Prijenos datoteka pomoću TCP/IP protokola

Čarić, Stipe

Sveučilište u Splitu

Prirodoslovno-matematički fakultet

Odjel za informatiku i tehniku

Stipe Čarić

PRIJENOS DATOTEKA POMOĆU TCP/IP PROTOKOLA

Završni rad

Split, 2023

Temeljna dokumentacijska kartica

Završni rad

Sveučilište u Splitu
Prirodoslovno-matematički fakultet
Odjel za informatiku
Ruđera Boškovića 33, 21000 Split, Hrvatska

Prijenos datoteka pomoću TCP/IP protokola

Stipe Čarić

SAŽETAK

Cilj ovog završnog rada bio je analizirati ključne aspekte i mehanizme koji čine TCP/IP protokole pouzdanim sredstvom za prijenos datoteka. U radu su razmatrani osnovni koncepti TCP/IP protokola, uključujući funkcionalnosti poput segmentacije podataka, uspostavljanja i zatvaranja veza te osiguranja pouzdanog prijenosa putem potvrde primitka. Također, istražene su prednosti i ograničenja ovog pristupa prijenosu datoteka u usporedbi s drugim protokolima.

Ključne riječi: Internetski protokoli, IP adresa, TCP protokol, UDP protokol, TCP/IP paket protokola, slojevi internetskog protokola

Rad je pohranjen u knjižnici Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Rad sadrži: 60 stranica, 45 grafičkih prikaza, 1 tablicu i 0 literaturnih navoda.

Izvornik je na hrvatskom jeziku.

Mentor: **Dr.sc. Divna Krpan**, docent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Ocjenjivači: **Dr.sc. Divna Krpan**, docent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Dr.sc. Saša Mladenović, izvanredni profesor Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Dr.sc. Goran Zaharija, docent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Rad prihvaćen: Rujan, 2023.

Basic documentation card

Thesis

University of Split
Faculty of Science
Department of Computer Science
Ruđera Boškovića 33, 21000 Split, Croatia

File transfer using TCP/IP Protocol

Stipe Čarić

ABSTRACT

The aim of this paper was to analyze the key aspects and mechanisms that make TCP/IP protocols a reliable means for file transmission. The paper examines the fundamental concepts of TCP/IP protocols, including functionalities such as data segmentation, establishment and termination of connections, and ensuring reliable transmission through acknowledgment mechanisms. Furthermore, the advantages and limitations of this approach to file transmission were explored in comparison with other protocols.

Key Words: Internet protocols, IP Address, TCP Protocol, UDP Protocol, TCP/IP Packet Suite, Layers of internet protocols

Thesis deposited in library of Faculty of science, University of Split.

Thesis consist of: 60 pages, 45 figures, 1 tables and 0 references

Original language: Croatian

Supervisor: **Divna Krpan**, **Ph.D.** Assistant Professor of Faculty of Science, University of Split

Reviewers: **Divna Krpan**, **Ph.D.** Assistant Professor of Faculty of Science, University of Split,

Saša Mladenović, Ph.D. Associate Professor of Faculty of Science, University of Split,

Goran Zaharija, Ph.D. Assistant Professor of Faculty of Science, University of Split

Thesis accepted: September, 2023.

Sadržaj

1. Uvod u internetske protokole 2

1.1. Arhitektura računalnih mreža 2

1.2. Jedinice informacija 3

1.3. Internetski protokol 4

1.4. Internet datagram 5

1.4.1. Format IPv4 datagrama 5

1.4.2. Format IPv6 Datagrama 7

1.5. IP adresa 9

1.5.1. IPv4 adresa 10

1.5.2. IPv6 adrese 12

1.6. ICMP 16

1.7. ARP 17

2. OSI model i TCP/IP paket protokola 21

2.1. OSI model 22

2.1.1. Slojevi OSI modela 26

2.1. TCP/IP paket protokola 31

2.1.1. Slojevi TCP/IP paketa protokola 33

2.1.2. Prijenosni protokoli TCP/IP paketa protokola 38

3. Praktični dio 50

4. Zaključak 57

Literatura 58

Sadržaj slika 59

Sadržaj tablica 60

Uvod

Danas je prijenos podataka između računalnih mreža sve važniji i važniji. Internetski protokol, pogotovo TCP/IP paket protokola što je skraćenica za *Transmission Control Protocol/Internet Protocol*, je postao temeljni standard za komunikaciju i prijenos podataka u mreži. Da nema internetskog protokola, ne bi bilo ni interneta, što znači da bi bilo puno teže doći do informacija.

Tema ovog rada je upoznati čitatelja sa općenitim pojmom internetskog protokola i fokusirati se na prijenos podataka pomoću internetskog protokola sa jednog uređaja na drugi.

U prvom poglavlju ćemo se upoznati sa pojmom internetskog protokola, poviješću i općim pojmovima koji se koristu sa internetskim protokolima za prijenose podataka. Važno je da prođemo kroz sve pojmove koji se koristu da bi mogli u detalje objasniti čitatelju kako ovo sve radi.

U drugom poglavlju ćemo se spomenuti OSI model i njegov način rada, te se onda prebaciti u potpunosti na ono što nas zanima u ovom završnom radu, a to je TCP/IP paket protokola. Upoznati ćemo se sa njegovim značajkama i onda ćemo u trećem poglavlju u praksi pokazati kako pomoću njega možemo slati datoteke sa jednog računala na drugi.

Cilj ovog završnog rada je upoznati čitatelja da ima više načina za prijenos datoteka u lokalnoj mreži između uređaja te na praktičnom primjeru ga pokušati zainteresirati za ovakav način prijenosa datoteka i pogurati ga u istraživanje ove teme još dublje i možda kreiranje njegove aplikacije za ovu upotrebu jer ih u ovom trenutku nema puno.

