boje (vidi sliku 6.) nego zelene.

Ukoliko se dogodila pogreška na paketu simulator imitira odbacivanje paketa. *PictureBox* paketa se prebacuje na originalno izvorište, resetiraju se sve do tada nastale promijene i povećavaju varijable vezane za brojanje odbačenih paketa i izmijenjenih bitova. Trenutna vrijednost polja *byte*-ova Paket se zanemaruje, te se u njega unose vrijednosti iz polja pohranjenog na samom početku prijenosa tog paketa, dakle prije nego se bilo kakva promjena mogla dogoditi.

6. FUNKCIONALNOST SIMULATORA PRIJENOSA PAKETA RAČUNALNE MREŽE

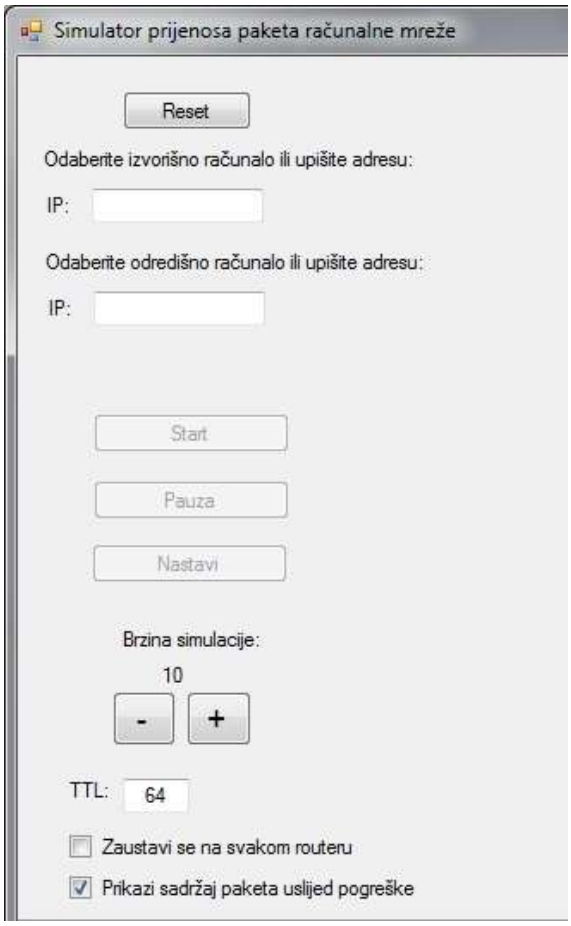
U ovom poglavlju biti će pojašnjene mogućnosti simulatora, bez opisivanja načina na koji su iste postignute (taj dio je obrađen u prethodnom poglavlju).

Pri otvaranju simulatora početni prozor izgleda kao na slici 13.



Slika 13. Početni izgled simulatora

Pogleda li se pobliže lijevi dio simulatora, tamo se nalaze tipke i polja za unos, te kućice za označavanje (engl. *CheckBox*). Slika 14. prikazuje upravo taj dio simulatora



Slika 14. Lijevi dio početnog prozora simulatora

Prva tipka, na samom vrhu, je tipka Reset. Pritiskom na nju poništava se sve do sada napravljeno unutar simulatora. To uključuje zaustavljanje BW-a ako je isti bio aktivan, vraćanje zadanih (engl. *default*) postavki u sva polja simulatora, brisanje vrijednosti svih internih varijabli i zastavica, odnosno njihovo postavljanje na početnu vrijednost. Dakle, pritisak na tipku Reset istovjetan je isključivanju i ponovnom uključivanju programa, pa će simulator izgledati identično slici 13.

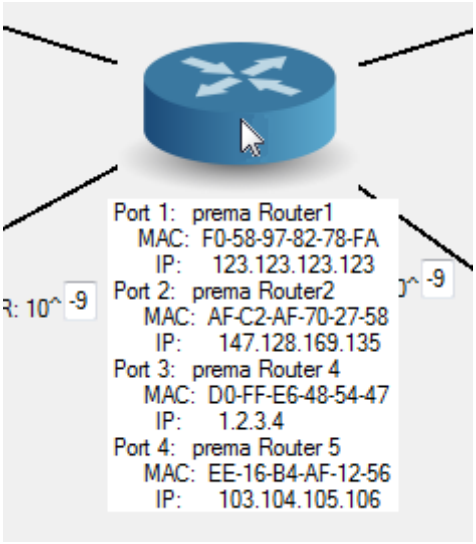
Ispod tipke Reset nalaze se dva polja za unos. U njih je potrebno unijeti izvorišnu i odredišnu IP adresu. Postoji nekoliko načina na koji se to može napraviti. Moguće je jednostavno upisati IP adresu jednog od četiri računala koji su prikazani na slici 13. Drugi način je odabir polja koje trenutno želimo popuniti, tako da se klikne na samo polje ili jednu od oznaka pored tog polja, dakle na element Label. Nakon odabira željenog polja može se jednostavno kliknuti na jednu od slika računala i njegova adresa će se sama upisati u odabrano polje. Ukoliko se kod upisa dogodi pogreška isto će biti dojavljeno korisniku preko teksta ispisanog na elementu

RichTextBox (veliki bijeli kvadrat u donjem dijelu prozora). Osim toga pored tog polja prikazat će se znak kao na slici 15., a ukoliko nije bilo pogrešaka prikazat će se znak istovjetan slici 16.

Slika 15. Znak za neispravan unos

Slika 16. Znak za ispravan unos

Nekoliko se situacija smatra neispravnim unosom: ukoliko je adresa nepravilno upisana (pravilan upis je u obliku: 123.456.789.012, dakle četiri broja od 0 do 255 odvojeni točkom), ukoliko su adrese izvorišta i odredišta iste, ukoliko nije ništa upisano, te ako adresa izvorišta nije jednaka IP adresi jednoga od 4 računala ili je jednaka jednoj od IP adresa usmjernika. Valja napomenuti kako adresa odredišta ne mora biti jednaka niti jednome od 4 računala, ali ne smije biti ista kao niti jedna IP adresa usmjernika. IP adrese usmjernika moguće je vidjeti ako se kursor miša pozicionira iznad slike usmjernika. Tada se prikazuje popis IP adresa utora na svakom usmjerniku i njima pripadajuće MAC adrese, prikazano na slici 17.



