

UNIVERZITET U NIŠU  
ELEKTRONSKI FAKULTET

## **ZAVRŠNI RAD**

**Implementacija prikrivenih kanala u TCP/IP protokol steku**

**Miloš Mladenović**

Niš, septembar 2017. god.

## ZAVRŠNI RAD

### Implementacija prikrivenih kanala u TCP/IP protokol steku

**Zadatak:** Proučiti prikrivene kanale na računarskim mrežama. Posebno sagledati način komunikacije prikrivenim kanalima koji manipulišu zaglavljima mrežnih paketa. Upoznati se sa *raw* Internet soketima. Projektovati i implementirati aplikaciju za komuniciranje prikrivenim kanalima između dva računara na mreži. Izvršiti evaluaciju rada aplikacije slanjem poruka između dva računara na mreži preko svih predloženih kanala.

**Mentor:**

Prof. dr Vladimir Ćirić

**Kandidat:**

Miloš Mladenović, broj indeksa: 14809

Komisija za ocenu i odbranu završnog rada:

1.

---

Datum prijave završnog rada:

2.

---

Datum predaje završnog rada:

3.

---

Datum odbrane završnog rada:

# Sadržaj

|  |    |
|--|----|
| 1. Uvod.....   | 4  |
| 2. Računarske mreže i osnove Interneta.....                        | 6  |
| 2.1. Razvoj Interneta kroz istoriju.....                           | 7  |
| 2.2. Korišćenje Interneta .....                                    | 8  |
| 2.3. Hijerarhija protokola .....                                   | 9  |
| 3. Referentni model OSI .....                                      | 11 |
| 3.1. Fizički sloj.....   | 12 |
| 3.2. Sloj veze podataka .....                                      | 12 |
| 3.2.1. Ethernet .....  | 13 |
| 3.3. Mrežni sloj .....   | 14 |
| 3.3.1. IP protokol.....  | 15 |
| 3.4. Transportni sloj .....  | 17 |
| 3.4.1. TCP (Transmission Control Protocol) .....                   | 17 |
| 3.4.2. UDP (User Datagram Protocol) .....                          | 19 |
| 3.5. Sloj sesije .....   | 20 |
| 3.6. Sloj prezentacije .....                                       | 20 |
| 3.7. Aplikativni sloj.....   | 20 |
| 4. Sigurna komunikacija na Internetu .....                         | 20 |
| 4.1.    Prikriveni kanali za komunikaciju na Internetu .....       | 21 |
| 4.1.1.    Primena prikrivenih kanala .....                         | 22 |
| 4.1.2.    Vrste prikrivenih kanala i načini komunikacije .....     | 23 |
| 4.1.3.    Postojeći prikriveni kanali .....                        | 24 |
| 5. Programiranje soketa i raw soketi u Linuxu .....                | 26 |
| 5.1. Životni vek paketa u mrežnom steku kod klasičnih soketa ..... | 27 |
| 5.2. Raw soketi.....   | 28 |
| 6. Implementacija prikrivenih kanala.....                          | 33 |
| 6.1. Arhitektura aplikacije.....                                   | 33 |
| 6.2. Pogled na slučajeve korišćenja .....                          | 33 |
| 6.3. Pogled na logičku arhitekturu sistema .....                   | 34 |
| 6.4. Pogled na implementaciju sistema.....                         | 35 |
| 6.4.1. covert_sender .....   | 35 |
| 6.4.2. covert_receiver.....  | 36 |

|  |    |
|--|----|
| 6.5. Pogled na osnovne funkcionalnosti.....              | 38 |
| 6.5.1. Odabir moda rada.....                             | 38 |
| 6.5.2. Kreiranje i popuna paketa za slanje na mrežu..... | 39 |
| 6.5.3. Slanje prikrivenim kanalom .....                  | 40 |
| 6.5.4. Prijem prikrivenim kanalom .....                  | 41 |
| 6.6. Primer implementacije prikrivenih kanala .....      | 43 |
| 7. Zaključak.....  | 47 |
| 8. Spisak slika .....                                    | 48 |
| 9. Literatura .....                                      | 49 |

# 1. Uvod

Od samih početaka ljudske civilizacije, potreba za komuniciranjem je bila jedna od osnovnih ljudskih potreba i oblast čijim je razvitkom napredovalo i samo čovečanstvo. Sredstva komunikacije su se menjala i razvijala – korišćene su reči, dimni signali, pisanje na pergamentu i slično. Napretkom civilizacije došlo je do značajnih pomaka i promena u načinu komuniciranja i prenosu podataka. Nova otkrića kao što su telegraf, telefon, elektronski računar i tranzistor donela su značajne pomake na polju komunikacije.

Prvi računari opšte namene su bili sastavljeni od vakuumskih cevi kao aktivnih elemenata i kablova kao veza između njih. Oni su bili skupi, trošili mnogo energije, zauzimali ogroman prostor i korišćeni su uglavnom za složena matematička izračunavanja. Pronalaskom tranzistora omogućeno je da se proizvode manji računari, sa manjim energetskeim zahtevima i prilagođeniji većem broju korisnika. Sa rastom broja računara rasli su i zahtevi korisnika i javila se potreba da oni međusobno razmenjuju informacije, pa je tako došlo do formiranja računarskih mreža. U prvim decenijama postojanja računarske mreže su uglavnom koristili istraživači sa univerziteta da razmenjuju elektronsku poštu i zaposleni u kompanijama da razmenjuju manje podatke ili zajednički koriste manji broj zajedničkih štampača. Bezbednost i skrivanje ili zaštita komunikacije tada nisu bili od velike važnosti. Međutim, u modernim vremenima, kada se računari koriste u svim oblastima ljudske delatnosti i ogromne novčane transakcije se obavljaju putem Interneta, bezbednost, odnosno zaštita i skrivanje informacija od neželjenih strana je od ključne važnosti.

Sigurna komunikacija na Internetu podrazumeva komuniciranje dva entiteta koji ne žele da budu prisluškivani od treće strane. Da bi ovo postigli, moraju da komuniciraju na način koji nije podložan prisluškivanju ili presretanju [1]. Postoje dve metode koje se koriste kako bi se postigla ova vrsta komunikacije: *kriptografija*, koja omogućava da se zaštititi sadržaj poruke od neovlašćene strane i *prikrivena komunikacija* koja sakriva samo postojanje komunikacije kako bi bilo sprečeno da neovlašćena strana detektuje postojanje komunikacije [2].

Sigurna komunikacija putem *prikrivenih kanala* (eng. *Covert channels*) obuhvata skrivenu komunikaciju koja štiti informacije koje se prenose, kao i vezu između pošiljaoca i primaoca poruke. *prikriveni kanali* u okviru pojma sigurnosti informacija podrazumevaju kanale za komunikaciju nestandardnim putevima, preko procesa kojima to nije prvobitna namena. Ovi kanali se zovu *prikriveni* zato što su prikriveni od mehanizama kontrole pristupa visoko-sigurnih operativnih sistema jer ne koriste klasične, legitimne načine za prenos željenih podataka kao što su pisanje i čitanje, pa stoga ne mogu biti detektovani i kontrolisani od strane hardverski baziranih sigurnosnih mehanizama u operativnim sistemima. Prikriveni kanali na Internetu mogu da budu implementirani tako što će klasične protokole koristiti na nekonvencionalan način, umetanjem podataka u zaglavlja okvira ili izmenom postojećih zaglavlja tako da odgovaraju validnom mrežnom saobraćaju i saglasni su sa ostalim podacima, a da samo pošiljalac i primalac na krajevima kanala budu svesni tajne komunikacije i kodiraju/dekodiraju bitove na odgovarajući način.

Pristupanje komunikaciji na mreži od strane samog korisnika i aplikacije se vrši pomoću tzv. *utičnice* (eng. *sockets*) koji predstavljaju krajnju tačku dvosmernog međuprocenog komunikacionog toka – *API* (*Application Programming Interface*) koji je obično podržan od strane OS-a. Kako bi bili iskorišćeni svi potencijali *prikrivenih kanala* za komunikaciju i isti bili praktično implementirani, potrebno je programiranje soketa na niskom nivou (tzv. *raw soketi*) - kada je programeru dozvoljena puna kontrola nad onim što će biti zapisano u zaglavlja mrežnih

paketa koji će biti poslati, kao i nad okruženjem u kome se piše dati program – operativni sistem čiji zaštitni zid (*firewall*) to dozvoljava.

Cilj ovog rada je proučavanje rada *prikrivenih kanala* za komunikaciju u mreži i projektovanje aplikacije koja ilustruje rad ovih kanala. Implementacijom kanala biće omogućeno da krajnji računari u lokalnoj mreži mogu da komuniciraju bez mogućnosti da njihova komunikacija može sa sigurnošću bude otkrivena i dešifrovana “običnim” nadgledanjem mreže i ukoliko ne postoji pretpostavka da je baš određeni tip kanala uspostavljen. U radu će najpre biti posvećena pažnja računarskim mrežama i najbitnijim protokolima, zatim će biti predstavljen sam pojam prikrivenih kanala i neke od češćih implementacija, nakon čega sledi i opis programiranja “raw” soketa, čime se omogućuje dublji pristup formiranju mrežnih paketa. Na kraju biće dat primer rada projektovane aplikacije i kodiranje, prenošenje i dekodiranje poruka između dva računara na lokalnoj mreži.

Nakon uvoda, u drugom poglavlju, biće objašnjen pojam računarskih mreža, Interneta, njegov kratak istorijat i hijerarhija protokola.

U trećem poglavlju biće predstavljen OSI referentni model, kao i protokoli koji su najznačajniji za izradu aplikacije uz opis odgovarajućih zaglavlja i načina rada.

Iza toga sledi opis sigurne komunikacije na Internetu, prikrivenih kanala, načina za analizu saobraćaja itd.

U petom poglavlju biće objašnjeni *raw* soketi i ograničenja odnosno mogućnosti koje pruža programiranje soketa u različitim operativnim sistemima, sa fokusom na Linux.

U poslednjem, šestom poglavlju biće predstavljena aplikacija projektovana za komunikaciju prikrivenim kanalima. Biće dat pogled na njenu logičku arhitekturu i opis svih implementiranih kanala i njihovog načina rada, prednosti i mana. Na kraju će biti dat primer rada aplikacije.

Zaključak će biti dat u sedmom poglavlju.

## 2. Računarske mreže i osnove Interneta

U prethodnim vekovima došlo je do ključnih promena i otkrića značajnih za celu civilizaciju, razvile su se tehnologije koje su bile aktuelne po ceo jedan vek i obeležile ga. U osamnaestom veku to su bili veliki mehanički sistemi, devetnaesti vek je doneo otkriće parne mašine, a 20. vek je bio doba prikupljanja, skladištenja, obrade i distribuiranja podataka. Ovakvu veliku revoluciju u prethodnom stoleću doživeli smo zahvaljujući računarima i njihovom međusobnom umrežavanju – mogućnosti da se brzo sa jednog kraja sveta na drugi prenesu sve informacije koje želimo. Sposobnost i želja za prikupljanjem, obradom i distribuiranjem podataka rastu svakim danom.

Stapanje računara sa komunikacijama je imalo snažan efekat na način organizovanja računarskih sistema. Ideja „računarskog centra“ – prostorije sa velikim računarom u koju korisnici donose svoje podatke na obradu – potpuno je prevaziđena. Stari model po kome je jedan računar zadovoljavao sve potrebe organizacije zamenjen je modelom u kome posao obavlja veći broj zasebnih, ali međusobno povezanih računara. Takvi sistemi su nazvani *računarske mreže* [3].

Računarska mreža je pojam koji se odnosi na računare i druge uređaje povezane međusobno u jednu celinu kablovima ili na drugi način, a sve u svrhu komunikacije, razmene i deljenja informacija. Mrežom je moguće prenositi računarske podatke, govor, sliku ili video. Programi koje korisnici mreže koriste mogu biti takvi da se zahteva i prenos podataka u realnom vremenu (govor, video i sl.). Računarska mreža se sastoji iz računara (čvorova), komunikacionih medijuma (žica, optičko vlakno, vazduh i sl.) i uređaja koji čine mrežnu infrastrukturu – habova, svičeva, rutera, bridževa.

Osnovni elementi računarskih mreža i komunikacije jesu: komunikacioni kanal, hardver, operativni sistem i korisničke aplikacije. Prilikom direktne komunikacije dva računara – oba poseduju sve elemente. Međutim, kada više računara učestvuje u komunikaciji i na velikoj su fizičkoj udaljenosti, ona se odvija putem posrednika – mrežnih uređaja, koji mogu biti jednostavniji i sastojati se samo od hardvera sa ugrađenim interfejsima i funkcijama, a mogu biti i složeni, takvi da poseduju sopstveni mrežni operativni softver. Elementi, tj. uređaji koji učestvuju u povezivanju računara sa komunikacionim kanalom i interfejsima na drugim uređajima nazivaju se pasivnom mrežnom opremom, dok se oni uređaji koji u sebi sadrže, sem hardvera, i operativni sistem, nazivaju aktivnom mrežnom opremom. Softversku celinu mreže čine protokoli (definisana pravila po kojima se obavlja komunikacija u mreži), operativni sistemi koji omogućavaju komunikaciju sa hardverskim delom sistema i korisnički mrežni softver [4].

Internet je globalni skup povezanih različitih računarskih mreža koje koriste neke zajedničke protokole i obezbeđuju slične usluge. To je „mreža svih mreža“ koja se sastoji od privatnih, javnih, akademskih, poslovnih i vladinih mreža lokalnog i globalnog značaja, povezana raznim vidovima elektronskih, bežičnih i optičkih tehnologija. On je jedinstven po tome što njime niko ne upravlja, potpuno je distribuiran i dostupan i u najudaljenijim delovima sveta. Internetom se prenosi široki spektar informacija i služi za funkcionisanje mnogih servisa – dostupnost međusobno povezanih hipertekst dokumenata i aplikacija WWW-a, elektronske pošte, telefonije i deljenja fajlova. Posebno je značajna njegova upotreba u poslovnom okruženju, gde veliki broj računara u firmama kao i njihovi klijenti koriste Internet servise za razmenu, prikupljanje i obradu podataka.

Prva primena Interneta u poslovne svrhe pokrenuta je od strane velikih finansijskih institucija. Njih su u tome ispratili velike firme povezujući svoje prodajne i distribucione mreže sa Internetom. On se odmah ponudio kao idealno rešenje jer je za svaku veliku poslovnu organizaciju brzina i kvalitet razmene informacija automatski značila veći profit, a često i opstanak na tržištu.

## 2.1. Razvoj Interneta kroz istoriju

Istorija Interneta započinje sa otkrićem i razvojem paketnog prebacivanja (eng. *packet switching*), tj. grupisanja podataka koje želimo da pošaljemo u formatirane jedinice podataka – pakete.

Razvoj primitivnih oblika Interneta počinje krajem pedesetih godina 20. veka. Ministarstvo odbrane SAD-a, u jeku hladnog rada, pozelelo je da dotadašnji model komunikacija koji postoji u okviru državnih institucija, a koji se zasnivao na telefonskim centralama i komunikacionim linijama unapredi, zato što je bio veoma ranjiv i infrastruktura je mogla biti lako uništena, čime bi ceo sistem mogao biti podeljen na više izolovanih ostrvaca. Kako bi ovaj problem bio rešen u okviru državnih institucija, osnovana je organizacija za poslove odbrane – DARPA (Defense Advanced Research Project Agency). Jedan od njenih zadataka bio je da se pronade i uspostavi mreža koja se zasniva na sistemu paketnog prenosa i koja bi povezivala i akademske i državne institucije i bila otporna na otkaze.

Odgovor organizacije DARPA bio je ARPANET. Nova mreža sastojala se od mini računara zvanih *obrađivači poruka na interfejsu* (eng. IMPs – Interface Message Processors), povezanih linijama brzine prenosa 56 kb/s. Radi veće pouzdanosti, svaki IMP bi bio povezan s barem još dva druga IMP-a. Podmreža bi bila datagramska, kako bi u slučaju uništenja nekih linija i IMP-a poruka mogla da se automatski preusmeri drugom putanjom. Svaki čvor mreže sastojao bi se od IMP-a i umreženog računara, smeštenih u zajedničku prostoriju i povezanih kratkim kablom. Računar bi IMP-u mogao da šalje poruke veličine do 8063 bita koje bi ovaj razbijao na pakete veličine 1008 bitova i nezavisno prosleđivao ka odredištu. Svaki paket bi pre daljeg prosleđivanja morao biti primljen u celini, tako da podmreža koju je zamislio Roberts predstavlja prvu mrežu po sistemu komutiranja „čuvaj i prosledi“ (eng. *store and forward*) [3].

Eksperimentalna mreža koja je puštena u rad 1969. godine imala je četiri čvora: UCLA (University of California, Los Angeles), UCSB (University of California, Santa Barbara), SRI (Stanford Research Institute) i Univerzitet Jute. Ove institucije su bile izabrane jer su posedovale ugovore sa DARPA-om i imale različite, nekompatibilne računare. Mreža je nakon toga veoma brzo rasla i polako su pokrivene sve važnije akademske institucije u SAD-u, za naredne 3 godine. Dalje tokom sedamdesetih godina 20. veka došlo je do povezivanja sve više univerziteta i institucija. Broj mreža, računara i korisnika priključenih na ARPANET brzo je rastao nakon što je 1. januara 1983. godine TCP/IP postao jedini zvanični skup protokola. Protokoli su standardi koji omogućavaju komunikaciju računara putem mreže, a sve do uspostavljanja TCP/IP skupa protokola, jedini koji se koristio bio je primitivni NCP (Network Control Protocol).

Uporedo sa ARPANET-om, NSF (National Science Foundation) je razvila svoju mrežu – NSFNET kojoj bi se mogli priključiti i sve istraživačke stanice i univerziteti koji nisu imali ugovor sa Ministarstvom odbrane, a koji su želeli da učestvuju u razmeni informacija. Prvobitna namena ove mreže slične ARPANET-u po infrastrukturi bila je da manje istraživačke centre poveže sa



velikim superračunarima u 6 centara: San Dijegu, Bulderu, Šampanji, Pittsburgu, Itaki i Princetonu. Hardverski je bila zasnovana na istoj infrastrukturi korišćenoj u ARPANET-u, sa brzinom 56 kb/s, dok se od njega razlikovala po tome što su se računari sporazumevali protokolom TCP/IP od početka svog rada. Veći rast mreže uslovio je i to da Vlada više nije mogla da finansira i njeno napredovanje, nego su želeli da iznajmljuju svoju infrastrukturu komercijalnim organizacijama. Ovo nije bilo u početku moguće jer su to zabranjivali propisi NSF-a. Zbog toga je NSF nagovorila kompanije MERIT, MCI i IBM da formiraju neprofitnu korporaciju ANS (*Advanced Networks and Services*), kao prvi korak ka komercijalizaciji. Kasnije je ANS preuzela NSFNET i formirala novu mrežu sa poboljšanim karakteristikama – ANSNET.

Kada su se NSFNET i ARPANET međusobno povezali, ovoj novoj mreži priključile su se mnoge regionalne mreže, a uspostavljene su i veze sa mrežama u Kanadi, Evropi i na Pacifiku. Polako je cela infrastruktura prerasla u Međusobno povezani sistem mreža – Internet. Glavna ideja na kojoj se zasniva Internet je otvorena arhitektura umrežavanja u kojoj individualne mreže mogu da se redizajniraju i razvijaju nezavisno i svaka ima interfejs i skup usluga koji nudi korisnicima i drugim mrežama. Vezivno tkivo koje drži Internet na okupu jeste skup protokola TCP/IP, koji je postao jedini standard i omogućava davanje univerzalne usluge. „Biti na Internetu“ zapravo znači da računar izvršava skup protokola TCP/IP, poseduje IP adresu i može da šalje IP pakete svim drugim računarima na Internetu.