1. Uvod u internetske protokole

1.1. Arhitektura računalnih mreža

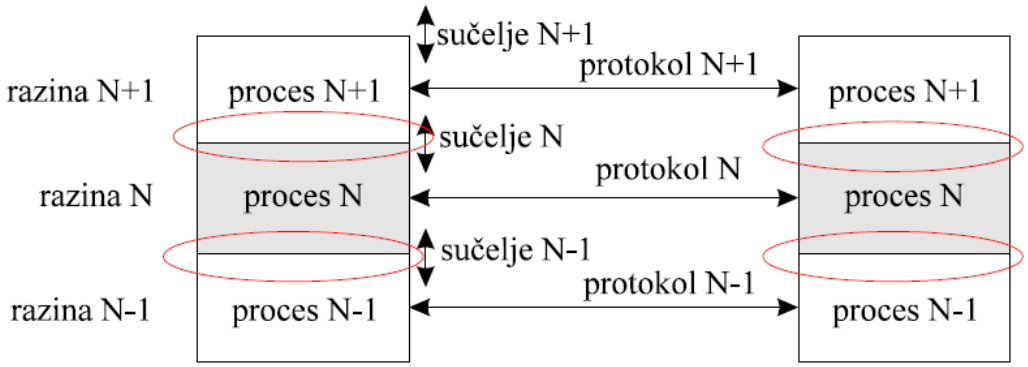
Računalna mreža se gradi hijerarhijski, tj. po slojevima. Za svaku razinu se definira protokol i sučelje sa drugim hijerarhijskim slojem. Svaki sloj obavlja određenu funkciju i pri tome koristi uslugu koju pruža niži sloj, a obavljajući svoju funkciju pruža višem sloju. Hijerarhijska struktura se sastoji od sljedećih koncepata:

- Koncept razine (eng. *Layer*)
- Koncept sučelja (eng. *Interface*)
- Koncept protokola (eng. *Protocol*) – skup pravila po kojima komuniciraju umreženi uređaji
- Koncept zaglavlja (eng. *Header*)
- Koncept fragmentacije (eng. *Fragment*)
- Koncept informacije

Odvajanjem mreže u slojeve dobivamo puno pogodnosti kao:

- Mrežna komunikacija svedena je na manje, jednostavnije dijelove
- Standardizacija mrežnih komponenti koja omogućava razvoja od više proizvođača
- Mogućnost komunikacije različitih tipova mrežnog hardvera i softvera
- Promjena na jednom sloju ne utječe na druge slojeve
- Mrežna komunikacija svedena je na manje komponente

Standardizacijom sučelja omogućavamo komunikaciju među procesima susjednih razina i komunikaciju unutar istog uređaja. Sučelja su dvosmjerna, a svaka razina komunicira preko 2 sučelja: preko gornjeg prema nadređenoj razini i preko donjeg prema podređenoj razini.



Slika 1.Koncept sučelja i razina (preuzeto iz [1])

1.2. Jedinice informacija

Internetski protokol koristi jedinice informacija za prijenos podataka između računalnih mreža [2]. Neke od najvažnijih su:

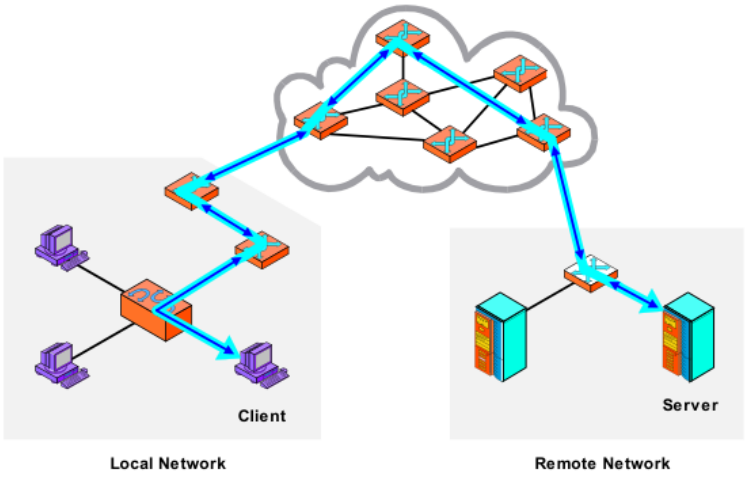
- Bit (eng. *Binary Digit*) – najmanja jedinica informacije koja predstavlja osnovnu binarnu vrijednost koja može biti 0 ili 1, a prenosimo ga na fizičkoj razini.
- Oktet (eng. *Bayt*) – skup od 8 bitova ili najmanja kodna riječ koja se često koristi za prijenos znakova, brojeva i drugih podataka. Oktet se nekad obrađuje na fizičkoj razini, a nekad na podatkovnoj razini.
- Okvir (eng. *Block*) – osnovni PDU (eng. *Protocol Data Unit*) podatkovne razine koji se sastoji od više okteta ili znakova. Početak okvira je određen sinkronizacijskom sekvencom koju zovemo okvirni znak. Okvir je najmanji PDU koji ima vlastito zaglavlje. Obavlja se provjera adrese odredišta i cjelovistosti okvira, a u slučaju oštećenja okvir će se odbaciti
- Paket (eng. *Packet*) – osnovni PDU mrežne razina s kojim se obavlja promet od početka do kraja mreže. Sadrži informaciju odredišta i nastoji prenjeti jednim okvirom podatkovne razine.
- Datagram – osnovni PDU prijenosne razine koji se koristi u IP (eng. *Internet Protocol*) komunikaciji. Predstavlja samostalnu jedinicu podataka koja se šalje od izvora do odredišta putem mreže.
- Segment – osnovni PDU prijenosne razine koji se koristi na transportnom sloju TCP/IP protokola. U TCP (eng. *Transmission Control Protocol*) komunikaciji, podaci se dijelu na manje segmente od kojih svaki sadrži informacije kao broj sekvenci i provjera grešaka kako bi se osigurala pouzdanost prijenosa.

1.3. Internetski protokol

Internetski protokol je najvažniji mrežni protokol za prijenos podataka kojeg koristu izvorni i odredišni uređaj za uspostavu podatkovne komunikacije preko računalne mreže. Internetski protokol osigurava vezu bez spajanja (eng. *Connectionless Protocol*), što znači da se dvije strane ne dogovaraju o početku ili završetku prijenosa podataka, ne dogovaraju prijenosni put, nego predajna strana šalje podatke i ako nakon nekog vremena ne dobije potvrdu o uspješnom prijemu, šalje podatke ponovno. Ovo znači da ako pošaljemo nekoliko paketa, on se u procesu prijenosa može promijeniti, može se promijeniti redoslijed paketa u odnosu na onaj redoslijed kojim su oni poslani sa izvora, mogu se duplicirati ili potpuno izgubiti tijekom prijenosa. Internetski protokol ne razumije upravljačke podatke za uspostavu veze s kraja na kraj mreže, nego se oslanja na protokole drugih slojeva koji trebaju uspostaviti vezu ako žele da to bude veza sa spajanjem [2].

Internetski protokol se oslanja na protokole viših i nižih slojeva za osiguravanje korekcije i detekcije pogreški, zbog čega se često zove „nepouzdani protokol“, odnosno ne garantira uspješnu dostavu paketa na odredište. Internetski protokol će prenjeti podatke mrežom, ali neće provjeriti jesu li podaci točno preneseni nego će to obaviti protokoli ostalih slojeva u TCP/IP arhitekturi.