Slika 17. Popis IP i MAC adresa na usmjerivaču

Ukoliko su obadvije adrese dobro unesene i ne krše niti jedno od gore navedenih uvjeta tada će pored oba elementa za unos biti prikazan znak istovjetan onom na slici 16.

Ispod navedenih polja za unos nalazi se tipka Start. Sve dok se ne unesu ispravne vrijednosti za izvorišnu i odredišnu IP adresu, tipka Start će biti onemogućena. Ispravnim unosom tipka se omogućuje te ju je potrebno stisnuti za nastavak simulacije. Pritiskom na nju otvara se izbornik za odabir datoteke koja se želi prenijeti s jednog računala na drugo (vidi sliku 7.). Odabirom datoteke simulacija započinje. Ukoliko nije odabrana niti jedna datoteka ispisuje se upozorenje korisniku i simulacija ne započinje.

Ispod tipke Start nalazi se tipka Pauza, koja je također onemogućena. Ona se omogućuje tek početkom simulacije, dakle nakon što je odabrana datoteka za prijenos. Pritiskom na nju simulacija se zaustavlja, odnosno slika paketa miruje. Kako bi se ponovno pokrenula potrebno je ponovno pritisnuti tipku Start. Ukoliko se to napravi simulacija se jednostavno nastavlja od tamo gdje je zaustavljena, dakle ne počinje iz početka. Dok je simulacija u takvom stanju (zaustavljena) moguće je pritiskom miša na sliku paketa otvoriti novi prozor koji prikazuje sadržaj paketa (ova mogućnost biti će objašnjena malo kasnije).

Sljedeća je tipka Nastavi, koja je također onemogućena. Ona je onemogućena sve dok se ne otvori novi prozor u kojemu se prikazuje stanje paketa, bilo zbog pogreške ili zato što je korisnik to zatražio. Nakon zatvaranja tog prozora potrebno je pritisnuti tipku Nastavi. Ukoliko ne bi bilo tipke Nastavi, simulacija bi se nastavila odmah po zatvaranju prozora.

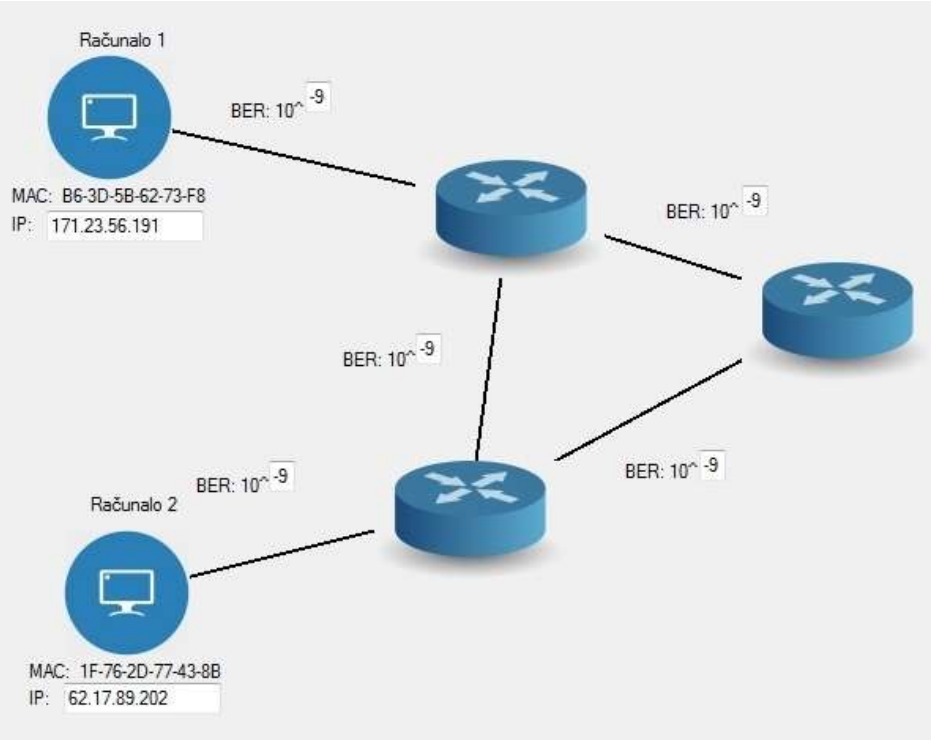
Ispod tipke Nastavi nalazi se broj ispod kojeg su dvije tipke „+“ i „-“. Broj označava vrijeme čekanja između dva pomaka slike paketa. Standardna vrijednost je 10, što predstavlja 10 milisekundi čekanja između svakog pomaka. Ukoliko se želi ubrzati simulacija potrebno je taj broj smanjiti, a ukoliko se simulacija želi ubrzati potrebno ga je povećati. Povećanje i smanjenje obavlja se tipkama „+“ i „-“. Ukoliko korisnik pokuša smanjiti vrijednost ispod 0 isto mu se neće dopustit. Gornja granica nije određena.

Nakon toga nalazi se polje za unos s opisom TTL, što je kratica od termina Time To Live (vidi poglavlje 2). Zadana vrijednost je 64, dakle paket može proći kroz 64 usmjernika prije nego bude odbačen. Korisnik može mijenjati ovu vrijednost samo na početku simulacije, dakle pri paljenju programa ili nakon pritiska na tipku Reset. Ukoliko korisnik pokuša unijeti nepravilnu vrijednost ispisuje se poruka (u *RIchTextBox*-u) koja ga obavještava o nepravilnom unosu i vrijednost se tada vraća na 64. Nepravilnim unosom smatra se vrijednost koja sadrži neki znak koji nije brojka i bilo koji broj koji nije iz intervala od 1 do 255 (polje TTL je 8 bitno pa je 255 najveća vrijednost koja se u njega može upisati).