## 2.2. Korišćenje Interneta

U doba nastanka i razvoja Interneta, u periodu između 1970. i 1990. godine usluge koje je on nudio mogle su da se svrstaju u nekoliko kategorija:

1. **E-pošta** (eng. *e-mail*) – mogućnost da se sastavi, pošalje i primi elektronska poruka. Postojala je od prvih dana ARPANET-a i danas je najpopularniji oblik komunikacije na računarskim mrežama.
2. **Diskusione grupe** (eng. *newsgroups*) – specijalizovani forumi kroz koje korisnici istih interesovanja mogu da razmene poruke i mišljenja. Prisutni su forumi koji se tiču svih sfera života, posvećeni tehničkim temama, nauci, rekreaciji, politici itd.
3. **Daljinsko upravljanje** (eng. *remote login*) – pomoću programa telnet, rlogin ili ssh korisnici koji su povezani na Internet mogu da se prijave na bilo koji drugi računar na kome imaju otvoren nalog.
4. **Prenos datoteka** (eng. *file transfer*) – pomoću protokola FTP, korisnici Interneta mogu da kopiraju datoteke s jednog računara na drugi. Tako se mnogi članci, baze podataka i druge informacije dele među više korisnika.

Do početka devedesetih godina, na Internetu su bile prisutne uglavnom ove usluge i glavni njegovi korisnici bili su istraživači, pripadnici bezbednosnih snaga ili zaposleni u kompanijama koje su isti koristile. Sve se ovo izmenilo, kada je nađena nova oblast primene – **WWW (World Wide Web)**. WWW je osmislio fizičar Tim Berners-Lee na institutu CERN 1989. Godine, a Marc Andressen iz Nacionalnog centra za superračunarske aplikacije u Urbani (Ilinois) je kreirao

Mosaic – prvi čitač Web-a, aplikaciju koja je tumačila Web stranice. **WWW** je informacijski prostor gde su dokumenti i drugi *web resursi* identifikovani URL-om (eng. *uniform resource locator* ), međusobno povezani pomoću hiperlinkova i moguće im je pristupiti preko Interneta [5]. WWW je omogućio da se na mrežnoj lokaciji napravi skup informativnih strana sa tekstom, slikama, zvukom i videom i sa ugrađenim vezama ka drugim stranama (hiperlinkovi). Razvoj Web-a doprineo je tome da Internet dobije ogroman broj novih korisnika, kojima je tako omogućeno lakše komuniciranje i dobijen je smisao korišćenja Interneta i za obične korisnike.

Glavni posrednik prilikom uspostavljanja veze sa Internetom običnim korisnicima postali su tzv. *davaoci Internet usluga* (Internet Service Providers, ISPs). To su firme koje privatnim licima omogućavaju povezivanje na Internet tako što će se kućnim računarom povezati na neki računar ISP-a i tako dobiti pristup Internetu. Ovakav razvoj događaja je potpuno promenio sliku Interneta koji je sada istinski postao globalan i dostupan svima, a ne samo pripadnicima akademskih krugova.

Jugoslavija se uključila u mrežu EARN (*European Academic and Research Network*) 1989. Godine. Prvi računar koji je uključen u mrežu bio je računar Republičkog zavoda za statistiku sa kojim je tadašnji PMF imao ugovor o saradnji i čije je računarske resurse koristio [6].

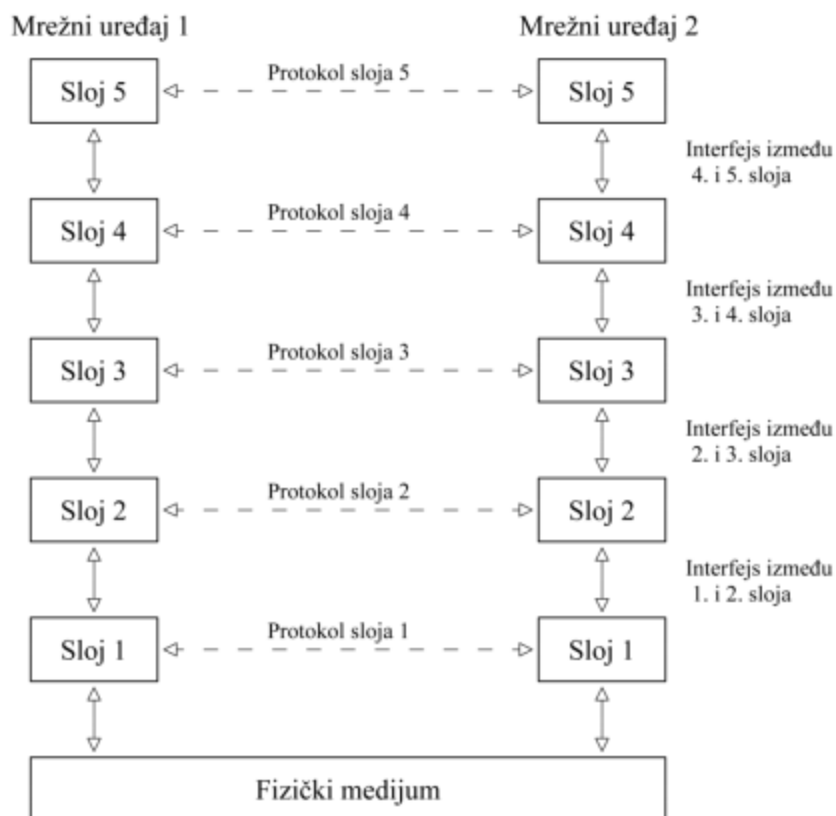
## 2.3. Hijerarhija protokola

Kako bi komunikacija između računara bila lakša i efikasnija, otpornija na otkaze i mogla neometano i nezavisno da se razvija, mreže se uglavnom projektuju tako da budu podeljene na slojeve (eng. *layers* ). Svaki sloj nudi određene usluge slojevima iznad sebe i koristi usluge slojeva ispod sebe, bez potrebe da poznaje način rada drugih slojeva sem svog. Zbog ovoga na svakom sloju su razvijeni interfejsi koji pružaju direktnu uslugu slojevima iznad i samo preko interfejsa slojevi imaju kontakt.

Svaki sloj na jednoj strani mreže komunicira direktno sa odgovarajućim slojem na drugoj strani mreže i ova komunikacija se obavlja uz poštovanje *protokola* – koji predstavlja dogovor između dve jedinice o tome kako da se obavlja komunikacija i kako će signali emitovani sa jedne strane biti protumačeni na drugoj strani. Organizacija mreže na slojeve uz poštovanje protokola omogućava da se svaki sloj nezavisno razvija i unapređuje, proširuje, a da to ne utiče mnogo na ostale slojeve i da se zadrži jedinstvenost početne organizacije, koja je jako bitna.

Prilikom organizacije i uspostavljanja slojeva, podaci se nikad ne prenose direktno od jednog sloja do drugog, nego se komunikacija obavlja kroz *hijerarhiju* slojeva, svaki podatak mora da prođe od najvišeg do najnižeg sloja kako bi stigao na dnu do fizičkog medijuma, preko koga biva prenesen na drugu stranu, gde takođe prolazi kroz sve slojeve u obrnutom redosledu.

Na *slici 1* prikazana je mreža sa pet slojeva. Za elemente odgovarajućih slojeva na različitim računarima kaže se da su ravnopravni (eng. *peers* ). Ravnopravni elementi mogu da budu procesi, hardverski uređaji, čak i ljudi [3].



Slika 1: Slojevi, protokoli i interfejsi [3]

Kada projektanti odlučuju o broju slojeva u mreži i njihovoj funkciji, najvažnije je da definišu jasne interfejsse između slojeva. Da bi se to postiglo, svaki sloj mora da izvršava određen skup funkcija s tačno definisanom namenom. Osim što smanjuje količinu podataka koja se mora prosleđivati između slojeva, precizno definisan interfejs olakšava i izmenu konstrukcije sloja, jer se od nove verzije sloja zahteva samo da sloju iznad sebe ponudi isti skup usluga kao i ranije.

Skup slojeva i protokola naziva se zajedničkim imenom **arhitektura mreže**. Specifikacija arhitekture mora da sadrži dovoljno informacija kako bi realizator mogao da za svaki sloj napiše program ili projektuje hardver koji će slediti pravila odgovarajućeg protokola. Ni detalji realizacije ni specifikacija interfejsa nisu deo arhitekture jer se ne vide spolja – oni su prikriiveni u računarima. Čak nije neophodno da interfejsi na svim umreženim računarima budu isti, pod uslovom da svaki računar ispravno koristi sve protokole. Lista protokola koju koristi određeni sistem (jedan protokol po sloju), naziva se **skup protokola** (eng. *protocol stack*).

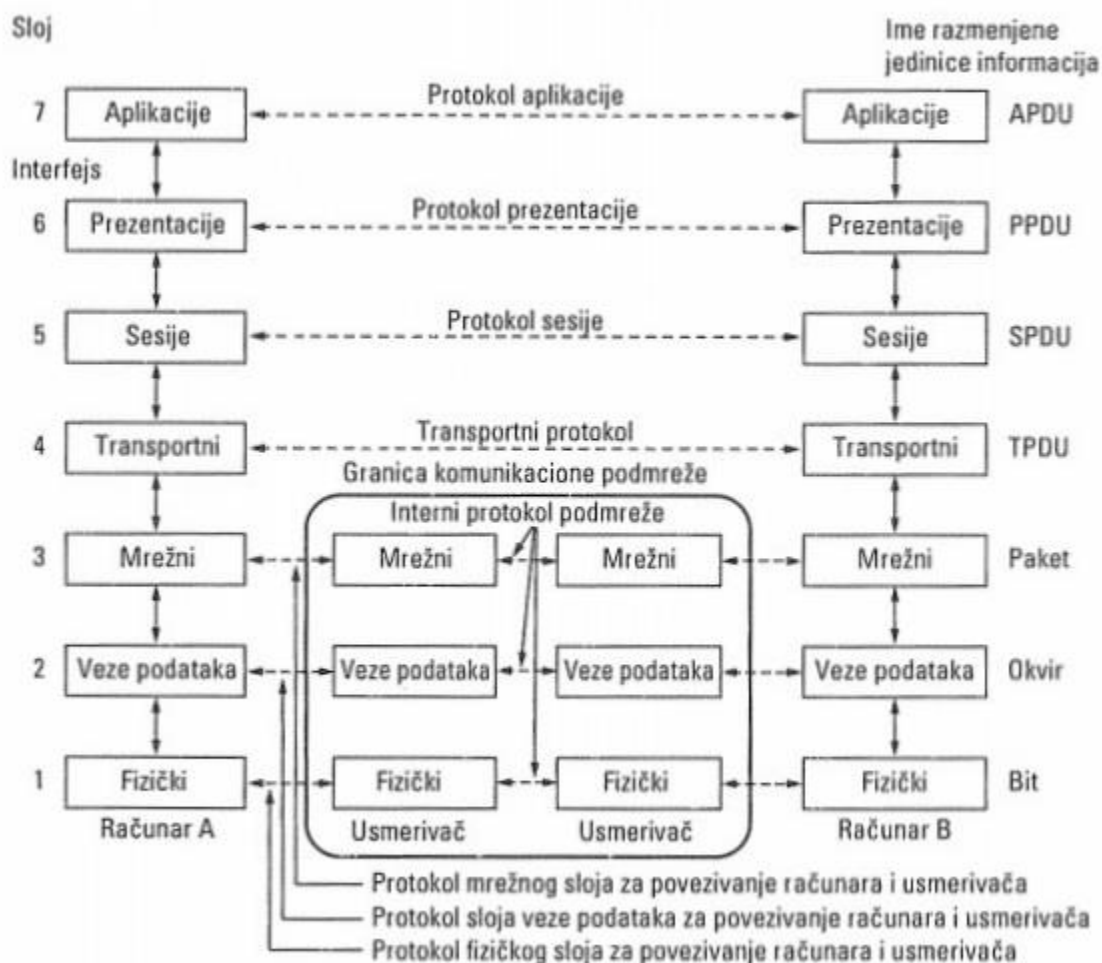
U narednom poglavlju biće opisana arhitektura mreže po referentnom modelu OSI, pri čijem su kreiranju i implementaciji poštovani principi komunikacije u računarskim mrežama i protokolima, primena hijerarhija navedena u prethodnom pasusu. U okviru opisivanja slojeva modela OSI biće predstavljeni i odgovarajući protokoli iz TCP/IP referentnog modela, a koji se upotrebljavaju u navedenom sloju OSI modela.

### 3. Referentni model OSI

U ovom poglavlju će biti reči o referentnom modelu za mrežu – OSI, biće objašnjeni njegovi slojevi svaki ponaosob i akcenat će biti stavljen na protokole koji se u njemu koriste, a koji su bitni za samo njegovo funkcionisanje, ali i cilj ovog rada.

Skup protokola čija je namena pri kreiranju bila da poveže sisteme različitih arhitektura naziva se otvoreni sistem. Prvi korak ka međunarodnoj standardizaciji protokola koji se koriste u različitim slojevima mreža bio je *Referentni model ISO OSI* (eng. *Interenational Standards Organization – Open System Interconnection* ), koji je razvijen sa ciljem da poveže sve sisteme koji su otvoreni za komuniciranje sa drugima. Ovaj model je razvila organizacija ISO, koja predstavlja jedno od najvažnijih svetskih tela za standardizaciju.

Radi razumevanja OSI modela potrebno je razumeti i svaku komponentu ponaosob, kao i njihov međusobni odnos i ulogu u celom sistemu. Na *slici 2* je grafički predstavljen OSI model, dok je u narednim poglavljima opisan svaki njegov sloj ponaosob.



Slika 2: Referentni model OSI i komunikacija među slojevima [3]

Principi na kojima se zasniva funkcionisanje modela OSI jesu da svaki sloj izvršava zasebne funkcije i ne vodi računa o tome kako ostali slojevi izvršavaju svoje servise, već je bitno samo da li su oni izvršeni. Pritom svaki put kada treba da postoji nova apstrakcija funkcionalnosti, potrebno je napraviti novi sloj, a funkciju svakog sloja treba izabrati imajući u vidu definisanje međunarodno standardizovanih protokola. Još jedno od svojstava ovog modela jeste da prilikom organizacije svakog sloja i definisanja granica između slojeva treba se voditi time da se minimizuje protok podataka između slojeva i da što veći broj funkcija nad podacima može da se obavi bez razmene sa ostalim slojevima.

Kao što je na slici iznad prikazano, isprekidanim linijama je naglašena tzv. *prividna* komunikacija među slojevima, koja se odnosi na to da svaki sloj razmenjuje podatke sa odgovarajućim slojem na drugoj strani komunikacije, ali se to ne obavlja direktno, nego preko posrednika – što su slojevi ispod i iznad, tako da je taj tok podataka u stvari *logički*, a jedini sloj koji vrši direktnu, *fizičku* komunikaciju sa suprotnom stranom jeste fizički sloj. Još jedna od karakteristika ovog modela jeste da *usmerivači*, koji su posrednici u komunikaciji između računara i služe za usmeravanje saobraćaja kroz mrežu, implementiraju prva tri sloja OSI modela – fizički, sloj veze podataka i mrežni.

U nastavku biće opisani svi slojevi redom, kao i najvažniji protokoli sa odgovarajućim poljima zaglavlja, što će biti od značaja kada kasnije budu obrađivani *prikriveni kanali* za komunikaciju.

### 3.1. Fizički sloj

Uloga fizičkog sloja je da podatke koje dobije u vidu niza bitova prenese duž komunikacijskog kanala [3]. Zaduženja fizičkog sloja sastoje se u tome da se na prijemnoj strani obezbedi pouzdanost razlikovanja bitova 0 i 1 koji se pošalju od strane pošiljaoca; zatim obezbeđivanje odgovarajućeg načina predstavljanja koji će da predstavlja jedinicu; trajanje transmisije bitova; mogućnost ili odsustvo obostranog transfera istovremeno, sinhronizacija itd. Svojstva medijuma kroz koji se prenosi signal ovde imaju veliku ulogu i na njih se obraća velika pažnja u projektovanju - da li je to električni provodnik, vazduh ili nešto drugo.

### 3.2. Sloj veze podataka

Uloga sloja veze podataka (eng. *data link layer*) jeste da nadgleda tok informacija između susednih čvorova u mreži, kao i da za mrežni sloj “pretvori” grubi prenosni uređaj (fizički medijum) u transportnu liniju koja niz bitova prenosi bez greške (kod većine protokola, ne svih).

Pouzdanost prenosa podataka između dve strane sloj veze podataka ostvaruje tako što deli niz bitova na okvire podataka (eng. *data frames*), najčešće od po nekoliko stotina do nekoliko hiljada bitova, zatim izračunava kontrolni zbir podataka (eng. *checksum*) i okvire šalje jedan za drugim. Kada okvir stigne na odredište, kontrolni zbir se izračunava ponovo. Ako se dobijeni zbir razlikuje od zbira sadržanog u okviru, sloj veze zna da je došlo do greške i preduzima korake da je ispravi, a ukoliko nije došlo do greške, šalje pošiljaocu tzv. okvir za potvrdu podataka (eng. *acknowledgement frame*).

Još jedan od problema čijem se rešenju teži u ovom sloju jeste usaglašavanje brzine slanja i brzine prijema podataka, tj. regulacija toka podataka, na šta utiču ne samo svojstva medijuma nego i to koliko mesta na prijemnoj strani za smeštanje podataka postoji.

Kod sloja veze podataka obično razlikujemo dva tipa komunikacije – rasprostranjene (eng. *broadcast*) i mreža od-tačke-do-tačke (eng. *point-to-point*). Prvi tip su obično organizovane kao lokalne mreže (LAN) a drugi kao regionalne mreže (WAN).

Sloj veze podataka se deli na dva podsloja:

1. **Podsloj kontrole pristupa medijuma** (eng. *Media Access Control (MAC)*) – donji podsloj DLL sloja koji se bavi time koji će od trenutnih učesnika u komunikaciji sledeći imati pristup medijumu preko kojeg se ista obavlja. Ovaj podsloj je naročito važan za lokalne mreže u kojima se za komuniciranje najčešće koristi kanal sa slobodnim pristupanjem.
2. **Podsloj kontrole logičke veze** (eng. *Logical Link Control (LLC)*) – gornji podsloj DLL sloja, služi za obezbeđivanje bezbednog transfera i korišćenja usluga MAC podsloja različitim protokolima mrežnog sloja (IP, IPX, Decnet, Appletalk itd.)

### 3.2.1. Ethernet

Ethernet je najšire korišćen LAN protokol razvijen od strane Bob Metcalfe 1973. Godine a koji su zajedno razvili Digital, Intel i Xerox. Standard Ethernet je prvi put zvanično objavljen 1985. godine pod formalnim *IEEE 802.3 – Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*.

Ethernet opisuje način na koji umreženi uređaji formatiraju i prosleđuju podatke drugim uređajima u okviru istog lokalnog umreženog segmenta [7].

U praksi Ethernet mreža je lokalna mreža koja prenosi podatke između uređaja sa ugrađenim adapterom (mrežnom karticom). Radom ovog adaptera upravlja drajver koji se izvršava u računaru. A svaki računar koji učestvuje u komunikaciji preko Ethernet mreže se naziva Ethernet stanica. Signal se kroz Ethernet šalje bit po bit – serijski. Ethernet stanica učestvuje u mrežnom saobraćaju samostalno – nezavisno od ostalih stanica na mreži [4].

Struktura Ethernet okvira je predstavljena na slici 3.



Slika 3: Struktura Ethernet okvira [3]

Svaki Ethernet okvir počinje *Preambulom* dužine 8 bajtova, koja je uvek predstavljena nizom bitova 10101010. Kada se taj niz kodira tehnikom Mančester, dobija se tokom 6.4 μs pravougaoni talas frekvencije 10 MHz, na mreži brzine 100Mb/s, koji primaocu omogućava da se sinhronizuje s pošiljaocem.

Sledeće polje je SOF (start of frame) polje – dužine jednog bajta i predstavlja graničnik za početak okvira.