Uređaji koji preusmjeravaju pakete na njihovom putu kroz mrežu se zovu usmjerivači (eng. *Router*). S obzirom na to da je sam koncept internetskih protokola oslobođen mehanizam, to znači da usmjerivači mogu presložiti raspored, promijeniti ili izgubiti pakete ali da to ne utječe previše na samu uslugu.

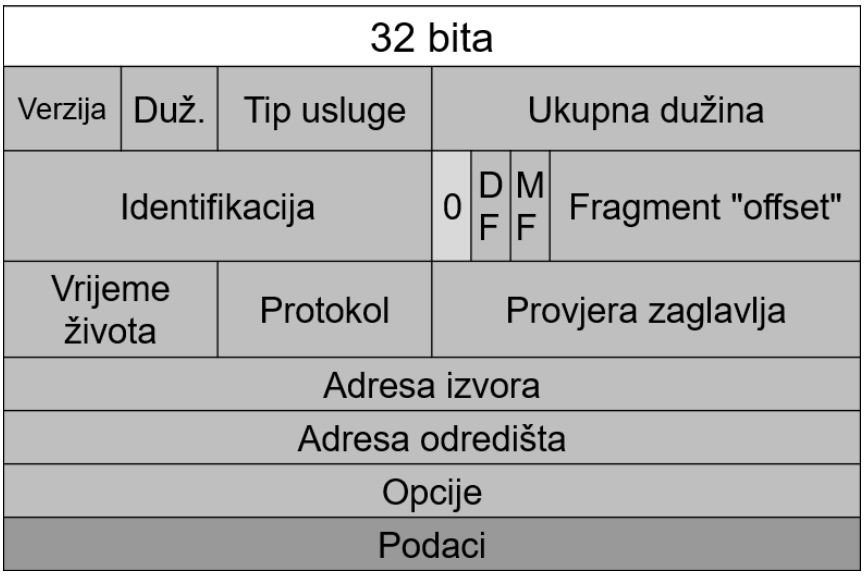


Slika 2. Dostava paketa preko internetskog protokola (preuzeto iz [3])

1.4. Internet datagram

Internet datagram (IP datagram) je osnovni paket za prijenos podataka [1]. Datagram ima zaglavlje koje sadrži informacije za internetski protokol i podatke koji su značajni protokolima viših slojeva. Svaki datagram se obrađuje nezavisno, zbog čega se datagrami mogu pojaviti krivim redoslijedom na odredištu jer se i njihovo usmjeravanje obavlja nezavisno. Imamo 2 formata IP datagrama: IPv4 i IPv6 format datagrama.

1.4.1. Format IPv4 datagrama



Slika 3. Format IPv4 datagrama (preuzeto iz [1])

Na slici 3. je prikazan format IPv4. Sad ćemo proći kroz sve dijelove IPv4 datagrama.

Verzija nam govori o verziji internetskog protokola i ona je 4bitna.

Dužina je isto veličine 4 bita, a govori nam koja je dužina IP zaglavlja u broju 32-bitnih jedinica, ali nije uključeno polje korisničkih podataka.

Tip usluge je veličine 8 bita, a on nam govori o kvaliteti usluge za taj datagram, da li datagram može kasniti, da li se traži normalna ili visoka propusnost i da li je poželjan pouzdan prijenos ili ne.

Ukupna dužina je veličine 16 bita, a govori o ukupnoj dužini datagrama, zaglavlja i podataka koji su u broju byte-ova.

Identifikacija je jedinstveni broj koji upisuje pošaljitelj kako bi se olakšalo sastavljanje fragmentiranih datagrama. Svi dijelovi istog datagrama će imati iste identifikacijske brojeve. Identifikacija je veličine 16 bita.

DF i MF su veličine jednog bita, a imaju 2 stanja: 0 ili 1. DF nam govori da li je dozvoljeno fragmentiranje, a MF nam govori da li je zadnji fragment ili ima još fragmenata. Pomak fragmenata (eng. *Fragment Offset*) je veličine 13 bitova, a koristi se u fragmentiranim datagramima. Kod prvog ili jedinog datagrama ova vrijednost je uvijek 0.

Vrijeme života određuje vrijeme u sekundama koliko ovaj datagram smije putovati. Veličine je 8 bitova. Svaki put kada datagram prođe kroz neki usmjerivač, oduzima se od ovog broja vrijeme obrade datagrama. Obično je vrijeme obrade manje od sekunde, a oduzima se sa 1. Tako ovo polje TTL (eng. *Time to live*) postaje brojač „skakanja“ u mreži. Kada TTL dođe do 0, datagram se odbacuje.

Protokol je veličine 8 bitova, a određuje protokol višeg sloja kome internetski protokol dostavlja podatke npr. TCP (eng. *Transmission Control Protocol*), UDP (eng. *User Datagram Protocol*)…

Provjera zaglavlja predstavlja kontrolni zbroj zaglavlja bez podataka i veličine je 16 bitova.

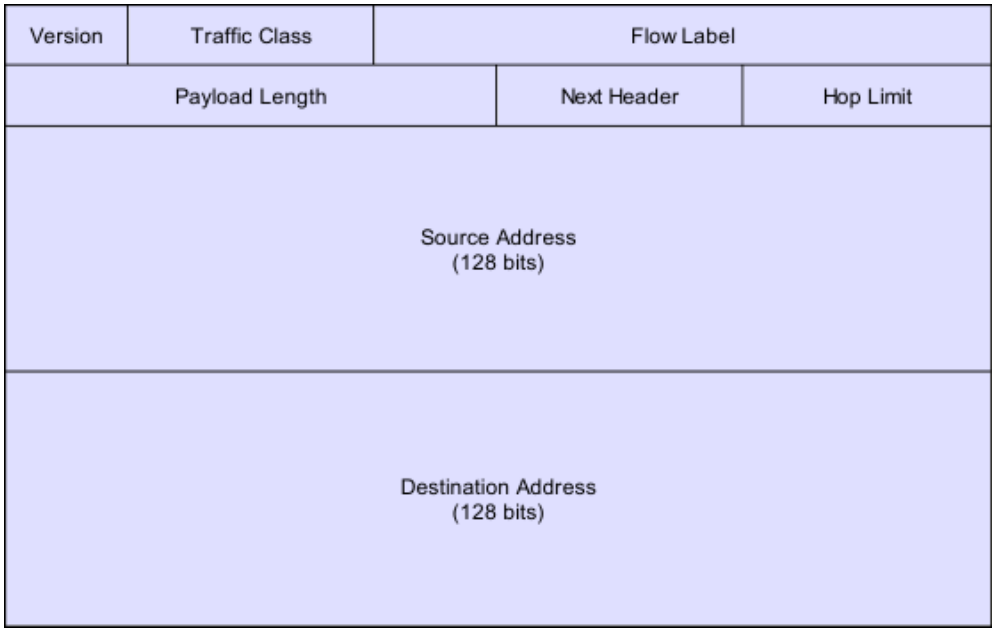
Adresa izvora je 32-bitna IPv4 adresa računala koja šalje datagram.