Ispod polja TTL postoji kućica za označavanje, koja je zadano neoznačena. Tekst kućice glasi „Zaustavi se na svakom routeru“. Ukoliko je ta kućica označena paket će stati svaki put kada dođe do usmjernika i potrebna je intervencija korisnika za nastavak simulacije (korisnik mora pritisnuti tipku Nastavi). Ovu je opciju moguće mijenjati tokom simulacije, ne samo na njenom početku.

Zadnja kućica za označavanje ima tekst „Prikaži sadržaj paketa uslijed pogreške“. Zadano (engl. *Default*) je označena i kao i prethodnu kućicu moguće ju je označavati i uklanjati oznaku tokom simulacije. Ukoliko je kućica označena pri pojavi pogreške se otvara prozor u kojemu je moguće vidjeti stanje paketa prije pogreške i stanje paketa nakon pogreške, gdje su nepravilnosti označene crvenom bojom. Sadržaj paketa je moguće prikazati cjelokupno ili prikazivati pojedina zaglavlja različitih slojeva.

Ostala polja koje korisnik može promijeniti prikazana su na slici 18. To su polja vezana za usmjerivače i stanje linkova. Slika ne prikazuje sve elemente početnog prozora koji su korisniku dostupni i koje on može mijenjati. Prikazana je sredina početnog prozora simulatora iz razloga što je pri ovakvom prikazu moguće lakše pročitati tekst i vidjeti elemente preciznije, a ostala dva računala i dva usmjernika identični su prikazanima.



Slika 18. Sredina početnog prozora simulatora

Može se vidjeti kako su na svi računalima upisane MAC adrese koje su unutar ovog simulatora nepromjenjive. Budući je svako računalo spojeno samo s jednim usmjernikom potrebna je samo jedna MAC adresa po računalu. Usmjernici imaju više poveznica, odnosno linkova prema drugim usmjernicima ili računalima. Za svaki taj link potrebna im je po jedna MAC adresa, ali i jedna IP adresa. Zbog preglednosti nisu upisane no već je prethodno u radu objašnjeno kako se popis adresa prikazuje ukoliko se kursor miša postavi iznad usmjernika (vidi sliku 17.). IP i MAC adrese usmjernika nije moguće mijenjati. IP adrese računala prikazane su u elementu *TextBox* ispod MAC adrese i njih je moguće promijeniti. Naravno, nije moguće upisati bilo kakvu vrijednost, stoga se u slučaju neispravnog unosa polje vraća na prethodnu vrijednost i korisniku se ispisuje poruka o pogrešci u *RichTextBox*-u. Neispravnim unosom smatra se IP adresa koja je već u upotrebi, bilo kod računala ili usmjernika, i vrijednosti koje nemaju smisla (već opisano unutar ovog poglavlja).

Posljednja opcija koju korisnik može mijenjati je vrijednost BER-a na pojedinom linku. Prije same brojke nalazi se tekst „BER: 10^“ što znači da je vrijednost unutar polja broj kojim se potencira broj deset. Ovakav način zapisa je

preuzet iz literature koja se bavi područjem računalnih mreža i prijenosa podataka. Unutar nje je moguće vidjeti vrijednosti poput 10^{-9} ili 10^{-6}. Ukoliko se u ta polja unese vrijednost manja od -9 ili veća od 0 simulator dojavljuje neispravnost u *RichTextBox*-u te vraća vrijednost polja na zadanu, a to je -9.

6.1 UČITAVANJE DATOTEKE , STVARANJE PAKETA i POHRANA DATOTEKE

Učitavanje datoteke olakšano je primjenom prozora za izbor (vidi sliku 7.). Iako je već rečeno kako se otvara predefinirana mapa unutar tog prozora fizički je moguće izaći iz te mape, no ne preporuča se. Odabir datoteke je također slobodan izbor korisnika no zbog trajanja simulacije preporuča se koristiti datoteke veličine nekoliko desetaka kB, do nekoliko stotina kB. Kako je već u radu prije navedeno, nikakva restrikcija ne postoji glede formata datoteke koja se učitava.

Stvaranje datoteke odvija se unutar simulatora te korisnik u istom ne može sudjelovati. Jedino gdje korisnik sudjeluje je izbor odredišne i izvorišne IP adrese te broj TTL, a sve ostalo simulator odrađuje sam.

Pohrana datoteke se ostvaruje tek kada je simulacija završila. Pohranjuje se u mapu koja je dodijeljena odredišnom računalu. Korisnik može tijekom izvođenja simulacije otvarati odredišnu mapu i vidjeti kako tamo uistinu ne postoji datoteka.

Prije pokretanja simulatora potrebno je stvoriti na Radnoj Površini (engl. *Desktop*) mapu pod nazivom „SIMULATOR“ i unutar nje četiri mape „PC1“, „PC2“, „PC3“ i „PC4“.

6.2 ODBACIVANJE I PONOVNO SLANJE PAKETA

Simulator predstavlja mrežu koja koristi TCP protokol i prenosi se preko *Ethernet* okvira, dakle ukoliko dođe do pogreške i jednog bita unutar paketa isti se odbacuje, te se u slučaju ispravnog zaprimanja paketa očekuje odgovor. U prethodnom poglavlju objašnjen je postupak stvaranja pogreške (vidi potpoglavlje 5.5), ali je potrebno znati i kako se ta greška otkriva. Nakon što paket prijeđe link i

dođe do usmjerivača, počinje ispitivanje paketa. Ispitivanje se vrši generiranjem FCS zaštitnog broja nad cijelim paketom (vidi poglavlje 2). Ukoliko tako izračunat broj daje 0 znači da se unutar paketa nije dogodila niti jedna pogreška, te se može nastaviti s promjenom *Ethernet* zaglavlja paketa i daljnjim slanjem ili pohranom podataka u datoteku za ispis ukoliko je paket stigao do odredišta. Ako je stigao potrebno je generiranje ACK paketa i slanje istog prema izvorištu podatkovnog paketa. Valja napomenuti kako postoji mogućnost da se pojavi određen, dosta velik, broj pogrešaka na specifičnim mjestima i da to također može rezultirati nulom pri izračunu FCS-a, ali vjerojatnost takvog scenarija je veoma malena, ali zato postoje i druga zaštitna polja, također već prije spomenuta: *Checksum* unutar TCP i IP zaglavlja.