Odredišna i izvorišna adresa su fizičke adrese izvora i odredišta. Najznačajniji bit odredišne adrese je 0 za obične adrese, a 1 za grupne adrese. Grupna adresa omogućava da više stanica prima poruke preko jedne adrese. Svaka Ethernet mrežna kartica ima fabrički određenu Ethernet (MAC) adresu koja je jedinstvena. Ethernet adapter prima samo pakete koji u polju adrese primaoca imaju njegovu vlastitu adresu ili adresu koja predstavlja *broadcast* ili *multicast* adresu.

Polje *Tip* saopštava primaocu šta da radi sa okvirom. Na istom računaru može istovremeno da se izvršava više protokola mrežnog sloja, pa jezgro OS-a mora da zna kome od njih da prosledi okvir koji je stigao preko Ethernet-a. Poljem *Tip* se specificira proces kome treba predati okvir.

Polje *Podaci* je dužine 1500 bajtova. Ova pomalo proizvoljna granica je izabrana zato što primopredajnik mora imati dovoljno radne memorije da prihvati čitav okvir, a u trenutku uspostavljanja Ethernet standarda RAM memorija je bila prilično skupa [3].

Zatim dolazi poslednje polje – Kontrolni zbir. To je u stvari 32-bitni ključ za heširanje podataka. Ako neki bitovi podataka budu pogrešno primljeni (zbog smetnji na kablo), kontrolni zbir će skoro sigurno biti pogrešan i greška će biti otkrivena. Kontrolni zbir se podvrgava cikličnoj proveru redundanse (CRC), koja ne ispravlja greške, već ih samo otkriva.

MAC adresa predstavlja fizičku adresu interfejsa preko kojeg je računar povezan na Ethernet i zapisuje se kao 6 bajtova najčešće u heksadecimalnoj notaciji. MAC adrese su jedinstvene i proizvođači ih zadaju uređajima pri proizvodnji upisom u ROM memoriju adaptera. Organizacija IEEE upravlja njihovom raspodelom, tako da do sukoba među adresama ne dolazi. Različiti proizvođači različito predstavljaju MAC adrese, a najčešće u obliku: 01-15-9A-74-B3-89-1C ili 01:15:9A:74:B3:89:1C.

### 3.3. Mrežni sloj

Glavni zadatak mrežnog sloja jeste da podacima dobijenim od transportnog sloja pridruži odgovarajuće parametre na osnovu kojih će biti moguće određivanje jednog ili više mrežnih članova kojima pomenute podatke treba isporučiti. Drugim rečima, zadatak mrežnog sloja je da obezbedi jedinstveni sistem adresiranja članova mreže i pravila čijim će poštovanjem biti moguća isporuka podataka na željenu adresu [4]. Pri njegovom projektovanju ključno je odrediti kako se paketi upućuju od izvora ka odredištu. Putanje se mogu zasnivati na statičnim tabelama koje su „ugrađene“ u mrežu i retko se menjaju.

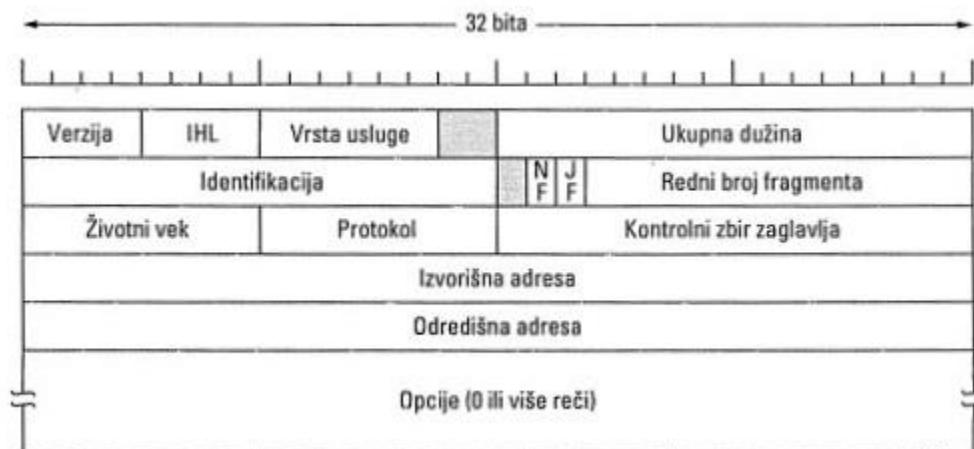
Ukoliko se u podmreži istovremeno nalazi previše paketa, moguće je da dođe do njihovog međusobnog ometanja, čime se stvaraju uska grla. Mrežni sloj treba da bude u stanju da kontroliše i takva zagušenja saobraćaja, tj. da vodi računa o kvalitetu ponuđene usluge (zadržkama, vremenu prolaska, neravnomernosti pristizanja itd.)

Protokoli mrežnog sloja treba da omoguće da se adresiranje neometano vrši i među heterogenim mrežama, zato što načini adresiranja u dve mreže mogu da se razlikuju.

Glavni protokol mrežnog sloja jeste IP protokol i njegove glavne karakteristike jesu univerzalnost i jednostavnost. Njegov kratak pregled biće dat u narednom pasusu, zato što je zaglavlje IPv4 protokola kasnije upotrebljeno za uspostavljanje jednog od *prikrivenih kanala* za komunikaciju u lokalnoj mreži.

### 3.3.1. IP protokol

IP protokol (eng. *Internet Protocol* ) je jedan od protokola mrežnog nivoa. Njegovo zaglavlje, koje je predstavljeno na *slici 4*, se sastoji od fiksnog dela dužine 20 bajtova i opcionog dela promenljive dužine. Ono se prenosi redosledom „*big-endian*“ - sleva udesno, pri čemu prvo ide najznačajniji bit polja *Verzija*.



Slika 4: Zaglavlje IP protokola [3]

U polju *verzija* je označena verzija protokola kome pripada datagram. Kada verziju uključite u datagrame, neće morati cela mreža da odjednom pređe na novu verziju protokola – zamena se može obavljati godinama. Danas je aktuelan prelazak sa IPv4 na IPv6 protokol.

Pošto dužina zaglavlja nije fiksna, postoji u njemu polje *DIZ* (eng. *IHL – Internet Header Length* ) u kome se beleži dužina zaglavlja u 32-bitnim rečima. Najmanja vrednost je 5, što se koristi kada nema opcija. Najveća vrednost ovog 4-bitnog polja je 15.

*Vrsta usluge* je jedno od polja koje je tokom godina promenilo svoju ulogu i koje je u početku bilo namenjeno razgraničavanju različitih klasa usluga, a u tu ulogu ima i sada. U tom pogledu, moguće su različite kombinacije pouzdanosti i brzine isporuke.

Polje *Ukupna dužina* obuhvata ono što se nalazi u datagramu – i zaglavlje i podatke. Maksimalna dužina je 65.536 bajtova. Ta gornja granica je zasad prihvatljiva, ali u budućim gigabitnim mrežama može se pojaviti potreba za većim datagramima.

Pomoću polja *Identifikacija (IPID)* odredišni računar određuje kom datagramu pripada pristigli fragment. Svi fragmenti istog datagrama imaju istu vrednost u polju *Identifikacija*.

Posle ovog polja sledi jedan neiskorišćen bit, a zatim dva jednobitna polja. *NF* znači *Ne Fragmentiraj*. To je naredba ruterima da ne fragmentiraju datagram jer odredište ne može da od njih ponovo sklopi datagram. Npr. kada se računar podiže, njegov ROM može zahtevati da mu se memorijska slika pošalje u obliku jedinstvenog datagrama. Kada datagram označi bitom *NF*, pošiljalac zna da će on stići u jednom komadu, pa makar morao da odstupi od optimalne putanje da bi izbegao mreže koje ograničavaju veličinu paketa [3].



*JF* znači *Još Fragmentata*. Svi fragmenti datagrama, sem poslednjeg, imaju postavljen taj bit. On je potreban da bi se znalo kada je datagram kompletiran.

*Redni broj fragmenta* pokazuje gde spada fragment unutar datagrama. Svi fragmenti datagrama, sem poslednjem, moraju biti umnošci od 8 bajtova – veličine elementarnog fragmenta. Pošto je polje dužine 13 bitova, dozvoljeno je najviše 8192 fragmenta po datagramu, što daje datagram maksimalne dužine 65.536 bajtova, za bajt više no što dozvoljava polje *Ukupna dužina*.

*Životni vek* (eng. *Time to Live - TTL*) je brojač koji ograničava trajanje paketa na mreži. Brojač mora smanjivati vrednost za jedinicu pri svakom skoku, a trebalo bi da je smanjuje i ako se duže zadrži u redu čekanja rutera. U praksi, međutim, on samo broji skokove. Kada njegova vrednost dostigne nulu, paket se odbacuje, a izvorišnom računaru šalje se paket upozorenja. Ovo polje onemogućava datagrame da večno lutaju mrežom, što bi se moglo dogoditi ako se poremete tabele rutera. Takođe, ovo polje obično ima svoju specifičnu vrednost za svaki operativni sistem, tj. svaki kernel operativnog Sistema stavlja vrednost ovog polja na određeni broj za sve odlazeće pakete iz tog računara. A pošto se njegova vrednost smanjuje za jedan pri svakom skoku, ova karakteristika se može iskoristiti kasnije u implementaciji prikrivenih kanala za komunikaciju, što će biti opisano u kasnijem delu rada [3].

Polje *Protokol* pomaže da se odredi proces kome paket treba predati nakon što je mrežni sloj sklopio potpun datagram. Taj proces može biti protocol TCP, ali i protocol UDP ili neki drugi procesi.

*Kontrolnim zbirom zaglavlja* proverava se da li je u slanju zaglavlja došlo do greške, koje su obično posledica neispravnih memorijskih reči rutera. Algoritam za proveru radi tako što se aritmetikom nepotpunih komplemenata sabiraju sve 16-bitne polureči onako kako pristižu, a zatim se od rezultata odbije nepotpuni komplement. Ovo polje je neophodno izračunavati pri svakom skoku, pošto se barem jedno polje uvek menja (*Životni vek*), ali postoje načini da se i to izračunavanje ubrza.

*Izvorišna i Odredišna adresa* ukazuju na broj mreže i broj računara. Svaki računar i svaki ruter na Internetu imaju svoju IP adresu koja obuhvata njihove mreže i broj računara. Ta kombinacija je na globalnom nivou jedinstvena i ne mogu postojati dva računara sa istim IP adresama. Ova adresa je u stvari adresa mrežnog interfejsa računara, pa ukoliko je računar povezan na mrežu preko više interfejsa, imaće više IP adresa.

Polje *Opcije* prihvata informacije koje nose novije verzije protokola (za koje se nije znalo u trenutku projektovanja formata), omogućava eksperimentatorima da isprobaju nove ideje i obezbeđuje mesto za informacije koje se retko koriste. Prvobitno je definisano pet opcija, ali ih je kasnije dodato još [3].

### 3.4. Transportni sloj

Transportni sloj ima osnovni zadatak da prihvata podatke “odozgo”, da ih po potrebi razvrstava u manje grupe i da ih prosleđuje mrežnom sloju, obezbeđujući da svi delovi ispravno stignu na odredište, znači vodeći računa o prenosu, kontroli i ispravljanju grešaka.

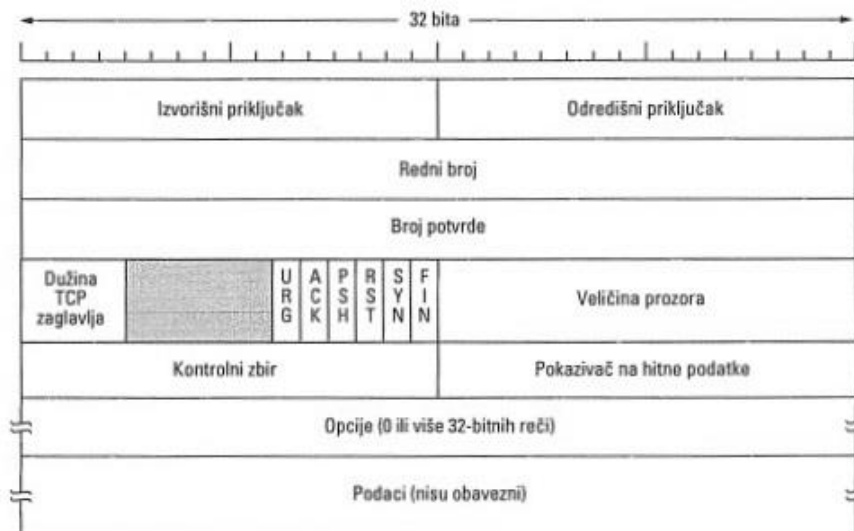
On potpuno povezuje dva kraja: izvor i odredište. Drugim rečima, program na izvornom računaru vodi konverzaciju sa sličnim programom na odredišnom računaru koristeći pri tome zaglavlja poruka i upravljačke poruke.

Najšire rasprostranjeni protokoli transportnog nivoa jesu TCP (Transmission Control Protocol) i UDP (User Datagram Protocol). Njihove karakteristike i primene u današnjim mrežama biće opisani u narednom poglavlju.

#### 3.4.1. TCP (Transmission Control Protocol)

Transmission Control Protocol (protokol za upravljanje prenosom) je protokol zadužen za rad sa podacima u transportnom sloju. TCP je protokol sa uspostavom veze dizajniran da za podatke koristi nizove bajtova i obezbedi pouzdan prenos podataka u oba smera (full-duplex). Ovaj protokol je pogodan za rad sa komunikacionim kanalima visoke pouzdanosti (npr. UTP i optički kablovi) a pokazuje slabije performanse na komunikacionim kanalima sa čestim oštećenjem podataka pri prenosu (npr. bežična komunikacija).

Svaki računar koji podržava protokol TCP ima i transportnu TCP jedinicu: kao proceduru u biblioteci, kao korisnički proces ili kao deo jezgra operativnog sistema. U sva tri slučaja, transportna TCP jedinica radi sa TCP tokovima i ostvaruje interfejs ka IP sloju. TCP jedinica prihvata tokove korisničkih podataka od lokalnih procesa, deli ih u blokove od najviše 64KB i blokove šalje kao zasebne IP datagrame. Kada datagrami sa TCP podacima stignu u računar, predaju je TCP jedinici koja iz njih rekonstruiše originalne tokove bajtova. Zaglavlje TCP protokola je predstavljeno na slici 5.



Slika 5: Zaglavlje TCP protokola [3]

*Izvorišni i Odredišni priključak* (port) identifikuju krajnje lokalne tačke veze. Broj priključka plus IP adresa računara na kome se nalazi obrazuju jedinstvenu 48-bitnu krajnju tačku veze. Izvorišna i odredišna krajnja tačka zajedno definišu određenu vezu, što se naziva soket.

Funkcija polja *Redni broj* je da označava redni broj poslatog segmenta. Broj potvrde označava sledeći očekivani bajt. Oba broja su 32-bitna jer se u TCP toku svaki bajt zasebno numerišu.

*Dužina TCP zaglavlja* označava broj 32-bitnih reči u zaglavlju. Taj podatak je neophodan zato što je polje *Opcije* promenljive dužine, pa je samim tim promenljiva i dužina celog zaglavlja. Ovo polje praktično označava mesto u segment (mereno 32-bitnim rečima) od koga počinju podaci.

Zatim dolazi 6-bitno polje koje se ne koristi. Iza njega sledi šest jednobitnih indikatora. *URG* se postavlja na 1 kada se koristi *Pokazivač na hitne podatke* (eng. *Urgent pointer*). *Urgent pointer* sadrži priraštaj koji treba dodati tekućem rednom broju da bi se dobio redni broj segmenta s hitnim podacima i predstavlja zamenu za slanje zahteva za prekid. Ovim mehanizmom jedna strana može drugoj da pošalje signal, a da se TCP pri tome ne pita šta je razlog prekida.

Kada je *ACK* bit postavljen na 1, to znači da je *Broj potvrde* ispravan. Kada je *ACK* nula, segment ne sadrži potvrdu, pa se *Broj potvrde* zanemaruje.

Bit *PSH* označava da podatke treba odmah proslediti (push). Od primaoca se zahteva da podatke ne čuva u baferu dok se ovaj ne napuni, već da ih prosledi čim ih primi.

Pomoću bita *RST* ponovo se uspostavlja veza oštećena zbog pada računara ili iz nekog drugog razloga. Taj bit se koristi i za odbijanje neispravnih segmenata, odnosno pokušaja da se uspostavi veza. Ukratko, ako dobijete segment s postavljenim bitom *RST*, to znači da negde postoji problem.

Bit *SYN* služi za uspostavljanje veza. Zahtev za uspostavljanje veze ima  $SYN = 1$  i  $ACK = 0$ , što znači da se ne koristi polje za šlepanje potvrde. Odgovor na zahtev, međutim, nosi potvrdu, tako da je u njemu  $SYN = 1$  i  $ACK = 1$ .

Bit *FIN* služi za zatvaranje veze. On označava da pošiljalac nema više podataka za slanje. Međutim, posle zatvaranja (jednog smera) veze, proces koji je raskida može i dalje da prima podatke. Segmenti *SYN* i *FIN* imaju svoje redne brojeve, što garantuje njihovu obradu ispravnim redosledom.

*Veličina prozora* saopštava broj bajtova koje se mogu još poslati uključujući i one već potvrđene. Nulta veličina prozora je sasvim legalna i govori da su primljeni svi bajtovi zaključno s bajtom *Broj potvrde - 1*, ali da primalac trenutno više ne može da prima podatke. Primalac kasnije može da obnovi dozvolu za slanje tako što će poslati segment sa istim *Brojem potvrde* i *Veličinom prozora* različitom od nule. U protokolu TCP, potvrde i dozvole za dodatno slanje potpuno su razdvojene. To znači da primalac može jednostavno da izjavi: Primio sam podatke do k, ali trenutno ih više ne želim. Pomenuto razdvajanje (u stvari, prozor promenljive veličine) daje protokolu dodatnu elastičnost [4].

*Kontrolni zbir* je dodat zbog veće pouzdanosti prenosa. U izračunavanju ovog polja učestvuju TCP zaglavlje, podaci koje TCP nosi i pseudozaglavlje. Pseudozaglavlje se sastoji od 32bitnih IP adresa izvorišnog i odredišnog računara, identifikatora protokola koji odgovara TCP

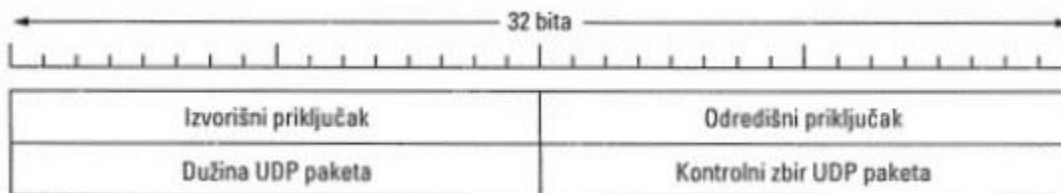
protokolu (6) i broja bajtova TCP segmenta uključujući i zaglavlje. TCP polje *Kontrolni zbir* se pri izračunavanju postavlja na nulu, a polje s podacima se dopunjava bajtom nula ako je njegova dužina neparan broj. Algoritam za izračunavanje kontrolnog zbira jednostavno sabira 16-bitne reči kao nepotpune komplemente i izračunava nepotpun komplement zbira. Zbog toga, kada primalac primeni isti postupak na čitav segment, uključujući i polje *Kontrolni zbir*, rezultat treba da bude 0 [3].