Adresa odredišta je 32-bitna IPv4 adresa odredišnog računala.

Opcija definira dodatne opcije koje omogućava internetski protokol, promjenjive je duljine, a sastoji se od: koda opcije, dužine opcije i podataka.

Podaci sadrži podatke, koji se prenosu datagramom, koji se predaju protokolima na višim slojevima koji su određeni poljem protokol.

1.4.2. Format IPv6 Datagrama



Slika 4. Format IPv6 datagrama (preuzeto iz [2])

Na slici 4 je prikazan format IPv6 datagrama koji je ponešto drugačiji od formata IPv4 datagrama.

Verzija je ista kao i kod IPv4 datagrama, samo ovdje ima vrijednost 6 jer koristimo IPv6 datagrame.

Tip prometa (eng. *Trafic Class*) određuje prioritet paketa, a može imati vrijednost 0-7 koju predaje prijenosu koji može pričekati u slučaju zagušenja ili može imati vrijednost 8-15 koja je namijenjena za prijenos u realnom vremenu.

Oznaka toka (eng. *Flow Label*) omogućava izvoru i odredištu da uspostave vezu sa posebnim svojstvima i zahtjevima za aplikacije u realnom vremenu.

Duljina (eng. *Payload Length*) govori koliko byteova slijedi iza 40 bytnog zaglavlja pošto se zaglavlje više ne računa u ukupnoj duljini kao u IPv4.

Iduće zaglavlje (eng. *Next Header*) govori koje zaglavlje dolazi iza postojećeg, a ako je zadnje zaglavlje, tada kaže kojem prijenosnom protokolu se predaju paketi. Imamo 6 dodatnih zaglavlja (eng. *Extension* Header) koja se uvodu da bi se nadoknadila neka polja kojih nema za razliku od IPv4 datagrama, a to su:

- *Hop-by-hop options* – raznovrsne informacije za usmjerivače
- *Destination options* – dodatne informacije za odredište
- *Routing* – lista usmjerivača koje paket treba posjetiti
- *Fragmentation* – upravljanje fragmentima datagrama
- *Authentication* – provjera identiteta pošaljitelja
- *Encrypted security payload* – informacije o šifriranom (eng. *Encrypted*) sadržaju

Ograničenje skoka (eng. *Hop Limit*) onemogućava pakete da traju vječno. Svaki paket ima početnu vrijednost koja se smanjuje nakon svakog skoka do nekog usmjerivača.

Adresa izvora (eng. *Source Address*) i adresa odredišta (eng. *Destination Address*) zauzimaju svaka po 128 bita. O formatu adresa ćemo pričati u sljedećem poglavlju.

1.5. IP adresa

IP adresa je jedinstveni broj pomoću kojeg označavamo uređaje na lokalnoj mreži ili internetu. Kada paket internetskog protokola stigne do usmjerenika, gleda se adresa odredišta u tablici usmjerivanja. Ako je paket za neku udaljenu mrežu, prosljeđuje se sljedećem usmjereniku, a ako je u lokalnoj mreži, šalje se direktno na odredište. No ako se adresa ne nalazi u tablici usmjeravanja, onda se šalje usmjereniku sa većom tablicom usmjeravanja [4].

Internetske adrese dijelimo u nekoliko klasa, tj. raspona adresa koji se koristu za različite primjene.

Svako računalo koje je povezano na internet ima jednoznačno dodijeljenu IP adresu. IP adresa je nužna da bi se poslani paketi sa izvornog uređaja mogli preusmjeriti na odredište. Kako je nužno da IP adrese budu jednoznačno dodijeljene, postoju međunarodne organizacije koje se brinu o raspodjeli IP adresa.

Ove IP adrese zovemo još i javnim adresama koje su jedinstvene, globalne i standardizirane. Imamo još i privatne IP adrese koje su se počele koristiti kao rješenje nedostatka slobodnih IP adresa zbog razvoja interneta. Privatne IP adrese mogu biti duplicirane uz uvjet da se ne nalaze na istoj lokalnoj mreži. Prilikom izlaska iz lokalne mreže na internet, privatna adresa se pretvara u javnu adresu.

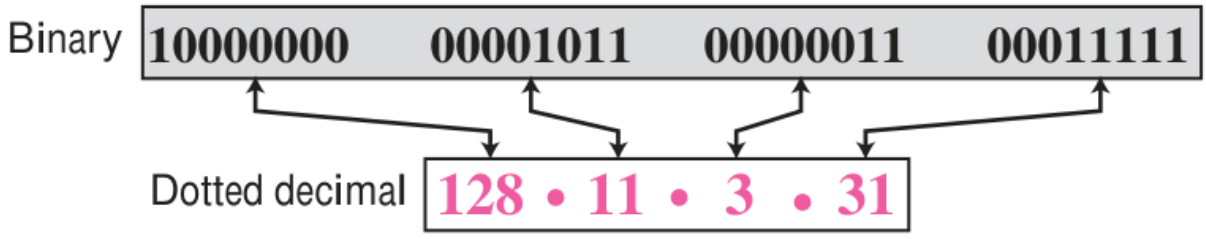
Problem je što je velik broj adresa unutar dodijeljenog bloka ostao neiskorišten jer je svaki korisnik uzimanjem jednog raspona adresa (mrežna klasa) rezervirao veliki broj pojedinačnih IP adresa za svoje buduće potrebe. Ovaj problem se može riješiti na više načina:

- Dijeljenjem adresnog prostora neke klase na manje blokove primjenom mrežnih adresnih klasa
- Ujedinjavanjem susjednih blokova neke klase u jednu veću klasu
- Korištenjem skrivenih podmreža s privatnim adresama
- Novom verzijom internetskog protokola sa adresom dovoljne duljine