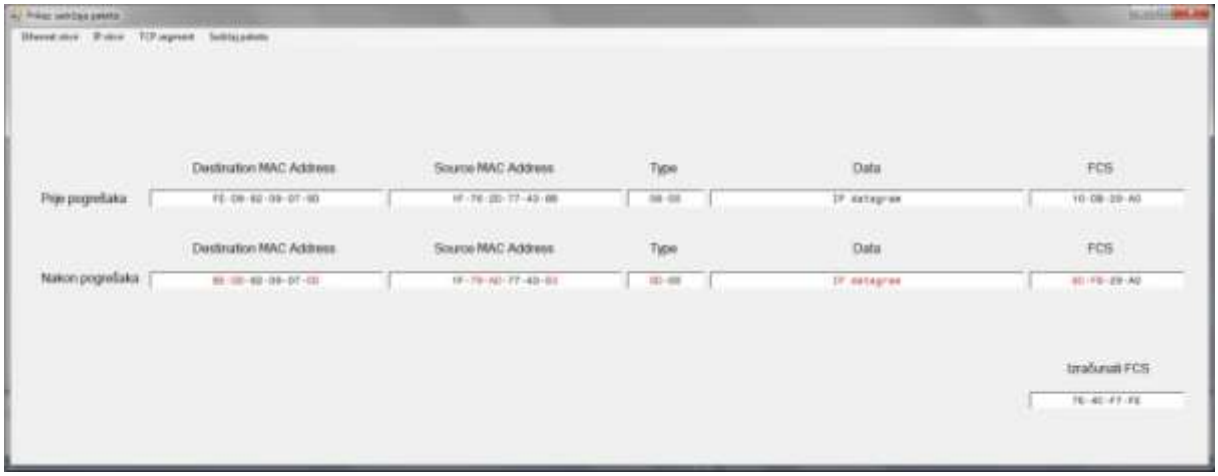
Ukoliko je došlo do pogreške, odnosno ukoliko novoizračunati FCS nije 0, usmjernik odbacuje paket i korisniku se ispisuje obavijest o tome unutar *RichTextBox*-a. Budući predajna strana nije zaprimila ACK paket ponovno se šalje isti okvir. U stvarnosti se ponovno slanje paketa odvija nakon isteka vremena za čekanje odgovora, a to je vrijeme koje računalo samo generira i prati. Zbog toga se unutar *RichTextBox*-a ispisuje kako je vrijeme čekanja na paket isteklo. Može se dogoditi i da pogreška nastane na ACK paketu te se on odbaci. U tom slučaju je prijemna strana primila ispravan paket, pohranila sadržaj tog paketa u datoteku za ispis, ali predajna strana nije dobila potvrdu, te se ponovno šalje isti paket. U tom slučaju prijemna strana jednostavno ne pohranjuje podatke iz tog ponovljenog paketa.

6.3 PRIKAZ SADRŽAJA PAKETA

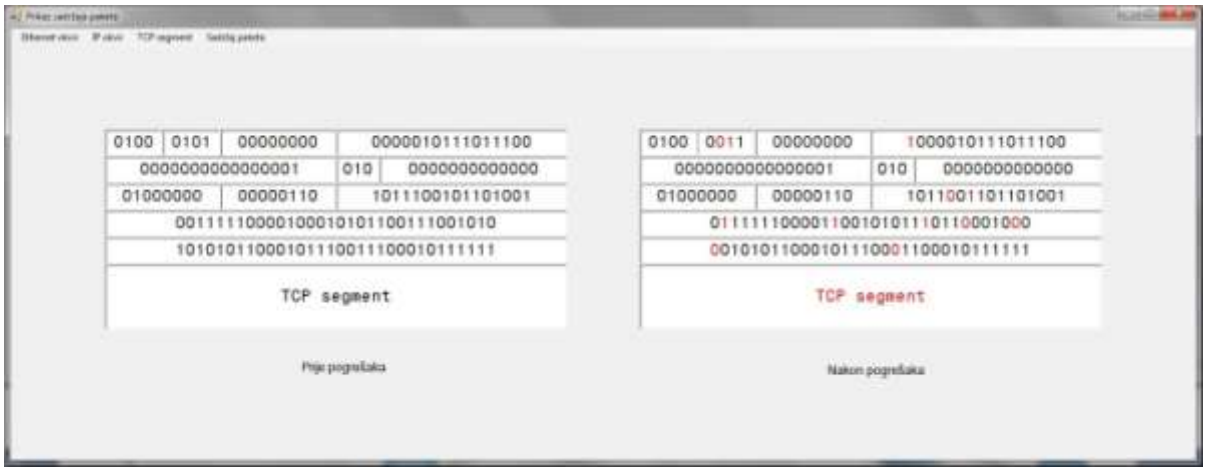
Do sada je nekoliko puta unutar ovog rada spominjana mogućnost prikaza sadržaja paketa. Dva su slučaja u kojima se to događa. Prvi slučaj je prikaz paketa prije i poslije pogrešaka, i ukoliko korisnik to želi taj se prikaz otvara odmah nakon detekcije neispravnosti FCS-a. Ukoliko korisnik to ne želi može odznačiti kvačicu unutar kućice „Prikaži sadržaj paketa uslijed pogreške“. Drugi slučaj jest tijekom zaustavljanja simulacije i klikom miša na sliku paketa. Simulacija može biti zaustavljena pritiskom tipke Pauza ili označavanjem kvačice u kućici „Zaustavi se na svakom routeru“.