*Opcije* omogućavaju uključivanje dodatnih mogućnosti koje ne predviđa redovno zaglavlje. Najvažnija je ona kojom svaki računar može da zada maksimalan broj bajtova aplikativnog nivoa koji je spreman da primi po jednom TCP segmentu. Efikasnije je koristiti veće segmente jer se povećava količina prenetih podataka po jednom 20-bajtnom zaglavlju. Tokom uspostavljanja veze, svaka strana može da saopšti svoj maksimum i da sazna maksimum druge strane. Ako računar ne koristi ovu opciju, podrazumevana vrednost je 536 bajtova. Od svih računara na Internetu zahteva se da prihvataju TCP segmente dužine  $536 + 20 = 556$  bajtova. Maksimalna veličina segmenta u dva smera iste veze ne mora da bude ista [3].

### 3.4.2. UDP (User Datagram Protocol)

UDP, pored TCP protokola predstavlja jedan od najčešće korišćenih protokola transportnog nivoa na Internetu i lokalnim računarskim mrežama. Od TCP protokola se razlikuje po tome što ne omogućava pouzdan prenos podataka putem ostvarivanja virtuelne veze, kontrolu grešaka, kontrolu redosleda segmenata i ne prilagođava brzinu slanja podataka prijemnoj moći odredišta. Sve ove karakteristike UDP protokol čine jednostavnijim za implementaciju i korišćenje, a samim tim i bržim, ali i protokolom koji ne garantuje pouzdan prenos podataka. U skladu sa prethodno navedenim, glavne primene UDP protokola jesu kod protočnog prenosa glasa i video materijala (Internet telefonija, video konferencije, računarske igre itd.), gde nije bitno ukoliko se neki paket izgubi ili ošteti na putu do odredišta, sve dok ostali paketi pristižu velikom brzinom i u realnom vremenu. Još jedna od prednosti UDP-a nad TCP-em je u tome što UDP ima mogućnost istovremenog slanja podataka svim računarima u lokalnoj mreži – broadcast.

Struktura UDP zaglavlja predstavljena je na *slici 6*.



**Slika 6: UDP zaglavlje [3]**

Po jedan priključak (eng. *Port*) na izvorišnom i odredišnom računaru identifikuju dva kraja “veze”. Izvorišni priključak u zaglavlju uglavnom je potreban kada treba poslati odgovor pošiljaocu. Kopirajući *izvorišni priključak* iz dolaznog segmenta u *odredišni priključak* odgovora, proces koji šalje odgovor može lako da naznači proces koji na drugom kraju treba da ga dobije.

*Dužina UDP paketa* obuhvata 8-bajtno zaglavlje i podatke. *Kontrolni zbir UDP paketa* nije obavezan i prikazuje se sa nulom kada nije izračunavan (kada je kontrolni zbir stvarno nula, to se

prikazuje samim jedinicama). Isključivanje izračunavanja kontrolnog zbira nije preporučljivo, osim ako vam nije važan kvalitet podataka (npr. kod digitalizovanog glasa).

### 3.5. Sloj sesije

Sloj sesije (eng. *Session layer*) omogućava korisnicima na različitim računarima da međusobno uspostave sesiju. Sesije nude različite usluge, uključujući *upravljanje dijalogom* (eng. *Dialog control*), tj. vođenje računa o tome na koga je red da šalje poruke, *rad sa žetonima* (eng. *Token management*), tj. sprečavanje učesnika da istovremeno pokrenu istu kritičnu operaciju i *sinhronizovanje* - proveravanje dugačkog niza podataka tokom prenosa da bi se omogućilo nastavljjanje od tačke prekida u slučaju pada sistema [3].

### 3.6. Sloj prezentacije

Za razliku od nižih slojeva koji samo premeštaju bitove sa jednog mesta na drugo, *sloj prezentacije* se bavi sintaksom i semantikom prenetih informacija. Da bi računari koji podatke predstavljaju na različit način mogli međusobno da komuniciraju, strukture podataka koji se prenose mogu se definisati na apstraktan način i standardno kodirati u cilju prenosa. Sloj prezentacije obrađuje te apstraktne strukture podataka i omogućava da se definišu i razmenjuju strukture podataka višeg nivoa (npr. kodiranje audio i video signala) [3].

### 3.7. Aplikativni sloj

Predstavlja najviši sloj OSI i TCP/IP referentnih modela i nalazi se najbliže korisniku. Elemente na ovom sloju čine korisničke aplikacije koje koriste mrežne resurse i komunikaciju. On sadrži sve protokole višeg nivoa. Na početku su to bili protokoli za virtuelni terminal (TELNET), za prenos datoteka (FTP) i za elektronsku poštu (SMTP), dok je sada razvijen i veći broj protokola kao što su HTTP, DNS itd.

## 4. Sigurna komunikacija na Internetu

U ovom poglavlju biće reči o načinima za sigurnu komunikaciju na Internetu, kako je moguće obezbediti je i koje se tehnike za to koriste. Posebna pažnja biće posvećena prikrivenim kanalima za komunikaciju na Internetu, njihovoj podeli i postojećim implementacijama, što čini i glavnu temu ovog rada.

Kada se u literaturi vezanoj sa bezbednost na Internetu govori o sigurnom komuniciranju dve strane i problemima koji nastaju u pokušaju da se ono ostvari, obično se kao primer uzimaju dva zamišljena lika koja predstavljaju dve strane komunikacije: Alisa i Bob. Alisa i Bob mogu da predstavljaju dva rutera koja žele da razmene tabele za rutiranje, dva hosta koji žele da uspostave sigurnu vezu na transportnom nivou ili dve e-mail aplikacije koje žele da međusobno

komuniciraju. Još jedan lik koji se uvodi u ovom razgovoru obično jeste Trudi (od eng. *intruder*) i koji žele da presretne, pročita ili vrši izmene nad podacima koje razmenjuju Bob i Alisa.

Pod sigurnom komunikacijom na Internetu obično podrazumevamo onu razmenu informacija koja ispunjava sledeće uslove:

1. **Tajnost** – samo primalac i pošiljalac treba da znaju sadržaj prosledene poruke. Pošto prislušivači mogu da presretnu poruku, ovo znači da ona treba da bude kriptovana na način da ne može da bude dekriptovana od strane treće strane, tj. od strane onoga kome nije namenjena. Ovom granom sigurne komunikacije se bavi *kriptografija*.
2. **Autentifikacija** – i prijemna i predajna strana u komunikaciji treba da potvrde svoj identitet jedna drugoj – dokaz da su ono čime se predstavljaju.
3. **Integritet poruke** – čak i nakon što prijemna i predajna strana potvrde svoj identitet, one moraju da budu sigurne da poruke koje razmenjuju nisu u toku prenosa promenjene – namerno ili slučajno. Postoje razne tehnike koje se ovim bave, kao što su kontrolna suma, navedena u prethodnim poglavljima i slično.

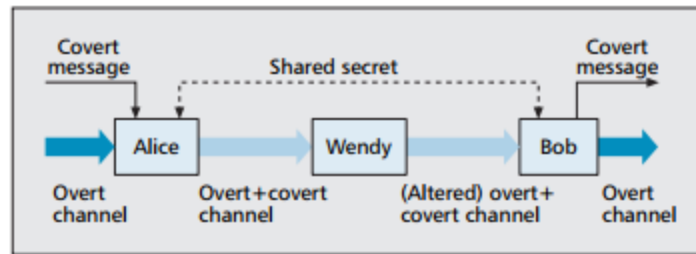
Postoje tri osnovne kategorije sigurne komunikacije na Internetu:

1. *Sakrivanje sadržaja ili prirode komunikacije* - kada se upotrebljava enkripcija (šifriranje), steganografija (sakrivanje fajlova unutar drugih fajlova, poruka) itd.
2. *Sakrivanje strana koje učestvuju u komunikaciji* – kada se sakriva identitet strana koje učestvuju u komunikaciji – pomoću proksija i slično.
3. *Prikrivena komunikacija (Covert communication)* – sakrivanje samog postojanja komunikacije i učesnika u njoj, a što se radi pomoću nekoliko tehnika:
  - a. Kreiranje nasumičnog saobraćaja – šuma, kojim se postojanje prave komunikacije čini težim za otkrivanje i analiza saobraćaja manje pouzdana.
  - b. “*Security by obscurity*” – kada se u samom šumu – nasumičnom saobraćaju nalazi prave tok komunikacije.
  - c. Prikriveni kanal (eng. *Covert channel*) - podrazumeva kanal za komunikaciju nestandardnim putevima, preko procesa kojima to nije prvobitna namena.

## 4.1. Prikriveni kanali za komunikaciju na Internetu

Postoji više definicija *prikrivenih kanala* za komunikaciju, a najpotpunija glasi: *prikriveni kanal* za komunikaciju predstavlja kanal kroz koji se prenosi informacije, a koji nije namenjen niti dizajniran da prenosi informacije uopšte [8]. Može da se kaže i da je *prikriveni kanal* komunikacioni kanal koji je baziran na tome da prenosi informacije u promenljivama koje opisuju stanje resursa. Najčešći model koji opisuje način komunikacije preko *prikrivenih kanala* i pomaže razumevanju samog problema koji pokušavamo rešiti jeste *Problem zatvorenika*, predstavljen je na slici 7 i biće pojašnjen.

Nakon toga, uslediće opisi primene prikrivenih kanala, njihova kategorizacija kao i neke od postojećih implementacija koje se koriste na Internetu.



**Slika 7: Problem zatvorenika** [9]

Dvoje ljudi, Alisa i Bob, nalaze se u odvojenim zatvorskim ćelijama i nameravaju da pobjegnu. Kako bi se dogovorili oko plana za bekstvo, oni moraju da nekako komuniciraju, ali je problem što Vendi – zatvorski nadzornik, nadgleda sve poruke koje oni razmenjuju. Ukoliko nađe bilo kakve znake sumnjive komunikacije, premestiće Boba i Alisu u samicu i oni neće moći da pobjegnu. Alisa i Bob mogu da razmenjuju “obične” poruke pismenim putem, koje mogu da sadrže skrivene informacije. U skladu sa stepenom sigurnosti, postoje tri tipa “nadzornika”:

1. *Pasivni nadzornik* – koji može da špijunira kanal komunikacije ali ne i da promeni sadržaj poruka
2. *Aktivni nadzornik* – koji može da menja sadržaj poruke, ali bez izmena njenog semantičkog značenja
3. *Zlonamerni nadzornik* – koji može da menja sadržaj poruke sa namerom i bez ikakvih posledica

Ovakav scenario može biti prenet na računarske mreže, gde Bob i Alisa predstavljaju dva umrežena računara koja žele da komuniciraju. Oni aktivno razmenjuju informacije preko neškodljivog otvorenog kanala, koji u sebi sadrži i prikriveni (prikriveni, *covert*) kanal. U praktičnim primerima, Alisa i Bob mogu da budu ista osoba, npr. haker koji iznosi zabranjene informacije iz sigurnog okruženja u spoljni svet. Vendi upravlja radom mreže i nadzire saobraćaj kako bi otkrio *prikriveni kanal* ili izmenio pakete koji prolaze da prikriveni kanal poremeti.

U navedenom *problemu zatvorenika* Alisa komunicira sa Bobom, ali generalno komunikacija ne mora da bude jedan-na-jedan, nego može da bude i *multicast* (jedan na više).

#### **4.1.1. Primena prikrivenih kanala**

Širok dijapazon individualaca i organizacija je našlo primenu *covert kanala* u komunikaciji i koordinaciji [8]. Ovo je obično motivisano time da postoji neka treća strana koja nadzire komunikaciju (kao što su vladine agencije u odnosu na terorističke organizacije, hakeri u odnosu na IT sektor neke kompanije ili vlast u odnosu na građane koji žele da slobodno komuniciraju). Sve ove navedene strane imaju u interesu da svoju komunikaciju drže tajnom, ali obična enkripcija nije dovoljna, s obzirom na to da je moguće otkriti ko učestvuje u razgovoru ili zbog samog postojanja kanala komunikacije razumeti strukturu organizacije ili strana koje ga koriste.

Kada hakeri ili špijuni kompromituju neki računar, oni obično iznose zabranjene informacije u zlonamerne svrhe pomoću Trojanskih konja (zlonamernih programa koji su uključeni u legitiman softver) ili alata kojima pokreću *napad uskraćivanja usluge* (DOS – *denial of service*). Ovakve aktivnosti na zaštićenoj mreži unutar neke kompanije, ukoliko se ne sprovode

putem *prikrivenih kanala* mogu da upoznaju administratore mreže koji nadgledaju komunikaciju na mreži, čime bi zaraženi računar bio otkriven.

Ponekad i obični zaposleni u firmama imaju potrebe za korišćenje *prikrivenih kanala*, kako bi zaobišli *zaštitni zid* (eng. *Firewall*) svoje kompanije i dobili pristup Internet resursima. Štaviše, neke države su u poslednje vreme ograničile slobodu govora na Internetu ili čak i zabranile upotrebu enkripcije, pa je jedna od primena *prikrivenih kanala* moguća i u ovoj oblasti.

Mrežni administratori mogu da koriste *skrивene kanale* da osiguraju komunikaciju vezanu za upravljanje mrežom, na taj način skrivajući je od hakera.

Kompjuterski virusi ili crvi mogu da koriste *skrivenе kanale* da se neometano šire kroz mrežu ili razmenjuju informacije potrebne za distribuiranu aktivnost (npr. izvršavanje brute-force napada na kriptosisteme).

*Prikriveni kanali* takođe mogu biti korišćeni da se šalju podaci za autentifikaciju. Brojne tehnike su razvijene koje dozvoljavaju autorizovanim korisnicima da pristupe otvorenim portovima u zaštitnom zidu, a istovremeno ih predstavljajući kao zatvorene za ostale korisnike [9].

#### **4.1.2. Vrste prikrivenih kanala i načini komunikacije**

U ovom radu biće stavljen naglasak uglavnom na *prikrивene kanale* koji se implementiraju u TCP/IP protokol steku, tako što se skrivaju informacije u poljima koja opisuju mrežne protokole (IP, TCP, UDP), a koja nisu namenjena da prenose korisničke podatke. Treba razlikovati *prikrивene kanale* od primene *Steganografije*, koja se odnosi na tehnike skrivanja informacija u poljima koja su namenjena prenosu korisničkih podataka (polje za podatke u TCP/UDP paketima i sl.).

Prikriveni kanali se obično dele na dve vrste:

1. *Covert storage channels* – prikriveni kanali koji koriste smeštanje podataka u neko polje za komunikaciju. Uključuje direktno/indirektno upisivanje vrednosti podataka od strane pošiljaoca i direktno/indirektno čitanje od strane primaoca.
2. *Covert timing channels* – prikriveni kanali koji koriste manipulaciju podacima ili resursima u određenim vremenskim intervalima da bi prosledili 0 ili 1, tako da prijemna strana može iste da nadzire i dekodira.

Postoji nekoliko različitih scenarija komunikacije preko *prikrivenih kanala* u zavisnosti od toga da li su Alisa i Bob na prijemnoj i predajnoj strani kanala ili su *u sredini* (eng. *Middlemen*), tj. manipulišu *otvorenim* kanalom između dva korisnika.

Ukoliko pošiljalac, koji želi da koristi covert kanal, upravlja i onim što se šalje *otvorenim* (eng. *Overt*) kanalom (tj. onim koji *skriva* covert kanal), moguće je maksimizovati kapacitet prikrivenog kanala i iskoristiti sav njegov potencijal. Međutim, nekad u mreži sa velikim ograničenjima, u kojoj je zabranjeno uspostaviti nove kanale za komunikaciju i otvarati nove portove, neophodno je da se pošiljalac ponaša kao posrednik (presretač) i ugradi (eng. *Embed*) *prikriveni* kanal u okviru već postojećeg *otvorenog* kanala. Kada su ovakve okolnosti na snazi, pošiljalac nema apsolutnu kontrolu nad *otvorenim* kanalom i maksimalni kapacitet *prikrivenog* kanala zavisi od kapaciteta *otvorenog* kanala.



Korisnik na prijemnoj strani prikrivenog kanala može da bude i primalac *otvorenog* (*overt*) kanala, mada može da i posrednik/prislušivač, koji će da izvlači skrivene informacije iz otvorenog kanala namenjenog nekom bezazlenom primaocu. Nakon što izvuče potrebne skrivene informacije, idealno bi bilo da primalac prikrivenog kanala ukloni sam kanal, kako bi se sprečilo otkrivanje *prikrivenog kanala* od strane krajnjeg primaoca otvorenog kanala, ili nekog čvora između njih.

Uloga presretača (middleman) ne mora da znači da primalac/pošiljalac *prikrivenog* kanala mora da fizički bude razdvojen od primaoca/pošiljaoca *otvorenog* kanala, već može da bude lociran na ruterima koji prenose komunikaciju otvorenim kanalom ili čak i na istim fizičkim uređajima korisnika *otvorenog kanala*, ali na nižim nivoima u protokol steku.

### **4.1.3. Postojeći prikriveni kanali**

U ovom poglavlju biće date neki postojeći *prikriveni kanali* i tehnike koje se koriste pri njihovom projektovanju, od kojih su neki implementirani u okviru ovog rada. Kanali će biti podeljeni u skladu sa mehanizmima koje koriste, a ne po OSI nivoima na kojima funkcionišu, zbog toga što neki metodi mogu da se koriste u više OSI slojeva.

**Nekorišćeni bitovi u zaglavljjima protokola** – prikriveni kanali mogu da budu ugrađeni u bitove koji se u zaglavljjima paketa ne koriste ili su sačuvani za buduće upotrebe. Ovako ugrađeni prikriveni kanali mogu biti posebno problematični za otkrivanje u slučaju kada se koriste polja koja nemaju neke standardne vrednosti zadate od strane OS-a ili kada primaoci *otvorenih* kanala ne proveravaju njihove vrednosti. Neki istraživači su predložili korišćenje polje *Tip servisa* (eng. *Type of service*) u zaglavlju IP paketa ili *Flags* polje u TCP paketima, koja su opisana ranije u poglavlju 3. Kundur i saradnici su predložili korišćenje polja *Ne fragmentiraj* u zaglavlju IP paketa za slanje jednog bita u jednom trenutku [10]. Hintz je predložio prenos prikrivenih podataka preko polja UP (Urgent Pointer) u TCP zaglavlju, opisano ranije u poglavlju 3.4.1., ukoliko ovo polje u paketima nije prethodno postavljeno [11]. U nekim radovima je predloženo da se koristi RST polje u TCP zaglavlju (poglavlje 3.4.1.), koje se obično postavlja na vrednost 1 u okvir koji se šalje kada želi da se prekine TCP konekcija između dve strane, a u drugim slučajevima nije popunjeno.

**Proširenje zaglavlja protokola i njihova dopuna** – mnogi protokoli podržavaju proširenje standardnog zaglavlja. Obično postoje neki predefinisani *dodaci* za zaglavlja koji pomažu da se prenesu neobavezne informacije na poseban zahtev, ali mnogi protokoli takođe dozvoljavaju da ti *dodaci* nose neke podatke koji nisu bili predviđeni originalnom specifikacijom, čime se proširuju mogućnosti samih protokola. Graf je predložio prenos prikrivenih podataka u zaglavlju *opcije za destinaciju* (eng. *Destination Options*) IPv6 [12]. Ovo zaglavlje nosi opcione informacije za destinaciju paketa, pa ukoliko je tip opcije podešen tako da primalac paketa ignoriše opcije, prikrivene informacije mogu biti direktno enkodirane u ovim poljima.

Prikriveni kanali mogu biti ugrađeni u dopunu samog okvira (eng. *frame*) ili paketa. Npr. Ethernet okviri moraju biti dopunjeni do najmanje veličine od 60 bajtova. Ako standard protokola koji se koristi ne podstiče upotrebu specifičnih vrednosti bajtova za dopunu, bilo koji podaci mogu da se ovde ugrade. Dopune IP i TCP zaglavlja do granice od 4 bajta takođe mogu biti upotrebljene za ugrađivanje prikrivenih kanala.