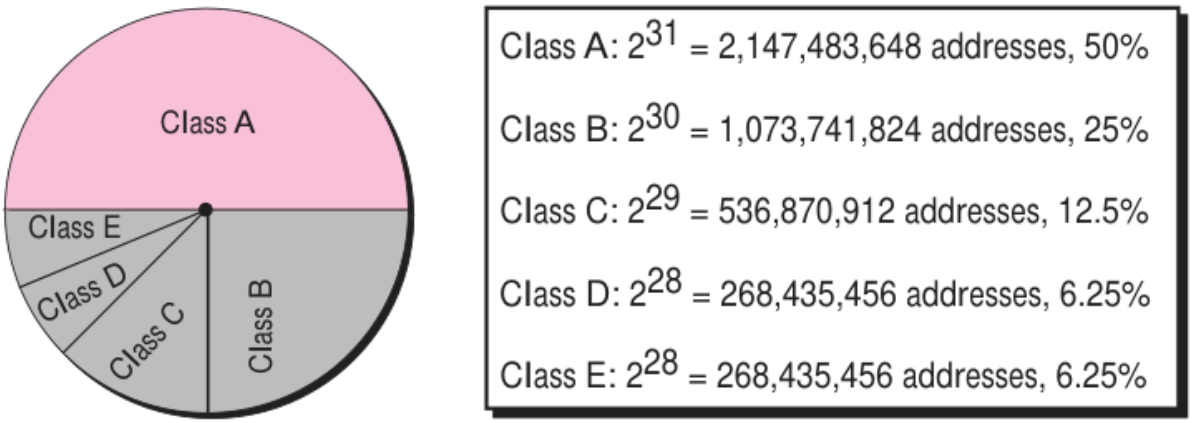
1.5.1. IPv4 adresa

IPv4 adresa je 32-bitna adresa koja jedinstveno i univerzalno definira vezu između izvora ili usmjerivača i interneta. One su jedinstvene u smislu da je svaka IPv4 adresa definirana samo jednu vezu na internet [1].

Sastoji se od 32 bita,tj. 4 okteta ili byta koji se odvaju točkama. Sastoji se od 2 dijela: mrežni broj (eng. *Network Number*) i broja računala (eng. *Host Number*).



Slika 5. Primjer IPv4 adrese u binarnom i decimalnom prikazu (preuzeto iz [2])



Slika 6. Podjela IPv4 adresa po klasama (preuzeto iz [2])

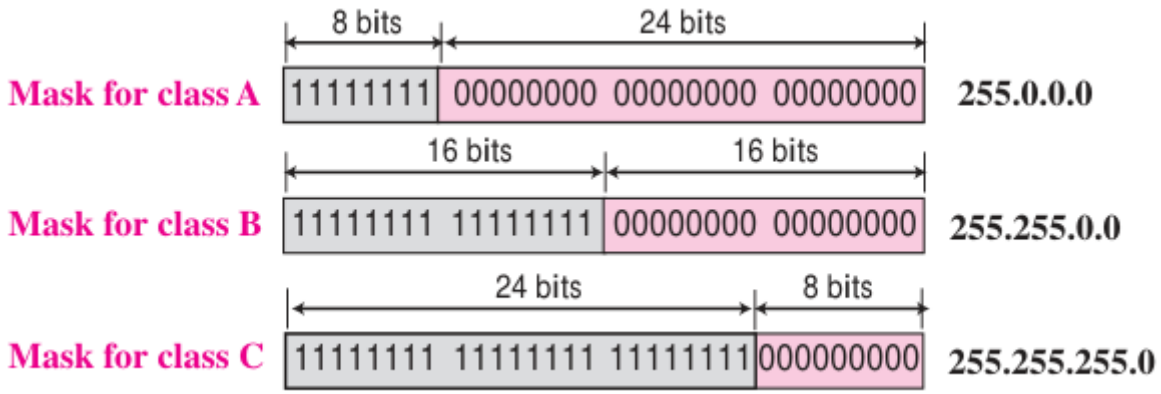
IPv4 adrese su podijeljene u 5 klasa:

- Klasa A – predviđena je za mreže sa velikim brojem uređaja, a može imati do 2^{24} uređaja i njoj može pripadati do 126 mreža. Raspon je od 0.0.0.0 do 127.255.255.255
- Klasa B – predviđena je za 2^{14} mreža koje mogu imati do 65 534 računala. Raspon je od 128.0.0.0 do 191.255.255.255
- Klasa C – predviđena je za 2^{21} mreža koje mogu imati do 254 uređaja. Raspon je 192.0.0.0 do 223.255.255.255
- Klasa D – koristi se za istovremeno pristupanje grupi računala (eng. *Multicast*). Raspon je od 224.0.0.0 do 239.255.255.255
- Klasa E – trenutno se čuva za budućnost, a raspon je od 240.0.0.0 do 255.255.255.255

Neke adrese imaju posebnu namjenu te se ne dodjeljuju određenom računalu na mreži. Pošto ovih adresa je ograničen broj, trebalo je naći nekakvo rješenje, makar privremeno, da proširimo broj adresa i zbog toga su se uvele mrežne maske.

Mrežna maska je 32-bitni broj koji kaže koje bitove originalne IP adrese treba promatrati kao bitove mrežnog broja. Ako je bit mrežne maske postavljen u 1, smatra se da taj bit pripada adresi mreže, a svi ostali bitovi koji su postavljeni u 0 definiraju broj računala. Prema van se mreža još uvijek ponaša kao jedna iako je podijeljena [4].

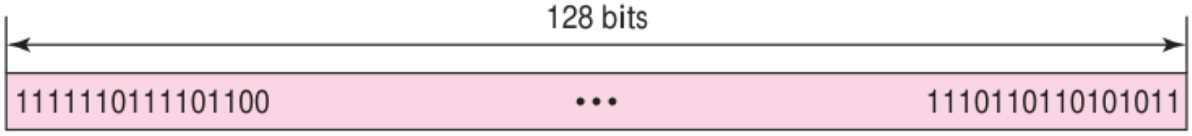
Primjenom mrežne maske omogućeno je formiranje podklasa i podmreža unutar jedne dodijeljene mrežne klase i na taj način se povećava broj mreža na račun broja računala. Uspostavom maske usmjerivačke tablice se mijenjaju jer moraju sadržavati podatke o podmrežama. Usmjerenik u podmreži mora znati kako doći do ostalih podmreža i računala u svojoj podmreži.