Ovisno o tome koji je od dva spomenuta slučaja uzrokovao otvaranje novog prozora postojat će nekoliko razlika. Ako je prozor otvoren zbog pogreške nad bitovima, tada će bitovi (ili okteti) koji se razlikuju biti obojani crvenom bojom, a ako je prozor otvoren željom korisnika tada će polja koja su različita biti obojana zelenom bojom. Razlog korištenja drugačijih boja je sljedeći: kada je došlo do pogreške potrebno je prikazati paket prije i nakon pogreške, pa ima smisla obojati promijenjene bitove crveno. No, ako korisnik samo želi vidjeti sadržaj paketa, tada se osim trenutnog sadržaja paketa prikazuje i prijašnji paket. Valja napomenuti kako se paket mijenja u svakom koraku, dakle na svakom usmjerniku i računalu. Nema smisla bojati razliku crvenom bojom budući su oba paketa ispravna, pa se zelenom bojom označuju cijela polja kako bi se korisniku pokazalo koja se polja mijenjaju u svakom koraku. Još jedna dodatna razlika između dva načina otvaranja prozora sa sadržajem paketa je dodatno polje koje ispisuje izračunati FCS zbog kojeg je i otkrivena pogreška. U drugom slučaju nema smisla računati i ispisivati FCS.

Omogućena su četiri različita načina prikaza paketa. Prikaz *Ethernet* zaglavlja, prikaz IP zaglavlja, prikaz TCP zaglavlja ili prikaz cijelog paketa. Slike 19., 20., 21., 22. prikazuju sve te različite prikaze nabrojanim redom.



Slika 19. Prikaz *Ethernet* zaglavlja paketa



Slika 20. Prikaz IP zaglavlja paketa

Slika 21. Prikaz TCP zaglavlja paketa

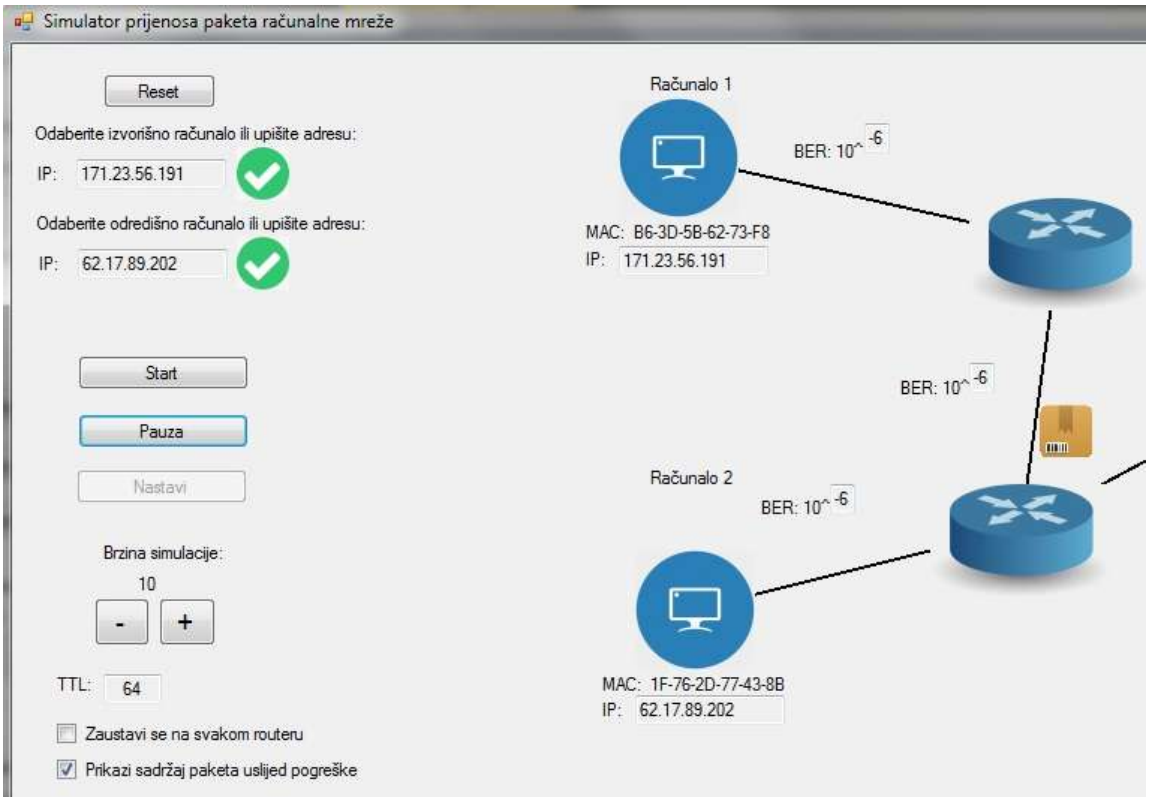
Slika 22. Prikaz ukupnog paketa

Nažalost zorniji prikaz nije moguć zbog ograničenja formata završnog rada u vidu broja stranica, ali i mogućnosti prikaza isključivo slika.