**Polje IP Identifikacija i Redni broj fragmenta** – kao što u poglavlju 3.3.1. naglašeno, IP ID polje u zaglavlju IP paketa se koristi da bi se sklopili fragmentirani IP paketi. Jedino što je neophodno po IP standardu što se tiče vrednosti ovog polja, jeste da ona jedinstveno identifikuje

IP paket za određeni vremenski period. *Redni broj fragmenta* se koristi da se odredi kojim redosledom fragmenti treba da budu ponovo sklopljeni. U nekim implementacijama prikrivenih kanala koje koriste ovo polje, predloženo je da se svaki bajt podatka koji želimo da pošaljemo pomnoži sa 256 i iskoristi direktno kao IP ID. U drugim se pak koristi slanje prikrivenih informacije kroz najviše bitove IP ID polja, a da se nižih 8 bitova izaberu nasumično. Jedna predložena implementacija koristi ugrađivanje prikrivenih kanala u ova polja u ulozi *presretača* na sledeći način: ako postojeći paket, koji se prenosi kroz mrežu, nije fragmentiran, pošiljalac prikrivenog kanala (presretač – middleman) u našem slučaju ubacuje podatke u polja IP ID i Redni broj fragmenta i postavlja vrednost jednog dogovorenog bita u *flags* polju na 1 (obično *more fragments* flag). Ovaj bit se koristi da se označe paketi koji sadrže prikrivene informacije tako da primalac prikrivenog kanala može da ih razlikuje od običnih paketa.

**Početna vrednost rednog broja paketa u TCP zaglavlju** – uloga rednog broja paketa u TCP zaglavlju opisana je u poglavlju 3.4.1. Početna vrednost ovog broja se zove *Initial Sequence Number (ISN)*. Ova vrednost se bira tako da se naredni brojevi koji će ga slediti u novo-uspostavljenoj TCP vezi neće da se preklape sa rednim brojevima paketa ranije uspostavljenih TCP veza. Neki istraživači su predložili da svaki bajt skrivene informacije koju želimo da prenesemo bude pomnožen sa 256 i direktno ugrađen u TCP ISN. Ovaj kanal je tzv. *bounce* kanal, zato što, umesto slanja TCP SYN paketa direktno primaocu skrivenih podataka, pošiljalac šalje TCP SYN paket na neki drugi host od koga će se paket „odbiti“ – *bounce*, sa postavljenom izvorišnom IP adresom u paketu na vrednost IP adrese hosta kome želimo da prosledimo prikrivene podatke. Nakon što *bounce* host primi ovaj paket, on će poslati ili SYN/ACK ili SYN/RST paket primaocu prikrivenih podataka (jer je njegova IP adresa zapisana kao izvorišna IP adresa paketa), sa brojem potvrde u TCP zaglavlju jednakim  $ISN+1$ . Primalac skrivenih podataka nakon toga dekrementira ACK broj za 1 i dekodira primljeni bajt [13].

**Polje za kontrolni zbir (checksum)** – ovo polje je kod IP paketa moguće koristiti za prikrivenu komunikaciju tako što se izvrši modifikacija njegove vrednosti u skladu sa onim što želimo da prosledimo kroz kanal, istovremeno proširujući IP zaglavlje za onu vrednost bajtova potrebnu da izmenjen kontrolni zbir bude ponovo ispravan za ceo paket. Istu tehniku je moguće koristiti za zaglavlje TCP paketa. Kod UDP paketa polje za kontrolni zbir je opciono, znači moguće je i ostaviti ga praznim, neki istraživači su predložili da se prisustvo ili odsustvo bilo kakve vrednosti u ovom polju tumači kao 1 ili 0 i tako kodiraju informacije kroz prikriveni kanal.

**Izmena polja *Životni vek* u IP paketima** – ovo polje je moguće iskoristiti u mrežama za kodiranje podataka na više načina. Pošto svaki skok kroz ruter smanjuje vrednost ovog polja, najefikasniji način kodiranja jesu dve različite vrednosti TTL koje će predstavljati vrednosti 0 i 1, a koje su dovoljno „daleke“ jedna od druge da ne postoji u stvarnosti dovoljan broj rutera kroz koji bi paket prošao a da se jedna i druga vrednost poklope kada stignu do destinacije. Ova tehnika će u 6. poglavlju biti detaljno opisana, jer je jedna od implementiranih u radu.

**Manipulisanje vremenom pristizanja paketa** – prikrivene kanale je moguće implementirati tako što se manipuliše vremenima pristizanja paketa, odnosno odsustvom ili prisustvom paketa na mrežnom interfejsu u određenom vremenskom trenutku. Ovo odsustvo ili prisustvo se dalje u programu može tumačiti kao 0 ili 1 i tako se svi bitovi iz poruke koju želimo da prosledimo mogu poslati prijemnoj strani, bez mogućnosti da neko presretanjem i dekodiranjem samih paketa i vrednosti u njihovim zaglavljima otkrije da postoji prikriveni kanal. Ovakve kanale je moguće i proširiti tako da nije potrebno sinhronizovati prijemnu i predajnu stranu da u istom

vremenskom intervalu vrše provere, tj. šalju, nego da sam interval između pristizanja dva različita paketa kodira podatke, tj. ukoliko vremenski interval obeležimo sa  $t$  i šaljemo uzastopno dva paketa – ako vreme između pristizanja prvog i drugog bude  $t$ , onda kodiramo 0, a ukoliko bude  $2t$  onda kodiramo 1 i sl.

**Izvorni port u UDP paketima** – ovo polje je moguće koristiti za kodiranje informacija koje želimo da prosledimo na proizvoljan način, uglavnom zato što zaštitni zidovi retko proveravaju izvorni port i ruteri ne menjaju njegovu vrednost, sem u slučaju dinamičkog NAT-a. Jedan način implementacije ovog kanala jeste da se svaki karakter poruke koju želimo da pošaljemo pomnoži sa 255 i dobije se neki veći opseg vrednosti za port, koje nisu sumnjive. Ovakav prikriveni kanal će biti bolje opisan u 6. poglavlju, gde će biti predstavljena i njegova implementacija.

U poglavlju koje sledi biće opisane tehnike programiranja soketa i API-ji koje je potrebno poznavati kako bi prikriveni kanali mogu da budu implementirani.

## 5. Programiranje soketa i *raw* soketi u Linuxu

U ovom poglavlju biće opisano šta u mrežnom programiranju predstavljaju soketi, kako se koriste, šta su *raw* soketi i zbog čega su značajni pri implementaciji prikrivenih kanala za komunikaciju na Internetu. Treba naglasiti da sve programske tehnike i svojstva koja će biti pominjana, biće objašnjena kako funkcionišu i koriste se na operativnom sistemu Linux, zbog specifičnih ograničenja koja na Windows-u važe od verzije Windows 7, a koja sprečavaju kreiranje i manipulaciju *raw* soketima, sem u slučaju korišćenja eksternih biblioteka, što je dovelo do toga da implementacija ovog rada bude prilagođena Linux operativnom sistemu.

*Soketi* dozvoljavaju komunikaciju između dva različita procesa na istom ili različitom računaru. Zapravo, oni su način za komunikaciju za drugim računarima koristeći standardni Unix deskriptor fajla. U *Unix*-olikim operativnim sistemima, svaka ulazno/izlazna operacija se vrši pisanjem ili čitanjem iz fajl deskriptora. Fajl deskriptor je samo celobrojna vrednost povezana sa otvorenim fajlom i može biti ili mrežna konekcija, tekstualni fajl, terminal ili nešto drugo [14]. Za programera, soket ne predstavlja ništa drugo, do jedan vid fajl deskriptora niskog nivoa. Ovo važi zato što komande kao što su *read()* i *write()* rade isto sa soketima, kao što rade sa fajlovima i tokovima.

Postoji nekoliko vrsta soketa dostupnih programeru. Procesi koji koriste sokete obično to rade tako što komuniciraju jedan sa drugim putem iste vrste soketa, ali ne postoji ograničenje koje sprečava da se soketi različite vrste koriste za komunikaciju dva procesa. Njihove vrste su:

1. **Stream soketi** - vrsta soketa koji obezbeđuju garantovanu komunikaciju između dva procesa u smislu da svaki paket poslat sa strane pošiljaoca stiže na adresu primaoca u redu u kom je poslat i celovit. Ovi soketi koriste TCP protokol za komunikaciju, što znači da su konekcioni. Ukoliko nije moguće dostaviti paket na željenu adresu, biće signalizirana greška. Ovi soketi nemaju ograničenja u smislu količine podataka koja može biti poslata preko TCP veze, već u skladu sa veličinom željenih podataka razdeljuju iste na pakete i odvojeno ih šalju, bez ikakve kontrole od strane

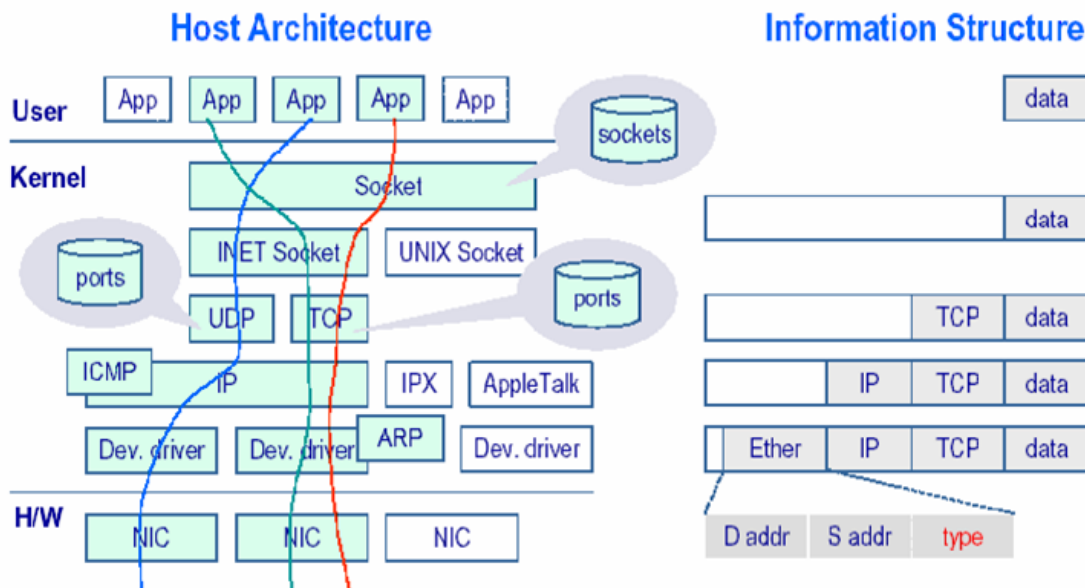
korisnika/programera.

2. **Datagram soketi** – vrsta soketa kod kojih nije zagarantovano da će svi podaci prosleđeni na izlaz biti dostavljeni prijemnoj strani. Oni su beskonekcioni, zato što nije potrebno da bude otvorena stalna konekcija sa drugom stranom, već se svaki put kada podaci treba da budu prosleđeni drugoj strani naprave novi paketi i šalju, bez ikakvih povratnih informacija o njihovom prijemu i redosledu prijema. Koriste UDP protokol.
3. **raw soketi** – tzv. *čisti* soketi, koji dozvoljavaju korisnicima pristup fundamentalnim komunikacionim protokolima (TCP, UDP, IP, Ethernet). Ovi soketi su uglavnom datagramski orijentisani, mada su njihove tačne karakteristike zavisne od toga koji interfejs pruža svaki protokol ponaosob. Nisu namenjeni opštoj primeni i služe uglavnom za razvijanje novih protokola ili pristup zaštićenijim delovima postojećih protokola.

U narednom delu biće više reči o *raw* soketima i njihovom programiranju, ali je prvo potrebno upoznati se kako u stvarnosti – operativnom sistemu, funkcioniše prijem, obrada i put mrežnih paketa koji dospeju na fizički interfejs preko koga je računar povezan na Internet.

## 5.1. Životni vek paketa u mrežnom steku kod klasičnih soketa

Kod klasičnih soketa (stream i datagram) kada mrežni paket stigne na interfejs preko koga je računar povezan na Internet ( ethernet kablom, preko WiFi-a i sl. ), jezgro operativnog sistema preuzima kontrolu nad njim i obrađuje njegova zaglavlja. Put paketa i način funkcionisanja ovog mehanizma dat je na *slici 8*.



Slika 8: Put mrežnog paketa kroz protokol stek [8]

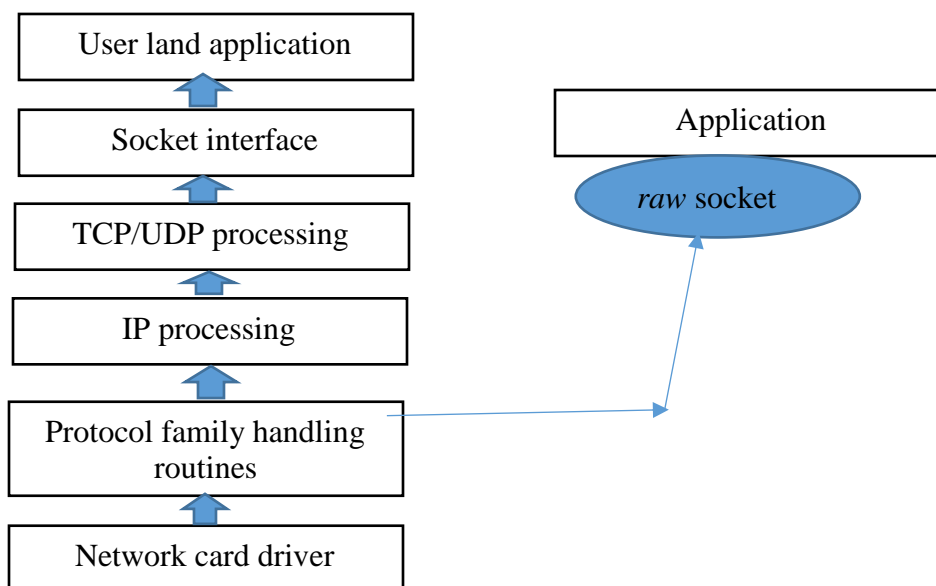
Kada mrežni paket stigne na interfejs, dolazi potpun, sa svim zaglavljima i podacima. Kao što je iznad na slici predstavljeno, *Frame header* predstavlja zaglavlje nekog od protokola data link nivoa – obično Ethernet. Nakon njega sledi IP zaglavlje, pa TCP/UDP zaglavlje i tek onda korisnički podaci. Kako se paket kreće naviše kroz protokol stek, zaglavlja se polako sklanjaju i obrađuju od strane odgovarajućeg mrežnog sloja.

Na taj način neke od korisnih informacija koje mogu da budu sadržane u zaglavljima paketa su potpuno izgubljene za krajnjeg korisnika, tj. aplikaciju. Za sve ovo je zaduženo jezgro (kernel) operativnog sistema i program nema nikakva ovlašćenja nad obradom zaglavlja paketa. Kada se koriste klasični *stream* i *datagram* soketi do korisnika mogu da stignu informacije i paketi koji su namenjeni samo njemu (*unicast*), namenjeni svim računarima u mreži (*broadcast*) ili namenjeni nekom broju računara u mreži (*multicast*). Ovo znači da nije moguće prihvatati pakete namenjene ostalim računarima – oslušivati mrežu. Još jedan od problema koji se javlja kada se soketi koriste na ovaj način jeste da pre nego što se paket pošalje sa našeg interfejsa, odnosno aplikacije, nije moguće modifikovati zaglavlja paketa. Ovo se dešava zato što pri slanju podataka kod klasičnih soketa, moguće je popuniti i modifikovati samo polje za podatke (data), a sva ostala polja i zaglavlja popunjava jezgro operativnog sistema, u obrnutom redosledu od onog opisanog pri prijemu.

Kako je ova ograničenja moguće zaobići i manipulirati naprednim funkcionalnostima soketa, biće dato u nastavku.

## 5.2. Raw soketi

Pomoću *raw* soketa moguće je pristupiti svim zaglavljima paketa koji stignu na mrežni interfejs, a ne samo polju za podatke. Na *slici 9* je predstavljen redosled obrade paketa koji stignu na mrežni interfejs u slučaju kada se koriste *raw* soketi.



Slika 9: Način rada *raw* soketa

Kao što je moguće zaključiti sa gore prikazane slike, *raw* soketi omogućuju zaobilazanje celog protokol steka u jezgri operativnog sistema i dostavljanje svih podataka iz paketa korisničkoj aplikaciji. To znači da se soket zapravo kreira direktnom vezom sa fizičkim interfejsom kojim je povezan računar na mreži. Zaobilazanje viših slojeva mrežnog steka omogućava da korisničkoj aplikaciji budu dostavljeni i paketi koji nisu namenjeni računaru na kojem je ona pokrenuta, zato što jezgro OS-a više ne sprečava da se paketi sa različitim MAC i IP adresom u odredištu dostave višim slojevima, ako su ovakvi paketi došli do interfejsa. Ovo je moguće ostvariti tek nakon što se mrežna kartica prebaci u tzv. slobodni mod (eng. *promiscuous mode*), što joj omogućava da prima sve pakete na mreži, čak i one koje njoj nisu namenjene. Ovakav mod mrežnih kartica se koristi kod alata za nadzor mreže kao što su *Wireshark*, *Ethereal* i sl. U Linuxu, postavljanje mrežne kartice u slobodni mod je moguće postići setovanjem `IFF_PROMISC` flega ili koristeći `ifconfig` komandu (`ifconfig eth0 promisc`). Treba napomenuti da neke mrežne kartice ne podržavaju ovaj mod rada. Treba naglasiti da je prilikom kreiranja *raw* soketa moguće i da se određeni nivo kontrole operativnog sistema nad paketima zadrži, tako da jezgro OS-a i dalje obrađuje pakete do nekog nivoa, u zavisnosti od toga kakve parametre navedemo pri kreiranju soketa. Ovo praktično znači da je moguće da korisnik sam izabere da li će primiti pakete sa data link nivoa ili recimo mrežnog nivoa, što dalje određuje kakva će zaglavlja ti paketi sadržati i kako je neophodno parsirati ih u odgovarajuće strukture da bi se dobile korisne informacije, a o čemu će biti reči malo kasnije.

Osim što je pomoću *raw* soketa moguće primiti cele pakete sa svim zaglavlјima Ethernet, IP i TCP/UDP protokola, pomoću njih je moguće i kreirati i na mrežu poslati delimično proizvoljne pakete, sa željenim vrednostima u odgovarajućim zaglavlјima. Valja naglasiti da su paketi „delimično“ proizvoljni u smislu da zaštitni zid na operativnom sistemu i ruteri neće dozvoliti da na mrežu prođe paket koji ima vrednosti u polјima u zaglavlјu koje nisu usklađene sa datim protokolom. Ova svojstva *raw* soketa ih čine idealnim za primene u mrežnim aplikacijama kojima je potreban dublji pristup zaglavlјima paketa, njihovoj analizi ili projektovanju novih protokola.

Pre nego što pređemo na to kako se *raw* soketi na Linuxu kreiraju u programskom jeziku C, na slici 10 su dati hederi koje je potrebno uključiti u C program da bi moglo da se manipuliše soketima, kao i zaglavlјima paketa koji stižu na mrežni interfejs ili koje želimo da pošaljemo.

```
#include<netinet/in.h>
#include<errno.h>
#include<netdb.h>
#include<stdio.h> //Za standardni ulaz i izlaz
#include<stdlib.h> //malloc funkcija
#include<string.h> //strlen, memset funkcije
#include<netinet/ip_icmp.h> //Deklaracije za ICMP zaglavlje
#include<netinet/udp.h> //deklaracije za UDP zaglavlje
#include<netinet/tcp.h> //deklaracije za TCP zaglavlje
#include<netinet/ip.h> //deklaracije za IP zaglavlje
#include<netinet/if_ether.h> //za parametar ETH_P_ALL
#include<net/ethernet.h> //za Ethernet zaglavlje
#include<sys/socket.h>
#include<arpa/inet.h>
```

Slika 10: Hederi koje je potrebno uključiti za rad sa *raw* soketima

Hederi *stdio.h* , *stdlib.h*, *string.h* i *errno.h* su standardni u programskom jeziku C, a u našem slučaju služe za sledeće operacije respektivno: standardni ulaz i izlaz, dinamičko alociranje memorije sa *malloc*, upotrebu funkcija *strlen* i *memset* i na kraju zabeleške o greškama. Hederi *ip\_icmp.h*, *udp.h*, *tcp.h*, *ip.h* i *ethernet.h* služe za definiciju odgovarajućih struktura koje predstavljaju zaglavlje protokola ICMP, UDP, TCP, IP i Ethernet. Heder fajl *socket.h* sadrži sve funkcije potrebne za rad sa soketima, a *arpa/inet.h* strukture za adresni format kod soketa.