Slika 7. Format mrežne maske po klasi (preuzeto iz [2])

1.5.2. IPv6 adrese

IPv6 je nastao zbog nagle expanzije interneta i mogućnosti ponestanka IP adresa. Cijelokupni model IPv6 adresa je sličan kao kod IPv4 adresa, ali sa nadogradnjom. IPv6 adresa se sastoji od 128 bitova [1].



Slika 8. Primjer IPv6 adrese u binarnom zapisu (preuzeto iz [2])

Računalo sprema adrese u binarnom zapisu, ali je očito iz slike 8 da čovjek ne može lako rukovati sa 128 bitova. Da bi se adrese bile čitljive, 128 bitova je podjeljeno u 8 grupa u kojima svaka grupa ima po 2 okteta. Dva okteta u heksadekadskom zapisu trebaju četiri heksadekadske znamenke. Tako se adresa sastoji od sveukupno 32 heksadekadske znamenke. Sve grupe od 4 heksadekadske znamenke su odvojene dvotočkom.

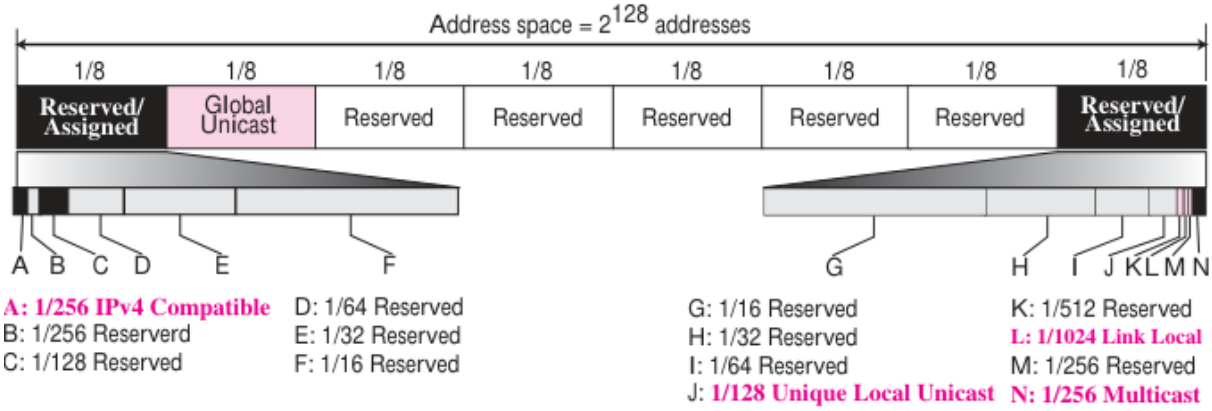
FDEC : BA98 : 7654 : 3210 : ADBF : BBFF : 2922 : FFFF

Slika 9. Primjer IPv6 adrese u heksadekadskom zapisu (preuzeto iz [2])

Sve IPv6 odredišne adrese spadaju u jednu od tri kategorije:

- Jednostruke adrese (eng. *Unicast Address*) – adrese koje definiraju jedno sučelje, obično računalo ili usmjerivač. Paket poslan prema jednostrukoj adresi se preusmjerava prema primatelju.
- *Anycast* adrese – adrese koje definiraju grupu uređaja koji dijelu zajedničku adresu. Paket sa *anycast* adresom se dostavlja samo jednom članu grupe i to onome koji je najpristupačniji. *Anycast* komunikacija se obično koristi kada imamo više servera koji mogu odgovoriti na upit. Zahtjev se šalje onomu koji je najpristupačniji. Hardver i softver generiraju samo jednu kopiju zahtjeva i to kopiju koja dolazi do servera. IPv6 nema zaseban block za *anycast* adrese nego se adrese dodjeluju iz bloka jednostrukih adresa.
- Višestruke adrese (eng. *Multicast Address*) – adrese koje definiraju grupu uređaja, ali za razliku od *anycast-a*, kod višestrukih adresa, svaki član grupe dobije kopiju zahtjeva.

IPv6 adrese su kao i IPv4 adrese podjeljene u nekoliko blokova različitih veličina i svaki blok ima svoju svrhu. Većina blokova je još uvijek bez svrhe i sačuvana za budućnost.

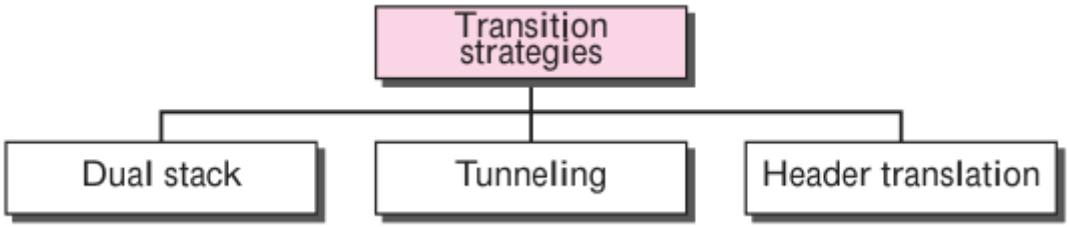


Slika 10. Podjela IPv6 adresa u odjeljke (preuzeto iz [2])

Svaki odjeljak predstavlja 1/8 cijelog adresnog prostora.

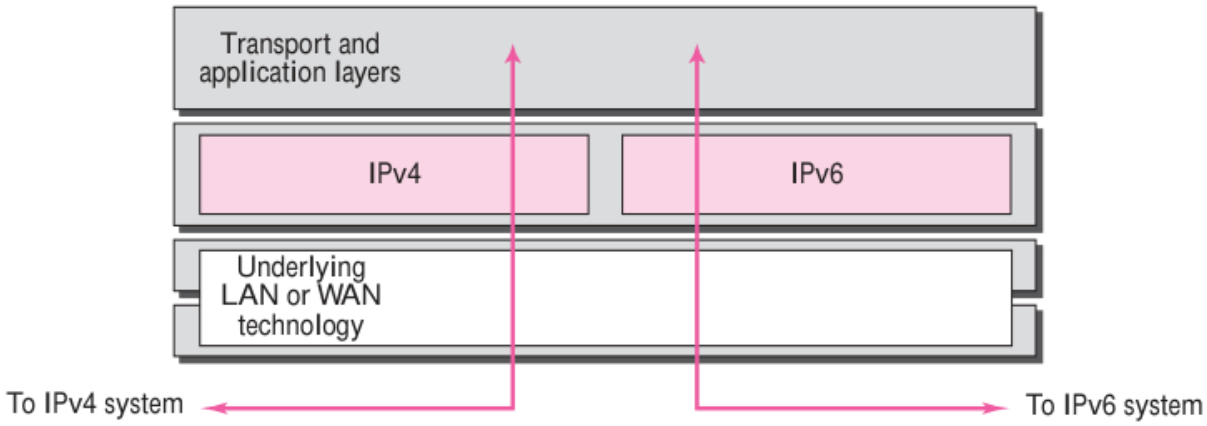
Prvi odjeljak sadrži šest blokova različitih veličina od kojih su prva tri zauzeta, a ostala tri nedodijeljena. Drugi odjeljak je jedan veliki blok koji se koristi za globalne jednostruke adrese koje se koristu za komunikaciju između dva uređaja koja su spojena na Internet. Sljedećih pet odjeljaka su nedodijeljeni. Zadnji odjeljak je podijeljen u osam blokova od kojih su neki nedodijeljeni, a neki su rezervirani za specijalne svrhe. Samo osmina adresnog prostora je korištena za jednostruku komunikaciju između uređaja.