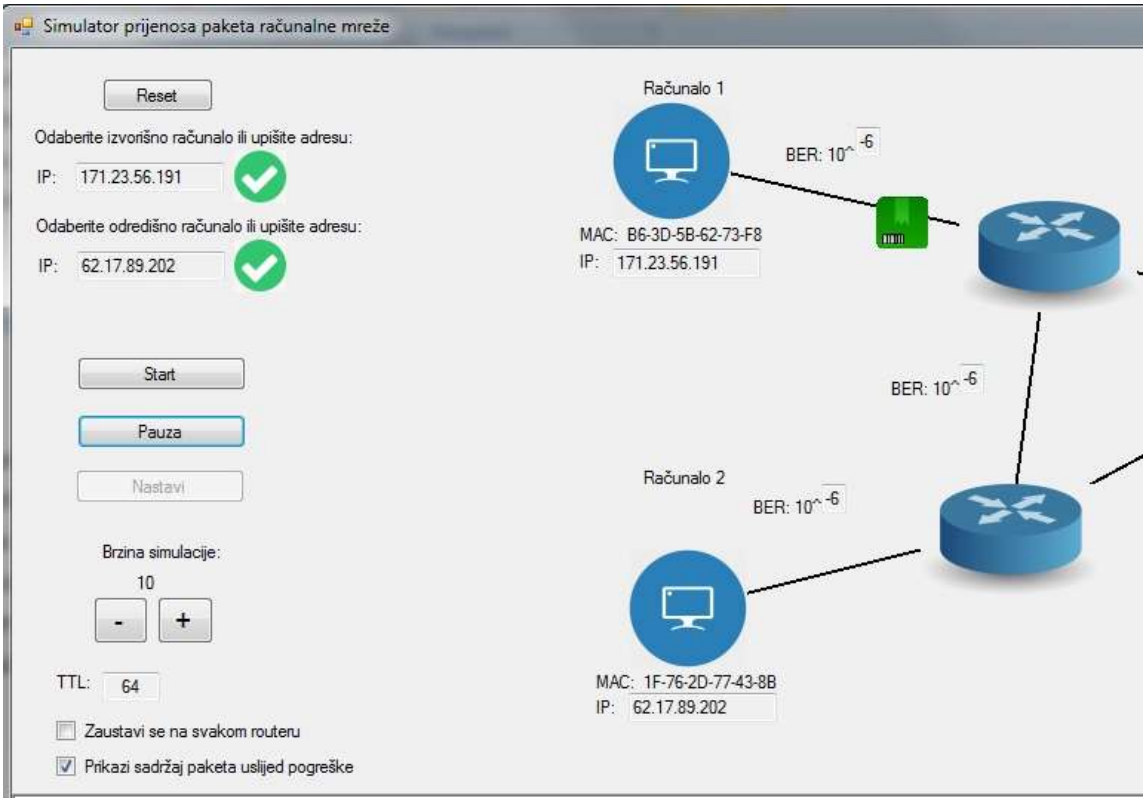
7. PRIMJER SIMULACIJE U DIZAJNIRANOM I IZRAĐENOM SIMULATORU

Kroz rad je nekoliko puta naglašavano kako je simulator zamišljen kao edukativni alat uz pomoć kojega je moguće dublje i kvalitetnije razumjeti tematiku vezanu za prijenos podataka putem računalne mreže. Zbog tog razloga simulacije je potrebno promatrati pri izvršavanju samog programa uz povremena zaustavljanje i čitanje rezultata ispisanih unutar elementa *RichTextBox*. Upravo zbog tog razloga teško je predočiti kvalitetne primjere simulacija unutar ovog rada, odnosno u obliku teksta i slika, no u nastavku će se pokušati što detaljnije predstaviti primjer uspješnog prijenosa datoteke s računala 1 na računalo 2.

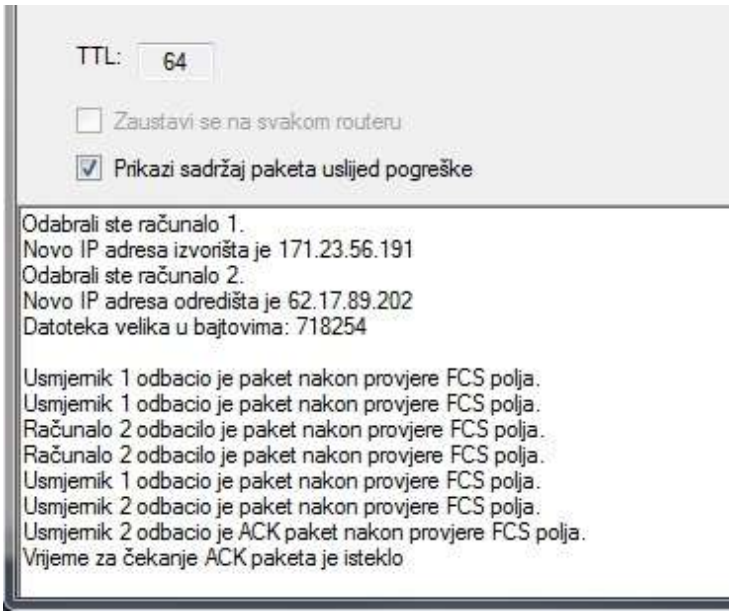
Slika 23. prikazuje simulator tijekom slanja paketa od računala 1 prema računalu 2. Može se vidjeti kako se unutar polja za unos izvora i odredišta nalaze vrijednosti IP adresa računala 1 i 2. Isto tako valja primijetiti kako su vrijednosti BER-a na linkovima između dva računala postavljeni na identičnu vrijednost, a to je 10^{-6}. Na slici 24. prikazan je prijenos ACK paketa od računala 2 prema računalu 1. Isti se prijenos odvija sve dok se ukupno odabrana datoteka ne prenese. U slučaju ove simulacije odabrana je datoteka veličine 718254 oktet. Slika 25. pokazuje tekst koji se ispisuje korisniku ukoliko je uspješno pokrenuo simulaciju, te se tokom nje dogodio određen broj pogrešaka i došlo je do odbacivanja paketa. Pri svakom odbacivanju ispisuje se koji je uređaj ispitao paket, o kojem se paketu radilo i koji je razlog odbacivanja. U najvećem broju slučaja odbacivanje se događa kod provjere FCS polja (vidi poglavlje 2). Prije svakog odbacivanja otvarao se prozor s ispisom sadržaja paketa koji je prikazan na slici 19.