Sledi isečak koda na slici 10 koji pokazuje kako se u programskom jeziku C kreira *raw* soket i povezuje na mrežni interfejs *eth0*.

```
// int socket(int domain, int type, int protocol);
int sock_raw = socket( AF_PACKET , SOCK_RAW , htons(ETH_P_ALL)) ;

// int setsockopt(int s, int level, int optname, const void *optval, socklen_t optlen);
setsockopt(sock_raw , SOL_SOCKET , SO_BINDTODEVICE , "eth0" , strlen("eth0")+ 1 );
```

**Slika 11: Kreiranje *raw* soketa i njegovo povezivanje na mrežni interfejs**

U prvoj liniji koda se poziva funkcija *socket()* kojom se kreira novi soket koji će funkcionisati na data link nivou, odnosno sva zaglavlja uključujući Ethernet, IP i TCP/UDP biće dostupni kada se primi paket na njemu, što specificira parametar *ETH\_P\_ALL*. Osim date vrednosti, ovaj parametar može da ima i neke od sledećih vrednosti: *IPPROTO\_RAW*, *IPPROTO\_IP*, *IPPROTO\_TCP*, *IPPROTO\_UDP* itd. Parametar *AF\_PACKET* označava da su paketi koji će biti primljeni niskog nivoa, tj. sve vrste Ethernet paketa na interfejsu. Povratna vrednost funkcije *socket()* je celi broj koji predstavlja fajl deskriptor asociiran sa datim soketom.

Druga linija koda se sastoji od poziva funkcije *setsockopt()* koja služi da se kreirani soket *sock\_raw* poveže sa mrežnim interfejsom preko kojeg treba da šalje i prima pakete. Parametar *SOL\_SOCKET* je podrazumevani parametar za manipulisanje operacijama na soketima, a *SO\_BINDTODEVICE* je parametar koji se odnosi na samu operaciju koju želimo da izvršimo, u ovom slučaju povežemo soket sa fizičkim interfejsom. Četvrti parametar, „*eth0*“ je naziv interfejsa sa kojim želimo da povežemo soket, onako kako ga vidi operativni sistem. Poslednji parametar (*optlen* u potpisu funkcije) predstavlja dužinu četvrtog parametra *optval* koji je u našem slučaju *eth0*.

Nakon što se soket poveže sa interfejsom i njegovo podešavanje završi, moguć je odabir šta će se sa njime dalje raditi – da li će biti kreirani paketi sa odgovarajućim zaglavljima za izabrani protokol i poslati na izlaz soketa, ili će pak soket biti u osluškujućem modu i primati pakete koji dolaze na mrežni interfejs za koji je vezan. Isečak koda kojim se poziva funkcija za primanje podataka sa interfejsa, sa odgovarajućim njenim potpisom, je data na slici 12.



```
// ssize_t recvfrom(int sockfd, void *buf, size_t len, int flags,  
//                  struct sockaddr *src_addr, socklen_t *addrlen);  
data_size = recvfrom(sock_raw, buffer, 65536, 0, &saddr, &saddr_size);
```

**Slika 12: Pozivanje funkcije za prijem podataka na *raw* soketu**

Funkcija koja je data na isečku iznad poseduje sledeće parametre:

- *sockfd* – fajl deskriptor soketa sa kog želimo da izvršimo prijem paketa, u našem slučaju deskriptor soketa kreiranog ranije u kodu prikazanom na slici 10
- *buf* – bafer u koji će biti primljen paket koji stiže na otvoreni soket, a koji mora da bude odgovarajuće veličine; u našem slučaju referencira se na promenljivu *buffer* ranije kreiranu u kodu
- *len* – veličina bafera prethodno specificiranog. Potrebno je da veličina bude jednaka maksimalnoj veličini jednog Ethernet okvira, zato što će na soket stizati Ethernet paketi.
- *flags* – posebni flegovi koji omogućavaju neke specifične operacije sa soketima, kao što su neblokirajući prijem itd. U našem slučaju nijedan fleg nije postavljen, tako da je poziv ove funkcije standardni – blokirajući.
- *src\_addr* – ukoliko je ovaj parametar postavljen na neku vrednost različitu od *NULL*, i protokol ispod protokola na kom funkcioniše soket pruža informacije o izvornoj adresi paketa, onda se ova struktura popunjava. U našem slučaju, parametar je popunjen pokazivačem na ranije kreiranu promenljivu *saddr* strukturnog tipa *sockaddr*.
- *addrlen* - pokazivač na strukturu tipa *socklen\_t* koja sadrži informacije o veličini adrese *src\_addr*. Ovu strukturu je potrebno inicijalizovati pre poziva funkcije, i u njoj će se nakon poziva funkcije nalaziti veličina adrese, tj. strukture *src\_addr*.

Nakon što je paket primljen na mrežni interfejs, potrebno je ekstrahovati iz njega odgovarajuća zaglavlja Ethernet, IP i TCP/UDP protokola, u zavisnosti od toga za koji nivo u protokol steku je vezan *raw* soket. Ovo se radi tako što se odgovarajući bafer u koji je smešten primljeni paket, koji se kod *raw* soketa zove *datagram* parsira u odgovarajuće strukture podataka koje predstavljaju zaglavlja protokola koji želimo da ispitamo. Primer koji ilustruje ovaj princip parsiranja paketa iz prijemnog bafera u IP zaglavlje, radi određivanja izvorišne i odredišne IP adrese je dat na *slici 13*.



```

//Get the IP Header part of this packet
struct iphdr *iph = (struct iphdr *) (buffer + sizeof(struct ethhdr) );
unsigned short iphdrlen;

iphdrlen =iph->ihl*4;

memset(&source, 0, sizeof(source));
source.sin_addr.s_addr = iph->saddr;

memset(&dest, 0, sizeof(dest));
dest.sin_addr.s_addr = iph->daddr;

```

**Slika 14: Izdvajanje IP zaglavlja iz paketa primljenog sa raw soketa**

U prvoj liniji koda dat je način parsiranja podataka iz bafera u *iphdr* strukturu koja je definisana u fajlu *ip.h* koji je uključen na početku programa komandom `#include <net/ip.h>`. Može se videti kako se obraća adresi u baferu koja se nalazi nakon Ethernet zaglavlja, koje enkapsulira ostale protokole unutar sebe, kao što je ranije u radu, na *slici 8* prikazano. Dalje u kodu može se videti kako se funkcijom *memset()* čisti bafer *source* koji predstavlja promenljivu tipa strukture *sockaddr\_in*, pristupa se njenom polju *sin\_addr* koje je tipa *in\_addr* pa se zatim vrednosti *s\_addr* ovog polja dodeljuje vrednost izvorne IP adrese, a kasnije i odredišne IP adrese pročitane iz zaglavlja. Opis navedene strukture *sockaddr\_in* je dat na *slici 14*, a *in\_addr* na *slici 15*.

```

struct sockaddr_in {
    sa_family_t    sin_family; /* address family: AF_INET */
    in_port_t      sin_port;   /* port in network byte order */
    struct in_addr sin_addr;    /* internet address */
};

```

**Slika 13: Format adrese koji se koristi kod soketa**

Format adrese koji se koristi kod soketa predstavlja kombinaciju IP adrese interfejsa i 16-bitnog broja porta. Klasičan IP protokol ne obezbeđuje brojeve portova, već su oni, kao što je ranije navedeno, implementirani u protokolima viših nivoa kao što su TCP i UDP. Međutim, kod soketa se oni koriste. U slučaju *raw* soketa, za koje je rečeno da „zaobilaze“ protokole nivoa višeg od IP protokola, polje *sin\_port* nije popunjeno. Polje *sin\_family* je uvek postavljeno na *AF\_INET*, i ova vrednost označava da se na soketu očekuju adrese sa Interneta – IP adrese.

```

/* Internet address. */
struct in_addr {
    uint32_t      s_addr;      /* address in network byte order */
};

```

**Slika 15: Struktura IP adrese koja se koristi kod soketa**

Ovu strukturu nije potrebno posebno objašnjavati, sem što je očigledno da sadrži polje *s\_addr* tipa *uint32\_t* i da se ono koristi za upis informacija o IP adresi hosta.

## 6. Implementacija prikrivenih kanala

U ovom poglavlju biće opisana aplikacija koja je projektovana za komunikaciju prikrivenim kanalima na mreži, kao i tehnike i alati korišćeni u tom procesu. Aplikacija je razvijena u cilju predstavljanja mogućih načina komunikacije između dva računara na lokalnoj mreži, tako da proces komunikacije može teško da bude otkriven i ometen od strane trećeg lica. Aplikacija svoju funkcionalnost ostvaruje manipulisanjem vrednosti u zaglavljinama paketa, tj. konkretno pomoću dva kanala prenosa: *TTL* polja u IP zaglavlju i *source port* polja u UDP zaglavlju. *raw* soketi, čije je poznavanje bilo neophodno za projektovanje ove aplikacije, opisani su u prethodnom poglavlju.

Aplikacija se sastoji od dva programa – primaoca i pošiljaoca prikrivenog kanala, koji se pokreću na dva različita računara u lokalnoj mreži.

### 6.1. Arhitektura aplikacije

Arhitektura aplikacije biće prikazana kao serija pogleda na aplikaciju: pogled na slučajeve korišćenja, pogled na logičku arhitekturu aplikacije, pogled na razmeštaj komponenti aplikacije i pogled na implementaciju. Neki od ovih pogleda biće predstavljeni odgovarajućim UML dijagramom.

Izabrana arhitektura aplikacije ima ograničenje u pogledu broja učesnika u komunikaciji – putem prikrivenog kanala je moguće da komuniciraju samo dva korisnika istovremeno, što znači da jedan *primaoc* može biti povezan samo sa jednim *pošiljaocem* u jednom vremenskom trenutku, a isto važi i obrnuto.

Ključni zahtevi i ograničenja koja imaju značajan uticaj na izbor arhitekture i projektovanja sistema su:

1. *covert\_receiver* i *covert\_sender* su implementirani kao desktop konzolna aplikacija u programskom jeziku C, namenjene za Linux operativni sistem
2. Aplikacija mora pružiti zadovoljavajuće performanse pri bilo kojim uslovima na mreži

Aplikacija je primarno projektovana za razmenu poruka između dve strane putem prikrivenih kanala, međutim moguće je modifikovati je da se preko nje šalju i fajlovi kodirani na odgovarajući način.

### 6.2. Pogled na slučajeve korišćenja

Slučajevi korišćenja su:

- Odabir moda rada
- Kreiranje i popuna paketa za slanje na mrežu
- Slanje prikrivenim kanalom
- Prijem prikrivenim kanalom

Svaki od slučajeva korišćenja može da inicira korisnik aplikacije.

### 6.3. Pogled na logičku arhitekturu sistema

Ovaj pogled sadrži opis slojeva i paketa u okviru sistema, kao i njihovu međusobnu organizaciju.

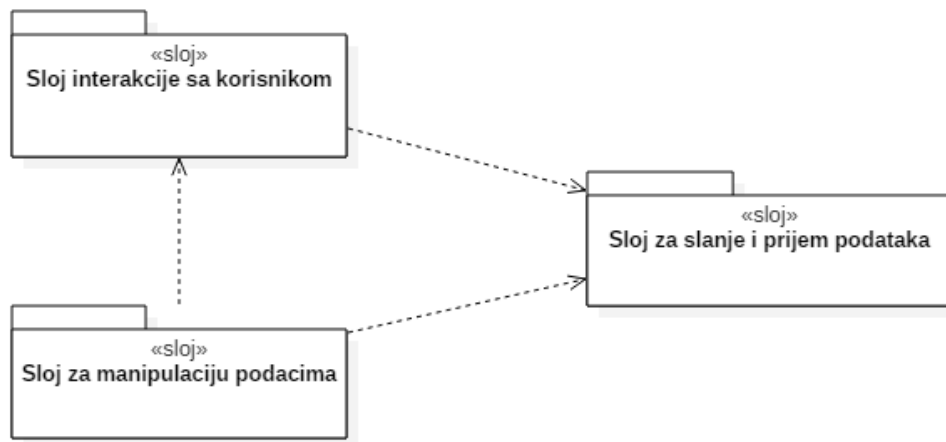
Logički pogled na aplikaciju obuhvata tri glavna sloja: *sloj interakcije sa korisnikom*, *sloj manipulacije podacima* i *sloj za slanje i prijem prikrivenim kanalom*.

*Sloj interakcije sa korisnikom* obuhvata funkcije koje se tiču odabira parametara i poruke za prenos prikrivenim kanalom: odabir moda rada, unos izvorišne i odredišne IP adrese preko parametara konzolne aplikacije, kao i same poruke za slanje.

*Sloj manipulacije podacima* obuhvata odgovarajuću obradu podataka radi prilagođavanja slanju na strani pošiljaoca i prikazu na strani primaoca.

*Sloj za slanje i prijem prikrivenim kanalom* obuhvata funkcionalnosti vezane za kreiranje raw soketa i njegovo prilagođavanje slanju i prijemu podataka.

Pogled na logičku arhitekturu opisanog sistema dat je na *slici 16*.



Slika 16: Logička arhitektura sistema

## 6.4. Pogled na implementaciju sistema

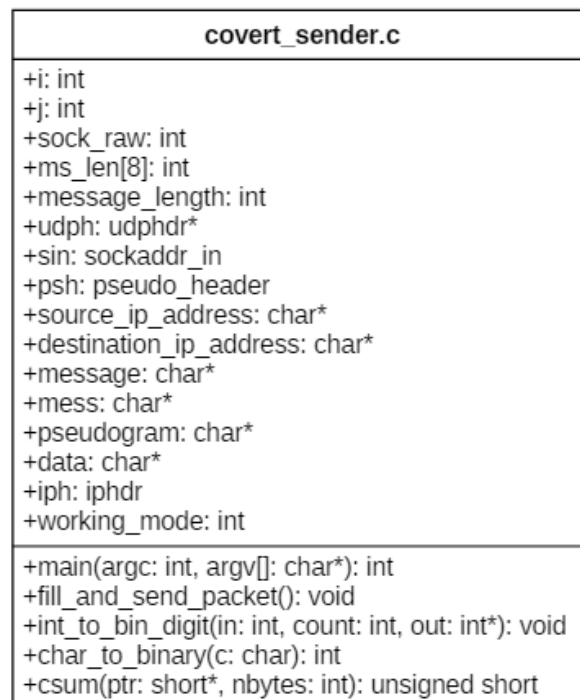
Pogled na implementaciju sistema sadrži dva elementa ključna u projektovanju i načinu funkcionisanja aplikacije: *pošiljaoca* i *primaoca*, odnosno *covert\_sender* i *covert\_receiver* koji su u praksi realizovani kao dva zasebna programa koja čine ceo sistem. Njihov prikaz i međusobni odnos dat je na *slici 17*.



Slika 17: Komponente koje sačinjavaju sistem

### 6.4.1. covert\_sender

Komponenta, odnosno program *covert\_sender* je elemenat implementirane aplikacije za komunikaciju prikrivenim kanalima koji služi za prihvatanje, pripremu, kodiranje i slanje poruke putem implementiranog kanala, izabranim načinom rada. Unutar samog programa definisano je više funkcija koje služe kao pomoćne funkcije u pripremi podataka, kao i onih koje služe za pripremu samih mrežnih paketa i slanje istih preko kanala. Dijagram na kojoj su predstavljene promenljive i funkcije koje sačinjavaju ovaj program, biće dat na *slici 18*.



Slika 18: Klasni dijagram covert\_sender komponente

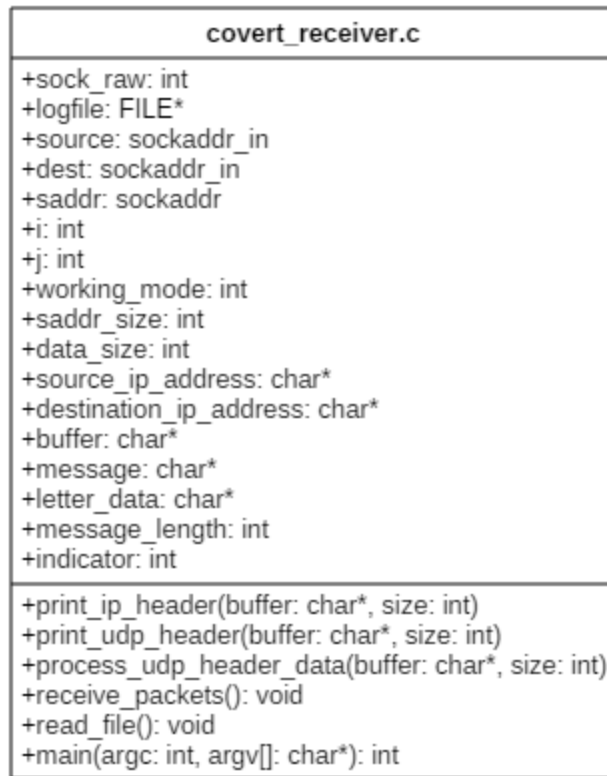
Sada će samo ukratko biti opisano dejstvo funkcija navedenih u klasnom dijagramu, dok će kasnije u slučajevima korišćenja njihov rad biti detaljnije objašnjen.

- *void main(int argc, char\* argv[])* – glavna funkcija u programu, kojoj se pozivanjem programa prosleđuju i njegovi argumenti. U njoj se vrši inicijalizacija soketa i pozivaju ostale funkcije.
- *void fill\_and\_send\_packet(int ttl, int source\_port, int ttl\_or\_sp)* – funkcija kojom se popunjavaju vrednosti svih polja u paketu koji se prosleđuje na odgovarajući način, u zavisnosti od prosleđenih parametara, odnosno moda rada programa.
- *void int\_to\_bin\_digit(unsigned int in, int count, int\* out)* – pomoćna funkcija, koja služi za pretvaranje celog broja u njegovu binarnu reprezentaciju.
- *int\* char\_to\_binary(char c)* – pomoćna funkcija koja služi za pretvaranje karaktera u njegovu binarnu reprezentaciju u skladu sa vrednošću u ASCII tabeli.
- *unsigned short csum (unsigned short \*ptr, int nbytes)* – pomoćna funkcija za izračunavanje kontrolne sume paketa.

Promenljive navedene u klasnom dijagramu su uglavnom postavljene kao globalne promenljive programa, kojima je moguće pristupiti iz svih funkcija. Neke od najznačajnijih za sam rad programa su: *ihhdr* i *udphdr* koji predstavljaju zaglavlja IP i UDP protokola onako kako su opisani u fajlovima koje treba uključiti u program – opisanih u poglavlju 5.2; zatim *working\_mode* kojim se bira između moda rada – IP *TTL* kanala ili UDP *source\_port* kanala za prikrivenu komunikaciju; *source\_ip\_address* i *destination\_ip\_address* koje predstavljaju adrese pošiljaoca i primaoca u prikrivenoj komunikaciji, a prosleđuju se putem argumenata, prilikom poziva programa. Takođe, tu su i bitni *pseudo\_header* i *pseudogram*, koji služe za izračunavanje UDP kontrolne sume. *Pseudo\_header* sadrži, sem informacija o UDP portu i informacije o IP adresi izvora i odredišta UDP paketa, jer se u samom UDP zaglavlju ove informacije ne nalaze, zato što su deo IP protokola, koji je na nivou ispod UDP-a.