Jedan problem kod korištenja IPv6 adresa je prijelaz sa IPv4 adresa na novi protokol jer se ovaj prijelaz ne može desiti instantno s obzirom na broj uređaja koji koristu stari protokol na internetu. Da bi olakšali ovaj prijelaz, napravljene su strategije za lakši prijelaz na novi protokol.



Slika 11. Strategije za prijelaz na IPv6 protokol (preuzeto iz [2])

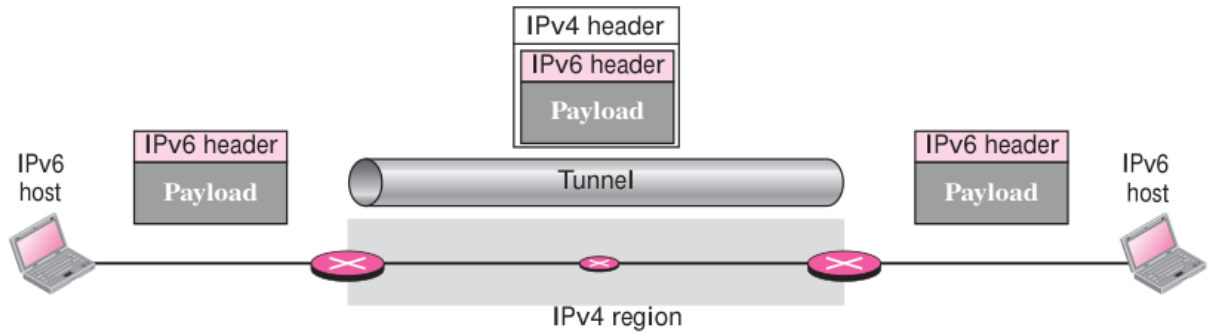
Dvostruki stog (eng. *Dual Stack*) je strategija u kojoj svi uređaji moraju biti kompatibilni sa IPv4 i IPv6 sve dok cijeli internet ne koristi IPv6 protokol i njegove adrese [2].



Slika 12. Dual Stack strategija (preuzeto iz [2])

Da bi se odredila verzija koja se koristi pri slanju paketa, izvorni uređaj postavlja upite u DNS. Ako DNS vrati IPv4 adresu, izvor pošalje IPv4 datagram, a ako DNS vrati IPv6 adresu, izvor pošalje IPv6 datagram.

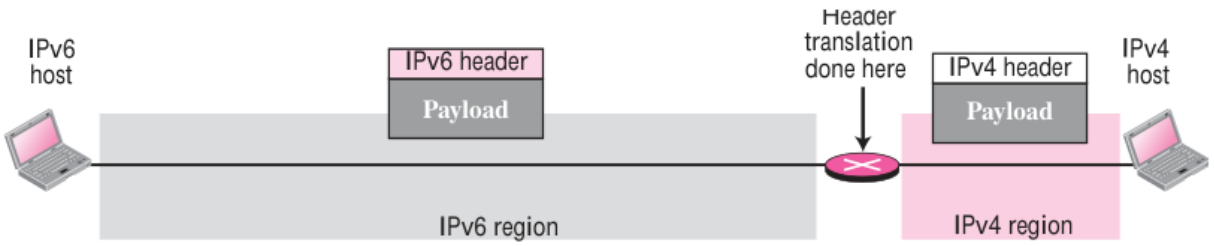
Tuneliranje (eng. *Tunneling*) je strategija korištena kada dva uređaja, koristeći IPv6, žele komunikacijom jedan sa drugim i paket mora proći kroz regiju koja koristi IPv4 [2]. Da taj paket prođe kroz tu regiju, on mora imati IPv4 adresu. Ovo znači da je IPv6 paket enkapsuliran u IPv4 paket kada uđe u regiju i onda izađe iz kapsule kada izađe iz te IPv4 regije. Tako izgleda kao da IPv6 paket prođe kroz tunel na jednom kraju i izađe na drugom kraju.



Slika 13. Strategija tuneliranja

Prevoditelj zaglavlja (eng. *Header Translation*) je strategija koja se primjenuje kada je većina interneta prešla na IPv6, ali ima još sistema koji koristu IPv4 kao npr. kada uređaj želi koristiti IPv6 za poslati nešto, ali primatelj ne razumije IPv6 [2]. Ovdje tuneliranje ne bi radilo jer paketi trebaju biti u IPv4 formatu da bi ih primatelj razumio. Zato se treba cijeli format zaglavlja promijeniti preko prevoditelja zaglavlja. Zaglavlje IPv6 paketa se pretvara u zaglavlje IPv4 paketa. Prevoditelj zaglavlja koristi mapirane adrese da prevede IPv6 adresu u IPv4 adresu. Imaju pravila koja se koristu u pretvaraju zaglavlja IPv6 paketa u zaglavlje IPv4 paketa, a to su:

- Oznaka toka u IPv4 je nula, a kod IPv6 se ignorira
- Kompatibilna dodatna zaglavlja se pretvaraju u opcije i onda se ubacuju u zaglalvje IPv4 s obzirom da se neki odbacuju
- Duljina zaglavlja IPv4 se računa i ubacuje u odgovarajuće polje
- Ukupna duljina IPv4 paketa je računa i ubacuje u odgovarajuće polje



Slika 14. Strategija sa prevoditeljom zaglavlja (preuzeto iz [2])

1.6. ICMP

ICMP (eng. *Internet Control Message Protocol*) je komunikacijski protokol koji je ugrađen u svaki IP modul da bi omogućio mrežnim usmjerivačima ili računalima slanje kontrolnih poruka o greškama [1]. On šalje poruke koje osiguravaju kontrol toka, prijavu pogreške, pojavu alternativnog puta do odredišta i slično. Iako je zadužen za prijavljivanje grešaka, on ih ne ispravlja.