Slika 23. Izgled simulatora tijekom prijenosa podatkovnog paketa

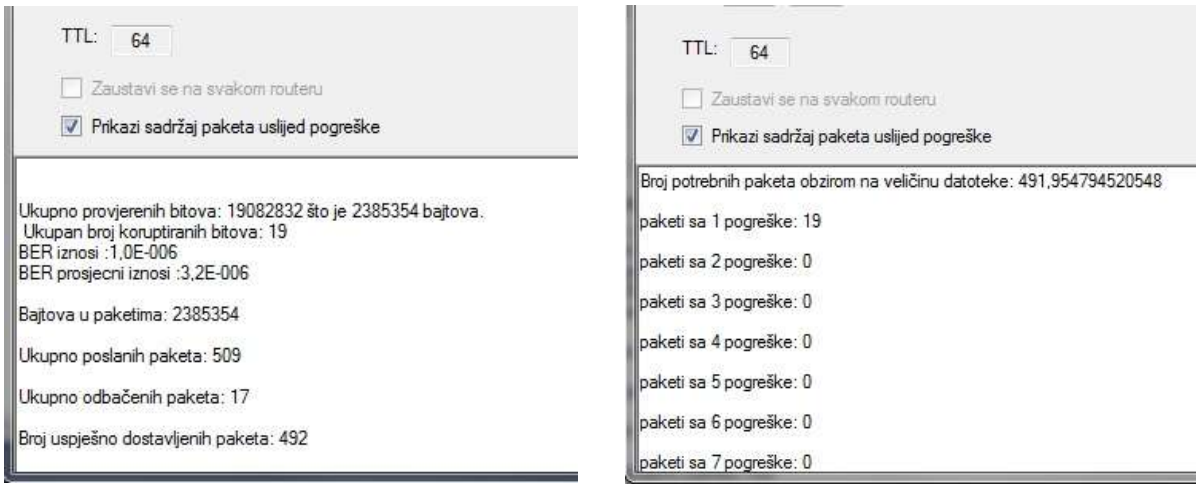


Slika 24. Izgled simulatora tijekom prijenosa ACK paketa



Slika 25. Ispis teksta korisniku tijekom simulacije

Nakon završetka procesa prijenosa svih paketa u elementu *RichTextBox* ispisuje se statistika simulacije što je vidljivo na slici 26.



Slika 26. Završna statistika simulacije

Može se vidjeti koliko je ukupno bitova bilo preneseno, uključujući pakete koji su odbačeni i one koji su zaprimljeni (vrijedi za podatkovne i ACK pakete). Ispod toga vidi se broj ukupno koruptiranih (neispravnih) bitova, koji je u slučaju ove simulacije iznosio 19. Nakon toga ispisane su dvije vrijednosti BER-a. Gornja vrijednost uzima u obzir sve bitove koji su pregledani unutar svakog usmjernika ili računala. Ukoliko se paket sastoji od 10000 bitova, a pregledali su ga jedno računalo i dva usmjernika, tada se za broj pregledanih bitova smatra 30000. Vidljivo je kako je vrijednost tako

izračunatog BER-a identična vrijednosti BER-a zadanoj na svakom linku (vrijednosti su jednake na svim linkovima). Druga vrijednost BER-a uzima u obzir samo broj bitova koje je predajno računalo stvorilo (uključujući i ponovljene pakete). Na neki način se ta vrijednost može predstaviti kao vrijednost BER-a na cijelom putu tog paketa, pa je razumljivo da je ta vrijednost veća.

Ispod vrijednosti BER-a nalazi se broj ukupno poslanih paketa i broj odbačenih paketa. Može se vidjeti kako je broj odbačenih paketa manji nego broj koruptiranih bitova (17 umjesto 19). Dva su moguća uzroka tomu. Prvi uzrok može biti kod pojave paketa s više od jedne pogreške, dakle ako je paket imao 2 pogreške obje se broje kao koruptirani bitovi, no takav paket je odbačen samo jednom. Drugi uzrok je nastanak pogreške unutar ACK paketa, koji se ne ubraja u statistiku brojanja paketa. Ukoliko je nastala promjena bita unutar ACK paketa brojač koruptiranih bitova se povećava (potrebno radi ispravnog izračuna BER-a), ali se ACK paket ne broji kao odbačen, budući ne nosi nikakve podatke. Sljedeće dvije vrijednosti veoma su slične. Prva se odnosi na broj ispravno dostavljenih paketa, dok je druga izračunata vrijednost broja paketa koje je potrebno prenijeti kako bi se odabrana datoteka prenijela u cijelosti. Drugi broj mora biti manji ili jednak prvom broju (mora biti manji za manje od jedan) .

Na kraju su prikazani brojevi paketa s određenim brojem pogrešaka. Budući je broj paketa s jednom pogreškom 19, a ostali su 0 može se zaključiti kako je kod ove simulacije došlo do ukupno 19 pogrešaka i da je svaka bila u drugome paketu, no kako je broj odbačenih paketa 17 znači da su se dvije pogreške dogodile unutra ACK paketa.

Zbog prostorne ograničenosti forme završnog rada u vidu broja stranica, ali i nemogućnosti prikaza određenih dijelova simulacija, u nastavku se samo navode ostale mogućnosti simulatora:

- onemogućavanje pokretanja simulacije uz ispisivanje razloga u vidu obavijesti korisniku ukoliko svi podatci nisu pravilno unijeti
- odbacivanje paketa zbog snižavanja vrijednosti TTL polja na 0
- generiranje povratne poruke usmjernika koji je odbacio paket (imitacija ICMP protokola) (engl. *Internet Control Message Protocol*)

- kruženje paketa unutar mreže ukoliko odredišna adresa ne odgovara adresi niti jednog računala
- simulacija prijenosa datoteke putem mreže uz primjenu HTTPS protokola i pohrana na odredišno računalo (stvaranje datoteke unutar zadanog direktorija).

8. ZAKLJUČAK

Internet je u današnjem svijetu nezamjenjiv te je poznavanje korištenja istog postao neophodan preduvjet pri zapošljavanju na većini radnih mjesta. Iako dublje znanje o načinu na koji radi nije od presudne važnosti za većinu, niti je preduvjet za svakodnevno korištenje Interneta, svejedno je korisno. Zbog složenost i interdisciplinarnosti tog područja pri svladavanju teorijske osnove često su potrebna i vizualna pomagala. Vjerojatno je najbolje pomagalo upravo simulator, koji zbog načina rada može pojasniti, naizgled komplicirane stvari, na veoma jednostavan način. Naravno ukoliko se zahtijeva prikaz može biti i kompleksniji, ovisno o predznanju i željama korisnika.