#### 6.4.2. *covert\_receiver*

Komponenta, odnosno program *covert\_receiver* je elemenat implementirane aplikacije za komunikaciju prikrivenim kanalima koji služi za osluškivanje saobraćaja na mreži, prijem odgovarajućih podataka namenjenih odredišnom računaru u zavisnosti od odabranog moda rada, dekodiranje istih i prikaz korisniku onog što je primljeno. Unutar samog programa definisano je više funkcija koje služe kao pomoćne funkcije za upis primljenih podataka u privremeni fajl, zatim čitanje iz istog i prikaz korisniku na displeju, da se ne bi podaci smeštali u dinamičke bafere koji mogu da zauzimaju veliku memoriju – zbog prethodno nepoznate dužine podataka. Takođe, tu su i pomoćne funkcije koje služe da, ukoliko korisnik to želi, ispisuju sadržaj *Ethernet*, *IP* i *UDP* zaglavlja u konzoli, radi kontrole onoga što pristizuje na mrežni intefejis i proveru u slučaju postavljanja loših parametara za komunikaciju. Dijagram na kojoj su predstavljene promenljive i funkcije koje sačinjavaju ovaj program, dat je na slici 19.



**Slika 19: Klasni dijagram komponente covert\_receiver**

Kratak opis funkcija navedenih u klasnom dijagramu će biti dat u nastavku, a njihova direktna uloga tokom rada programa i detaljnije objašnjenje sa isečcima koda, biće dati u slučajevima korišćenja.

- *void print\_ip\_header (char\* buffer, int size)* – služi za štampanje zaglavlja IP paketa
- *void print\_udp\_header (char\* buffer, int size)* – služi za štampanje zaglavlja UDP paketa
- *void process\_udp\_header\_data (char\* buffer, int size)* – obrađuje zaglavlje UDP paketa u slučaju kada se koristi mod rada za slanje paketa prikrivenim kanalom preko *UDP source\_port* polja
- *void receive\_packets()* – funkcija koja se poziva kada započne prijem paketa sa mrežnog interfejsa, analizira pristigle pakete i poziva ostale funkcije koje treba da obrade te pakete i ekstrahuju korisne podatke
- *void read\_file()* – učitava podatke prethodno upisane u fajl i prikazuje ih korisniku
- *int main (int argc, char\* argv[])* – glavna funkcija u programu, koja učitava argumente pristigle pri pozivu, inicijalizuje raw soket i poziva ostale funkcije.

Promenljive koje se koriste u programu su uglavnom slične onima upotrebljenim u *covert\_sender* programu i opisanim u prethodnom delu, sa dodatkom nekih kao što su *message\_length*, *logfile* i *indicator* koje se koriste u funkcijama specifičnim primaocu.

## 6.5. Pogled na osnovne funkcionalnosti

U ovom poglavlju biće dati osnovne funkcionalnosti koje karakterišu projektovanu aplikaciju.

### 6.5.1. Odabir moda rada

Odabir moda rada aplikacije predstavlja odabir između dva prikrivena kanala implementirana u sklopu aplikacije: *TTL* polja u IP zaglavlju i *source\_port* polja u UDP zaglavlju. Na *slici 20* biće dat isečak koda koji se koristi kod odabira moda za rad aplikacije u *covert\_sender* komponenti, uz napomenu da se na isti način bira i mod rada kod *covert\_receiver*-a.

```
working_mode = 0;
while(working_mode!=1 && working_mode!=2)
{
    printf("Choose the covert channel working mode.\n
           \tFor sending message via TTL field in IP type 1;
           \n \tFor sending message via Source port field in UDP type 2:\n");
    scanf("%d", &working_mode);
}
```

Slika 20:Odabir moda rada aplikacije

Ova dva kanala su odabrana pri implementaciji kanala zbog toga što su pre svega nekonvencionalni za prikrivene kanale, odnosno nisu jedni od onih koji se prvi proveravaju u slučaju da nadzornik mrežne komunikacije želi da otkrije komunikaciju preko prikrivenih kanala na mreži.

*TTL* polje u IP zaglavlju se modifikuje tokom putovanja paketa kroz mrežu od strane rutera, kao što je ranije objašnjeno, međutim, način na koji je kanal implementiran, odnosno svojstvo da se *TTL* polje koristi samo kako bi se kodirale dve različite vrednosti - 0 i 1, postavljajući početnu vrednost *TTL* u paketima na 255 i 64 za ta dva bita respektivno, dozvoljava ruterima i da menjaju to polje pri putovanju kroz mrežu, ali da se nikad vrednosti njihove ne susretnu i pomešaju međusobno, tako da strana primaoca uvek može da razlikuje šta je pošiljalac hteo da pošalje. Takođe, još jedno svojstvo *TTL* polja i razlog odabira ovih vrednosti, a ne čistog kodiranja karaktera u *TTL* polje jeste i to što određeni operativni sistemi pokazuju određen obrazac ponašanja što se tiče *TTL* polja u paketu, tako da u zavisnosti od sistema postavljaju njegovu vrednost na 64, 128 ili 255 obično, a ne druge nasumične vrednosti, pa bi te vrednosti bile sumnjive potencijalnom nadzorniku mreže.

*Source\_port* polje u UDP paketima je za prikriveni kanal odabrano, kao što je u poglavlju 4.1.3 naglašeno – zato što većina zaštitnih zidova ne obraća pažnju i ne filtrira ovo polje pošto pošiljalac ga koristi da identifikuje svoj slobodan port, za koji posrednik koji bi presreo saobraćaj ne može da zna da li je stvarno slobodan ili se samo koristi u prikrivenoj komunikaciji. Kod ovog kanala, iskorišćen je način implementacije koji podrazumeva množenje karaktera poruke koju želimo da prosledimo sa 255, kako bi se dobili veliki brojevi portova koji više liče na stvarne portove slobodne za komunikaciju.

### 6.5.2. Kreiranje i popuna paketa za slanje na mrežu

Nakon što se kreira raw soket i pripremi interfejs za slanje, potrebno je da se kreira datagram koji će predstavljati paket, odnosno popuni jedan bafer sa zaglavljima prokola koji će se koristiti za slanje prikrivenim kanalom. U slučaju pošiljaoca, kreiran je raw soket sa trećim parametrom *IPPROTO\_RAW* koji kaže da je raw soket u stvari pozicioniran na mrežnom nivou i da je kada se kreiraju paketi i prosleđuju preko ovog soketa na mrežu, potrebno popuniti sva zaglavlja mrežnog i transportnog nivoa, a zaglavlje Ethernet okvira biće popunjeno od strane jezgra operativnog sistema. Na *slici 21.* biće dat isečak iz koda koji se bavi popunom paketa koji treba biti prosleđen na mrežnu, a nalazi se u funkciji *fill\_and\_send\_packet()* opisanoj u prethodnom poglavlju.

```
memset (datagram, 0, 4096);
iph = (struct iphdr *) datagram;
udph = (struct udphdr *) (datagram + sizeof (struct ip));
data = datagram + sizeof(struct iphdr) + sizeof(struct udphdr);
.....
iph->ihl = 5;
iph->version = 4;
iph->tos = 0;
iph->tot_len = sizeof (struct iphdr) + sizeof (struct udphdr) + strlen
iph->id = htonl (54321); //Id of this packet
iph->frag_off = 0;
if(ttl_or_sp == 0) //ovo znaci da cemo koristiti ttl kanal
    iph->ttl = ttl;
else
    iph->ttl = 255;
.....
if(ttl_or_sp == 1)
    udph->source = htons (source_port);
else
    udph->source = htons (6000);
udph->dest = htons (6666);
//Now the UDP checksum using the pseudo header
psh.source_address = inet_addr ( source_ip );
psh.dest_address = sin.sin_addr.s_addr;
psh.placeholder = 0;
psh.protocol = IPPROTO_UDP;
psh.udp_length = htons (sizeof(struct udphdr) + strlen(data) );
.....
udph->check = csum( (unsigned short*) pseudogram , psize);
```

Slika 21: Deo funkcije *fill\_and\_send\_packet()*

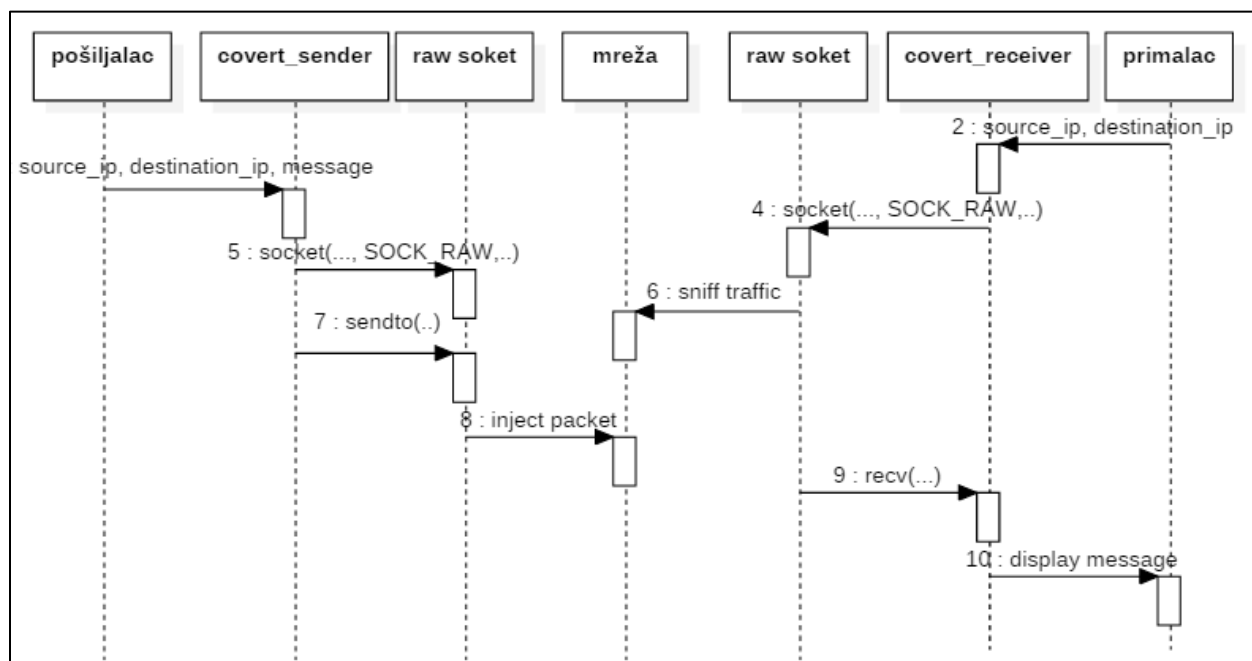
Na isečku iznad je moguće videti kako se pristupa poljima u strukturama *iphdr* i *udphdr* i kako se popunjavaju njihove vrednosti odgovarajućim podacima za paket koji želimo da prosledimo. Pre svega je potrebno očistiti polje *datagram* od mogućih vrednosti koje se u memoriji mogu naći, pa onda inicijalizovati strukture *iphdr* i *udphdr* kastovanjem čitavog datagrama u skladu sa pravilima organizacije paketa. U trećem delu koda se može videti i način na koji se popunjavaju vrednosti u pseudozaglavlju koje je opisano u prethodnom poglavlju u skladu sa



ostatkom paketa. Moguće je videti i način na koji se koristi polje TTL odnosno `source_port` u zavisnosti od parametra prosleđenog funkciji. Izostavljeni su neki delovi koda kojim se popunjavaju ostali delovi zaglavlja IP i UDP paketa, koji nisu toliko bitni za samu aplikaciju.

### 6.5.3. Slanje prikrivenim kanalom

Slanje prikrivenim kanalom se obavlja nakon što se kreira raw soket, učitaju parametri bitni za isporuku, paketi pripreme za slanje, popune sva odgovarajuća polja u njima i pokrene funkcija `sendto(...)` kojom se paket prosleđuje socketu, a socket dalje radi ubacivanje (eng. *inject*) paketa u mrežu. Tok komunikacije prikrivenim kanalom korišćen u aplikaciji prikazan je na sledećem dijagramu sekvence, na slici 22. Ista slika biće korišćena i kasnije pri opisu funkcionisanja prijema podataka.



Slika 22: Proces komunikacije prikrivenim kanalom

Na slici je moguće videti kako se pri komunikaciji izdvaja nekoliko slojeva: sloj interakcije sa korisnikom, sloj interakcije sa socketima i sloj mreže. Svi slojevi na prijemnoj i predajnoj strani su povezani međusobno, međutim sloj mreže, koji je u samoj sredini zapravo služi kao prenosni medijum koji omogućava razmenu podataka između dve strane, bez uspostavljanja direktne veze između dve strane radi komunikacije, pošto strana pošiljaoca zapravo ubacuje pakete u mrežu, a primalac osluškuje sav saobraćaj, pa prihvata i analizira pakete namenjene njemu. Još jedna prednost ovog pristupa i pogodnost u slučaju korišćenja kod prikrivenih kanala jeste u tome da izvorišna i odredišna IP adresa u paketima mogu da budu bilo koja dogovorena adresa sa lokalne mreže, tako da je teže pratiti komunikaciju dve strane i otkriti o kome se zapravo radi.

Isečak koda na kome je prikazano kako se u zavisnosti od moda rada pozivaju funkcije za kreiranje i slanje paketa na mrežu, kao i sam poziv funkcije `sendto(...)` biće prikazani na slici 23.

```

/*kod za pocetak slanja u zavisnosti od moda rada*/
fill_and_send_packet(128, 127 * 255, working_mode-1);
for(i=0; i<strlen(message); i++) {
    if(working_mode==1) {
        for(j=0; j<8; j++) {
            if(binaryMatrix[i][j]==1)
                fill_and_send_packet(255, 0, working_mode-1);
            else
                fill_and_send_packet(64, 0, working_mode-1);
        }
    }
    else {
        int s_port = (int)message[i] * 255;
        printf("Forged source port for character %c: %d \n", message[i], s_port);
        fill_and_send_packet(0, s_port, 1);
    }
}
//poslednji paket sa DEL(128) karakterom - kraj prijema ili 128 u TTL polju
fill_and_send_packet(128, 127 * 255, working_mode-1);
.....

if (sendto (sock_raw, datagram, iph->tot_len ,
           0, (struct sockaddr *) &sin, sizeof (sin)) < 0)
    perror("sendto failed");
else
    printf ("Packet sent. \n");

```

**Slika 23: Slanje podataka prikrivenim kanalom**

Kao što je moguće videti na slici, slanje poruke započinje ubacivanjem početnog paketa u mrežu, koji po dogovoru između pošiljaoca i primaoca izgleda specifično – u slučaju slanja preko *TTL* polja, početni paket odlazi na mrežu sa vrednošću *TTL* jednako 128. U slučaju slanja preko *source\_port* polja, početna vrednost je karakter DEL koji ima vrednost 127 u ASCII tabeli, pomnožen sa 255. Isti paketi se šalju i kada pošiljalac želi da prekine komunikaciju. U slučaju kada se koristi komunikacija preko *TTL* polja, poruka koja je prosleđena kao parametar programa se kodira u matricu binarnih vrednosti, odnosno pravi se binarna matrica  $8 \times \text{dužina\_poruke}$ , gde svaki red u matrici predstavlja po jedan karakter poruke, preveden u svoju binarnu reprezentaciju. Ovo prevođenje iz *char* u *int* pa u binarne vrednosti se obavlja pri pripremi za slanje. Ista vrsta dekodiranja, samo u suprotnom redosledu se kasnije vrši na prijemnoj strani.

U drugom delu isečka koda prikazana je funkcija *sendto(...)*, koja ukoliko je uspešno izvršena vraća 1.

#### **6.5.4. Prijem prikrivenim kanalom**

Proces prijema prijemnim kanalom počinje u trenutku kada na raw soket stigne paket koji u IP zaglavlju u poljima za odredišnu i izvorišnu IP adresu ima adrese koje je korisnik preneo kroz argumente pri startovanju konzolnog programa za prijem. Naravno, prvo je potrebno kreirati raw soket na prijemnoj strani i povezati ga sa željenim interfejsom, u našem slučaju sa *eth0*. Kao što je na slici 22 prikazano, nakon što je kreiran raw soket, isti se onda prebacuje za mod osluškivanja

i poziva se funkcija *receive\_packets()* – isečak ove funkcije je dat na slici 24. Funkcija vrši analizu svih paketa koji dospeju na mrežni interfejs određivanja računara i tek nakon što naiđe na početni paket – započinje prijem i beleženje primljene poruke.

```

if(recvfrom(sock_raw, buffer, 65536, 0, &saddr, (socklen_t*)&saddr_size)<0){
    printf("Error in receiving data.\n");
    exit(-1);
}
else {
    struct iphdr *iph = (struct iphdr *) (buffer + sizeof(struct ethhdr) );
    .....
    if(strcmp(destination_ip_address, inet_ntoa(dest.sin_addr))==0 &&
        strcmp(source_ip_address, inet_ntoa(source.sin_addr))==0){
        if(working_mode == 1){ //prvi kanal - TTL polje
            if((unsigned int)iph->ttl <= 255 && (unsigned int)iph->ttl >64)
                fprintf(logfile,"%d",1);
            else if(iph->ttl <= 64)
                fprintf(logfile,"%d",0);
            else if(iph->ttl==128 && start==1){
                indicator=1;
                printf("Finished receiving data.\n");
            }
            else {
                start=1;
                printf("Started receiving data.\n");
            }
        }
        else { //drugi kanal - source_port polje
            switch (iph->protocol){ //Check the Protocol and do accordingly...
                case 17: //UDP Protocol
                    process_udp_header_data(buffer, data_size);
                    break;

                default:
                    break;
            }
        }
    }
}

```

**Slika 24: Prijem poruke preko Prikrivenog kanala**

Na slici iznad u prvom delu koda može se videti poziv funkcije prijem za prijem paketa sa raw soketa - *recvfrom(..)* koja je opisana u poglavlju 5.2. Nakon što su podaci primljeni i smešteni u bafer, vrši se kastovanje datog bafera redom u IP zaglavlje i UDP zaglavlje po potrebi. Deo o kastovanju je takođe naveden i objašnjen u poglavlju 5.2. pa je ovde preskočen. Kada se u drugom delu koda prvo proveriti izvorišna i odredišna IP adresa i utvrdi da je u skladu sa dogovorenim adresama sa pošiljaocem – započinje prijem.

Kada se za komunikaciju koristi kanal *TTL polje* u IP zaglavlju, onda nije potrebno dodatno kastovanje primljenog bafera, sem u IP zaglavlje, pa je odatle moguće izvući sve podatke koji su potrebni, odnosno IP adrese izvorišta i odredišta i TTL vrednost u paketu, pa u zavisnosti od nje – početi, završiti ili nastaviti prijem. Svaki primljeni paket, odnosno njegova „vrednost“ za naš program koja je 0 ili 1, se beleži u fajl koji je na početku kreiran – *messageLog.txt*. Ovaj fajl je privremeno kreiran zbog toga što ne znamo koliko ćemo tačno paketa primiti, a pošto se primaju

karakteri od kojih svaki ima po 8 bitova, lakše je upisivati u fajl pa posle proći kroz njega i dekodirati iste, kako bi bili prikazani korisniku u konzoli.

Kada se za komunikaciju koristi kanal *source\_port* u UDP zaglavlju, potrebno je naknadno obraditi paket, nakon što je on bio kastovan u IP zaglavlje – potrebno ga je kastovati i u UDP zaglavlje, da otkrijemo vrednosti polja u njemu. Pre toga je potrebno proveriti da li je paket koji je stigao sa odgovarajućom IP adresom izvorista i odredišta u stvari i UDP paket, tj. da li je namenjen UDP protokolu. Ovo se radi u *switch* iskazu, gde se proverava da li je protokol polje u IP zaglavlju jednako 17, što je kod za UDP protokol. Ukoliko jeste, poziva se funkcija *process\_udp\_header\_data()* čije je delovanje opisano u poglavlju 6.4.2, a njen kratak isečak je dat na slici 25.

```
unsigned short iphdrlen;
struct iphdr *iph = (struct iphdr *) (buffer + sizeof(struct ethhdr));
iphdrlen = iph->ihl*4;
struct udphdr *udph = (struct udphdr *) (buffer + iphdrlen + sizeof(struct ethhdr));
int source_port = ntohs(udph->source);
if (source_port == 32385 && start==1) { //ovu vrednost dobijamo kad 127 pomnozimo sa 255
    indicator=1; //a DEL karakter ce biti granicnik za pocetak i kraj prijema podataka
    printf("Finished receiving data.\n");
    return;
}
else if (source_port == 32385 && start==0) {
    start=1;
    printf("Started receiving data.\n");
    return;
}
printf("Source port: %d \n", source_port);
char c = source_port / 255;
printf("ASCII value %d \t of character%c \n", c, c);
```

```
struct udphdr {
    __u16 source;
    __u16 dest;
    __u16 len;
    __u16 check;
};
```

Slika 25: Funkcija *process\_udp\_header\_data(..)*

Na početku koda vidimo standardno kastovanje ranije objašnjeno za odgovarajuću memorijsku strukturu, a desno se nalazi uokvirena struktura koja u Linux kernelu predstavlja definiciju zaglavlja UDP paketa i nalazi se u fajlu *udp.h*. Nakon kastovanja vrši se provera vrednosti izvorišnog porta u paketu i ukoliko je vrednost porta jednaka broju 127\*255, onda započinje prijem paketa ili kraj prijema, a ukoliko je različita, onda se dekodira, odnosno deli sa 255 i vrednost primljenog karaktera štampa na standardni izlaz.

U narednom poglavlju biće dati primeri rada aplikacije i skrinšotovi na prijemnoj i predajnoj strani.

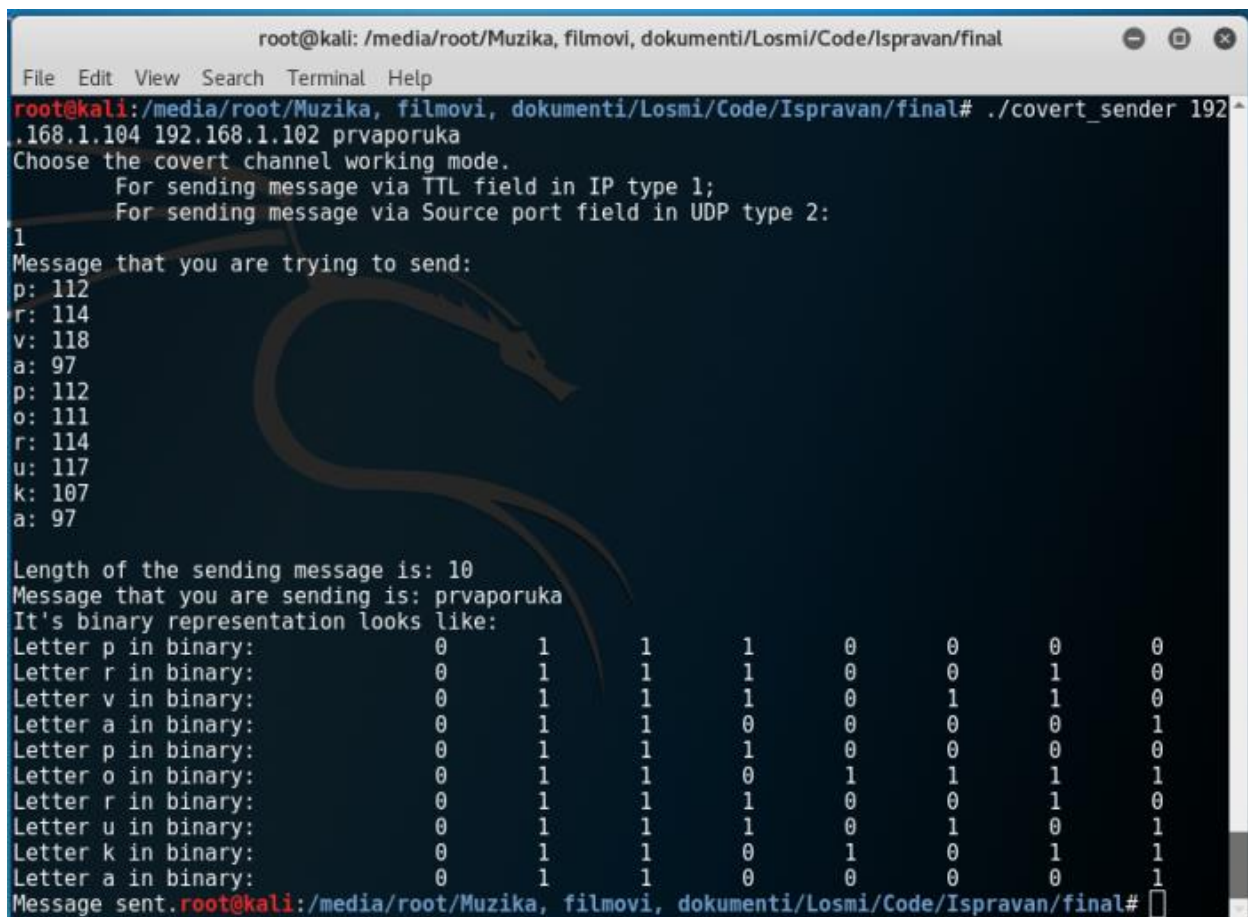
## 6.6. Primer implementacije prikrivenih kanala

U svrhu prikaza rada projektovana aplikacija je testirana u dva slučaja: slanje prikrivene poruke *TTL* kanalom, kao i slanje preko drugog implementiranog kanala – *source port* u UDP zaglavlju. Testiranje rada aplikacije je izvršeno na dva računara povezana Ethernet kablom na ruter u lokalnoj mreži:

1. Računar – Intel Core i3-3110m 2.40GHz procesor, sa RAM memorijom 6GB DDR3 na 798 MHz. Na njemu je pokrenut operativni sistem Linux sa verzijom kernela 4.12.9, a distribucija Linuxa je bila *Kali Linux*. Posedovao je mrežnu LAN karticu brzine 100Mb/s.
2. Računar – AMD A8 6410 2.0 GHz, sa RAM memorijom od 6GB DDR3 na 1033MHz. Na njemu je bio pokrenut operativni sistem Linux sa verzijom kernela 4.11.9, a distribucija Linuxa je takođe bila *Kali Linux*. Posedovao je mrežnu LAN karticu brzine 100Mb/s.

## Primer 1

U ovom primeru su biće izgledi ekrana aplikacije koja radi u modu slanja podataka *TTL* prikrivenim kanalom. Na *slici 26* je dat prikaz strane pošiljaoca, a na *slici 27* strane primaoca.



```

root@kali: /media/root/Muzika, filmovi, dokumenti/Losmi/Code/Ispravan/final
File Edit View Search Terminal Help
root@kali:/media/root/Muzika, filmovi, dokumenti/Losmi/Code/Ispravan/final# ./covert_sender 192.168.1.104 192.168.1.102 prvaporuka
Choose the covert channel working mode.
  For sending message via TTL field in IP type 1;
  For sending message via Source port field in UDP type 2:
1
Message that you are trying to send:
p: 112
r: 114
v: 118
a: 97
p: 112
o: 111
r: 114
u: 117
k: 107
a: 97

Length of the sending message is: 10
Message that you are sending is: prvaporuka
It's binary representation looks like:
Letter p in binary: 0 1 1 1 0 0 0 0
Letter r in binary: 0 1 1 1 0 0 1 0
Letter v in binary: 0 1 1 1 0 1 1 0
Letter a in binary: 0 1 1 0 0 0 0 1
Letter p in binary: 0 1 1 1 0 0 0 0
Letter o in binary: 0 1 1 0 1 1 1 1
Letter r in binary: 0 1 1 1 0 0 1 0
Letter u in binary: 0 1 1 1 0 1 0 1
Letter k in binary: 0 1 1 0 1 0 1 1
Letter a in binary: 0 1 1 0 0 0 0 1
Message sent.root@kali:/media/root/Muzika, filmovi, dokumenti/Losmi/Code/Ispravan/final#

```

Slika 26: Slanje TTL prikrivenim kanalom

```

root@kali:~/Downloads# ./covert_receiver 192.168.1.104 192.168.1.102
Choose the covert channel working mode.
For receiving message via TTL field in IP type 1;
For receiving message via Source port field in UDP type 2:
1
Starting...
Started receiving data.
Finished receiving data.
Message in binary:
0 1 1 1 0 0 0 0
0 1 1 1 0 0 1 0
0 1 1 1 0 1 1 0
0 1 1 0 0 0 0 1
0 1 1 1 0 0 0 0
0 1 1 0 1 1 1 1
0 1 1 1 0 0 1 0
0 1 1 0 1 0 1 1
0 1 1 0 0 0 0 1
Message length is : 10
And the message is:
prvaporuka
Finished receiving and displaying data.
root@kali:~/Downloads#

```

Slika 27: Prijem TTL prikrivenim kanalom

## Primer 2

U ovom primeru su dati izgledi ekrana aplikacije kada radi u modu slanja i prijema poduka putem *source port* polja u UDP paketima. Na *slici 28* je dat prikaz strane pošiljaoca, a na *slici 29* strane primaoca.

```

Message sent.root@kali:~/media/root/Muzika, filmovi, dokumenti/Losmi/Code/Ispravan/final# ./cove
.168.1.104 192.168.1.102 prvaporuka
Choose the covert channel working mode.
For sending message via TTL field in IP type 1;
For sending message via Source port field in UDP type 2:
2
Message that you are trying to send:
p: 112
r: 114
v: 118
a: 97
p: 112
o: 111
r: 114
u: 117
k: 107
a: 97

Forged source port for character p: 28560
Forged source port for character r: 29070
Forged source port for character v: 30090
Forged source port for character a: 24735
Forged source port for character p: 28560
Forged source port for character o: 28305
Forged source port for character r: 29070
Forged source port for character u: 29835
Forged source port for character k: 27285
Forged source port for character a: 24735

```

Slika 28: Slanje UDP source port prikrivenim kanalom



```
root@kali: ~/Downloads
File Edit View Search Terminal Help

root@kali:~/Downloads# ./covert_receiver 192.168.1.104 192.168.1.102
Choose the covert channel working mode.
  For receiving message via TTL field in IP type 1;
  For receiving message via Source port field in UDP type 2:
2
Starting...
Started receiving data.
Source port: 28560
ASCII value: 112      of character: p
Source port: 29070
ASCII value: 114      of character: r
Source port: 30090
ASCII value: 118      of character: v
Source port: 24735
ASCII value: 97       of character: a
Source port: 28560
ASCII value: 112      of character: p
Source port: 28305
ASCII value: 111      of character: o
Source port: 29070
ASCII value: 114      of character: r
Source port: 29835
ASCII value: 117      of character: u
Source port: 27285
ASCII value: 107      of character: k
Source port: 24735
ASCII value: 97       of character: a
Finished receiving and displaying data.
root@kali:~/Downloads#
```

Slika 29: Prijem poruke UDP source port prikrivenim kanalom

## 7. Zaključak

U ovom radu je predstavljena tehnika za implementaciju *prikrivenih kanala* za komunikaciju u mreži i projektovanje aplikacije koja ilustruje rad ovih kanala. Implementacijom kanala je omogućeno da krajnji računari u lokalnoj mreži mogu da komuniciraju bez mogućnosti da njihova komunikacija može sa sigurnošću bude otkrivena i dešifrovana “običnim” nadgledanjem mreže i ukoliko ne postoji pretpostavka da je baš određeni tip kanala uspostavljen. U radu je najpre posvećena pažnja računarskim mrežama i najbitnijim protokolima, zatim je predstavljen sam pojam prikrivenih kanala i neke od češćih implementacija, nakon čega je usledio i opis programiranja *raw* soketa, čime se omogućuje dublji pristup formiranju mrežnih paketa. Na kraju je dat primer rada programa i kodiranje, prenošenje i dekodiranje poruka između dva računara na lokalnoj mreži.

Aplikacija je napravljena tako da šalje tekstualne poruke koje joj korisnik prosledi odabranom primaocu, putem prikrivenih kanala za komunikaciju na mreži. Poruke koje je trebalo proslediti su prilagođavane odabranom tipu prikrivenog kanala i slanju preko istog. Data aplikacija ima dobre i loše strane. Loša strana se ogleda u tome da je implementirane kanale moguće detektovati detaljnijom analizom saobraćaja i praćenjem ponašanja svakog polja u paketima ponaosob u dužem vremenskom periodu. Sa druge strane, implementirane kanale je teže detektovati i zaustaviti u odnosu na druge prikrivene kanale koji koriste naivnije pristupe, tj. kodiranje karaktera i bitova direktno u poljima zaglavlja paketa. Pristup iskorišćen u radu je bolji u odnosu na ove *naivne* pristupe, zato što polja koja su prikrivenim kanalima iz rada izmenjena retko podležu kontroli i menjanju od strane zaštitnog zida ili rutera. U dobru stranu spada i proširljivost aplikacije, odnosno mogućnost dodavanja novih kanala, kao i njeno moguće prilagođenje slanju tekstualnih i binarnih datoteka.

Nakon uvoda, u drugom poglavlju, objašnjen je pojam računarskih mreža, njihova struktura i korišćenje. Takođe, dat je istorijski osvrt na razvoj Interneta i uloge koje on omogućuje, kao i osvrt na hijerarhiju protokola, koja leži u osnovama funkcionisanja Interneta.

Zatim je u trećem poglavlju predstavljen OSI referentni model. Struktura ovog modela je pogodna za predstavljanje principa funkcionisanja slojeva i protokola koji se provlače kroz funkcionalnost implementirane aplikacije. Predstavljeni su i objašnjeni svi slojevi ovog modela, a bačen je akcenat i na najvažnije protokole i njihove karakteristike koje su od značaja za ispunjenje cilja ovog rada.

Nakon toga je u četvrtom poglavlju usledilo objašnjenje pojma sigurne komunikacije na Internetu, njenih karakteristika i samog pojma prikrivenih kanala koji su i tema ovog rada.

Zatim, u petom poglavlju su objašnjeni *raw* soketi i ograničenja odnosno mogućnosti koje pruža programiranje soketa u različitim operativnim sistemima, sa fokusom na Linux.

U šestom poglavlju predstavljena je aplikacija projektovana za komunikaciju prikrivenim kanalima. Dat je pogled na njenu logičku arhitekturu, opis svih implementiranih kanala i njihovog načina rada, prednosti i mana. Takođe, dati su i delovi koda koji pružaju bolji uvid u implementaciju aplikacije. Na kraju je dat primer rada aplikacije.



## 8. Spisak slika

|   |    |
|---|----|
| Slika 1: Slojevi, protokoli i interfejsi [3] .....                            | 10 |
| Slika 2: Referentni model OSI i komunikacija među slojevima [3] .....         | 11 |
| Slika 3: Struktura Ethernet okvira [3] .....                                  | 13 |
| Slika 4: Zaglavlje IP protokola [3].....                                      | 15 |
| Slika 5: Zaglavlje TCP protokola [3].....                                     | 17 |
| Slika 6: UDP zaglavlje [3] .....  | 19 |
| Slika 7: Problem zatvorenika [9] .....  | 22 |
| Slika 8: Put mrežnog paketa kroz protokol stek [8].....                       | 27 |
| Slika 9: Način rada raw soketa.....   | 28 |
| Slika 10: Hederi koje je potrebno uključiti za rad sa raw soketima.....       | 29 |
| Slika 11: Kreiranje raw soketa i njegovo povezivanje na mrežni interfejs..... | 30 |
| Slika 12: Pozivanje funkcije za prijem podataka na raw socketu .....          | 31 |
| Slika 13: Format adrese koji se koristi kod soketa.....                       | 32 |
| Slika 14: Izdvajanje IP zaglavlja iz paketa primljenog sa raw soketa .....    | 32 |
| Slika 15: Struktura IP adrese koja se koristi kod soketa .....                | 32 |
| Slika 16: Logička arhitektura sistema .....                                   | 34 |
| Slika 17: Komponente koje sačinjavaju sistem.....                             | 35 |
| Slika 18: Klasni dijagram covert_sender komponente .....                      | 35 |
| Slika 19: Klasni dijagram komponente covert_receiver .....                    | 37 |
| Slika 20: Odabir moda rada aplikacije .....                                   | 38 |
| Slika 21: Deo funkcije fill_and_send_packet().....                            | 39 |
| Slika 22: Proces komunikacije prikrivenim kanalom .....                       | 40 |
| Slika 23: Slanje podataka prikrivenim kanalom .....                           | 41 |
| Slika 24: Prijem poruke preko Prikrivenog kanala .....                        | 42 |
| Slika 25: Funkcija process_udp_header_data(..) .....                          | 43 |
| Slika 26: Slanje TTL prikrivenim kanalom .....                                | 44 |
| Slika 27: Prijem TTL prikrivenim kanalom.....                                 | 45 |
| Slika 28: Slanje UDP source port prikrivenim kanalom.....                     | 45 |
| Slika 29: Prijem poruke UDP source port prikrivenim kanalom .....             | 46 |

## 9. Literatura

- [1] D. P. A. a. Q.-A. Zeng, Introduction to Wireless and Mobile Systems (2nd Edition), Thomson, 2005.
- [2] R. Soltani, B. Bash, D. Goeckel, S. Guha i D. Towsley, Covert single-hop communication in a wireless network with distributed artificial noise generation, Liverpool, 2014.
- [3] S. A. Tanenbaum, Computer networks, 2005.
- [4] A. J. P. d. M. Veinović, Uvod u računarske mreže, Beograd: Čugura print, 2008.
- [5] [Na mreži]. Available: [https://en.wikipedia.org/wiki/World\\_Wide\\_Web](https://en.wikipedia.org/wiki/World_Wide_Web). [Poslednji pristup 18 August 2017].
- [6] „Istorija i razvoj Interneta,“ [Na mreži]. Available: <https://fmk117108.wordpress.com/2008/12/13/istorija-i-razvoj-interneta/>. [Poslednji pristup 18 August 2017].
- [7] „What is Ethernet?,“ [Na mreži]. Available: <http://searchnetworking.techtarget.com/definition/Ethernet>. [Poslednji pristup 20 August 2017].
- [8] „Covert Channel Analysis,“ [Na mreži]. Available: <https://fas.org/irp/nsa/rainbow/tg030.htm#2.0>. [Poslednji pristup 22 August 2017].
- [9] „A Survey of Covert Channels and Countermeasures in Computer Network Protocols,“ [Na mreži]. Available: <http://caia.swin.edu.au/cv/szander/publications/szander-ieee-comst07.pdf>. [Poslednji pristup 22 August 2017].
- [10] „Practical Internet Steganography: Data Hiding in IP,“ u *Proc. Texas Wksp. Security of Information*, Houston, 2003.
- [11] „Covert Channels in TCP and IP Headers,“ 2003. [Na mreži]. Available: <http://www.defcon.org/images/defcon-10/dc-10-presentations/dc10-hintz-covert.ppt>. [Poslednji pristup 23 August 2017].
- [12] T.Graf, „Messaging over IPv6 Destination Options,“ 2003. [Na mreži]. Available: <http://gray-world.net/papers/messip6.txt>. [Poslednji pristup 23 August 2017].
- [13] „Covert channels in the TCP/IP protocol suite,“ [Na mreži]. Available: <http://firstmonday.org/ojs/index.php/fm/article/view/528/449>.
- [14] „What is a Socket?,“ [Na mreži]. Available: [https://www.tutorialspoint.com/unix\\_sockets/what\\_is\\_socket.htm](https://www.tutorialspoint.com/unix_sockets/what_is_socket.htm). [Poslednji pristup 24 August 2017].
- [15] K. W. R. James F. Kurose, Computer Networking: A top-down approach featuring the Internet, 2000.