Većina utrošenog vremena pri stvaranja završnog rada uložena je upravo u izradu simulatora, koji je kako je već nekoliko puta navedeno, neizostavan i podjednako važan dio ovog završnog rada te zajedno s ovim dokumentacijskim dijelom tvori cjelinu. Za potpuno razumijevanje rada potrebno je određeno predznanje: poznavanje rada s programskim jezikom C#, način funkcioniranja mreža, odnosno prijenosa podataka kroz sve elemente određene računalne mreže, koja je uzeta kao primjer u ovom završnom radu.

Simulator unutar ovog završnog rada, kako je već mnogo puta kroz sam rad navođeno, zamišljen je da pojasni ponašanje paketa koji putuju računalnom mrežom. Od njegova stvaranja, postavljanja vrijednosti unutar zaglavlja svih slojeva, prijenosa i izmjena njegova sadržaja na putu kroz mrežu, stvaranja pogreški uslijed smetnji na prijenosnom mediju, odbacivanja samog paketa i slanja potvrde o zaprimljenom paketu. Iako naizgled malen dio teorije cijelog tog složenog multidisciplinarnog sustava, ovo je važan i neizostavan dio i kvalitetna podloga za razumijevanje složenijih i naprednijih funkcija mreže.

LITERATURA

[1] Internetski izvor:
https://gikk.ru/bs/internet/knowledge-of-the-osi-model-how-the-osi-model-works/
[Pristupano: rujan 2018.]

[2] Internetski izvor: PCCHIP
https://pcchip.hr/wp-content/uploads/2017/01/Usporedba-izme%c4%91u-OSI-i-TCPIP-modela.jpg [Pristupljeno: rujan 2018.]

[3] Internetski izvor: Lifewire
https://www.lifewire.com/tcp-headers-and-udp-headers-explained-817970
[Pristupljeno: rujan 2018.]

[4] Internetski izvor:
https://www.whichvoip.com/blog/what-is-an-ip-packet.html
[Pristupljeno: rujan 2018.]

[5] Internetski izvor: Oxford Dictionaries
https://en.oxforddictionaries.com [Pristupljeno: rujan 2018.]

[6] Internetski izvor: Stack Overflow
https://insights.stackoverflow.com/survey/2016 [Pristupljeno: rujan 2018.]

[7] Internetski izvor:
https://medium.com/sololearn/why-is-c-among-the-most-popular-programming-languages-in-the-world-ccf26824ffcb [Pristupljeno: rujan 2018.]

[8] Internetski izvor: Microsoft
https://docs.microsoft.com/en-us/dotnet/csharp/whats-new/
[Pristupljeno: rujan 2018.]

[9] Internetski izvor: NET MARKETSHARE
https://www.netmarketshare.com/operating-system-market-share.aspx
[Pristupljeno: rujan 2018.]

POPIS KRATICA I AKRONIMA

ACK (engl. *Acknowledgment*)

B (engl. *Byte*)

BER (engl. *Bit Error Rate/Ratio*)

BGP (engl. *Border Gateway Protocol*)

BW (engl. *BackgroundWorker*)

CRC (engl. *Cyclic Redundancy*)

DARPA (engl. *Defence Advanced Research Projects Agency*)

DHCP (engl. *Dynamic Host Configuration Protocol*)

DNS (engl. *Domain Name System*)

FCS (engl. *Frame Check Sequence*)

FIN (engl. *Finish*)

FTP (engl. *File Transfer Protocol*)

GB (engl. *Gigabyte*)

HTTP (engl. *HyperText Transfer Protocol*)

HTTPS (engl. *HyperText Transfer Protocol Secure*)

IANA (engl. *Internet Assigned Numbers Authority*)

ICMP (engl. *Internet Control Message Protocol*)

IHL (engl. *Internet Header Length*)

IMAP (engl. *Internet Message Access Protocol*)

IP (engl. *Internet Protocol*)

IPv4 (engl. *Internet Protocol version 4*)

IPv6 (engl. *Internet Protocol version 6*)

kB (engl. *kilobyte*)

LAN (engl. *Local Area Network*)

MAC (engl. *Media Access Control*)

OSI (engl. *Open System Interconnection*)

OSPF (engl. *Open Shortest Path First*)

POP3 (engl. *Post Office Protocol version 3*)

RIP (engl. *Routing Information Protocol*)

SMTP (engl. *Simple Mail Transfer Protocol*)

SYN (engl. *Synchronise*)

TCP (engl. *Transmission Control Protocol*)

TTL (engl. *Time To Live*)

UDP (engl. *User Datagram Protocol*)

URG (engl. *Urgent*)

WAN (engl. *Wide Area Network*)

POPIS SLIKA

Sveučilište u Zagrebu
Fakultet prometnih znanosti
10000 Zagreb
Vukelićeva 4

IZJAVA O AKADEMSKOJ ČESTITOSTI I SUGLASNOST

Izjavljujem i svojim potpisom potvrđujem kako je ovaj završni rad isključivo rezultat mog vlastitog rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu što pokazuju korištene bilješke i bibliografija.

Izjavljujem kako nijedan dio rada nije napisan na nedozvoljen način, niti je prepisan iz necitiranog rada, te nijedan dio rada ne krši bilo čija autorska prava.

Izjavljujem također, kako nijedan dio rada nije iskorišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

Svojim potpisom potvrđujem i dajem suglasnost za javnu objavu završnog rada pod naslovom **Razvoj simulatora prijenosa paketa računalne mreže**

na internetskim stranicama i repozitoriju Fakulteta prometnih znanosti, Digitalnom akademskom repozitoriju (DAR) pri Nacionalnoj i sveučilišnoj knjižnici u Zagrebu.

Student: Mateo Posavec

U Zagrebu, 12.9.2018

(potpis)