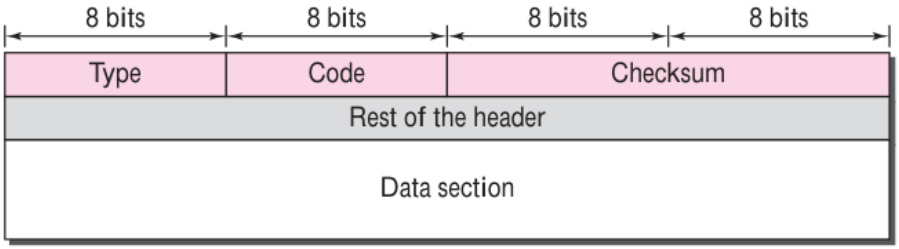
ICMP poruke su podjeljenje u dvije kategorije:

- Poruke koje prijavljuju pogreške (eng. *Error-Reporting Messages*) – ove poruke prijavljuju probleme koje usmjerivači ili izvorni ili odredišni uređaji mogu susresti tijekom obrade paketa.
- Poruke sa upitom (eng. *Query Messages*) – ove poruke, koje dolazu obično u paru, pomažu izvornom uređaju ili upravitelju mreže da dobije specifične informacije od usmjerivača ili drugog izvora.

Category	*Type*	*Message*
Error-reporting messages	3	Destination unreachable
	4	Source quench
	11	Time exceeded
	12	Parameter problem
	5	Redirection
Query messages	8 or 0	Echo request or reply
	13 or 14	Timestamp request or reply

Slika 15. Tipovi ICMP poruka (preuzeto iz [2])

ICMP poruka ima 8-bytno zaglavlje i odjedljak podataka (eng. *Data Section*) koji je varijabilne veličine. Prvo polje, koje se još zove tip (eng. *Type*) nam govori tipu poruke. Polje kod (eng. *Code*) specificira razlog za baš taj tip poruke, a zadnje polje, koje se još zove kontrolni zbroj (eng. *Checksum*), služi za provjeru cjelovitosti poruke, tj. da bi se osiguralo da primljena poruka nije oštećena prilikom prijenosa.



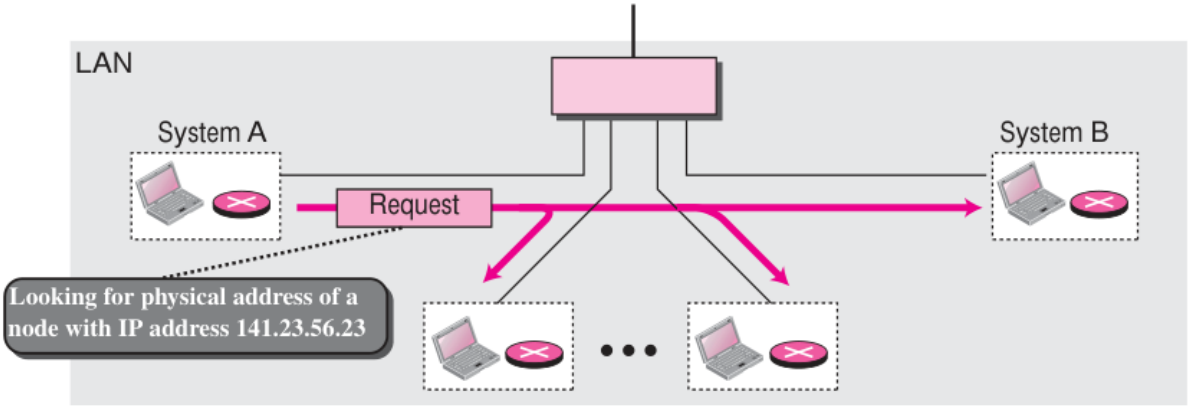
Slika 16. Općeniti format ICMP poruka (preuzeto iz [2])

1.7. ARP

ARP (eng. *Address Resolution Protocol*) je komunikacijski protokol kojim se dobiva fizička adresa na lokalnoj mreži iz poznate mrežne adrese [1]. Na svakoj fizičkoj mreži su računala identificirana svojom mrežnom adresom. Protokoli na višim slojevima za adresiranje računala koriste IP adrese. Kada protokol na višem sloju želi poslati datagram na odredišnu IP adresu, mrežno sučelje ne prepoznaje tu adresu.

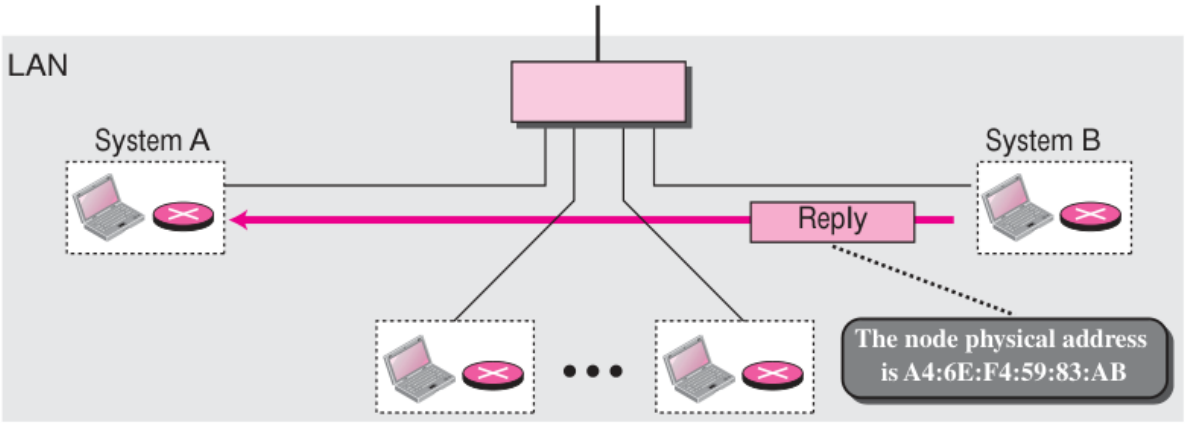
ARP obavlja prevođenje IP adrese u fizičku adresu odredišnog računala. Za to se koristu tablice koje se obično nazivaju ARP keš (eng. *ARP Cache*).

Ako se adresa ne nalazi u tablici, šalje se broadcast zahtjev na mrežu posebnog oblika koji se naziva ARP zahtjev (eng. *ARP Request*).

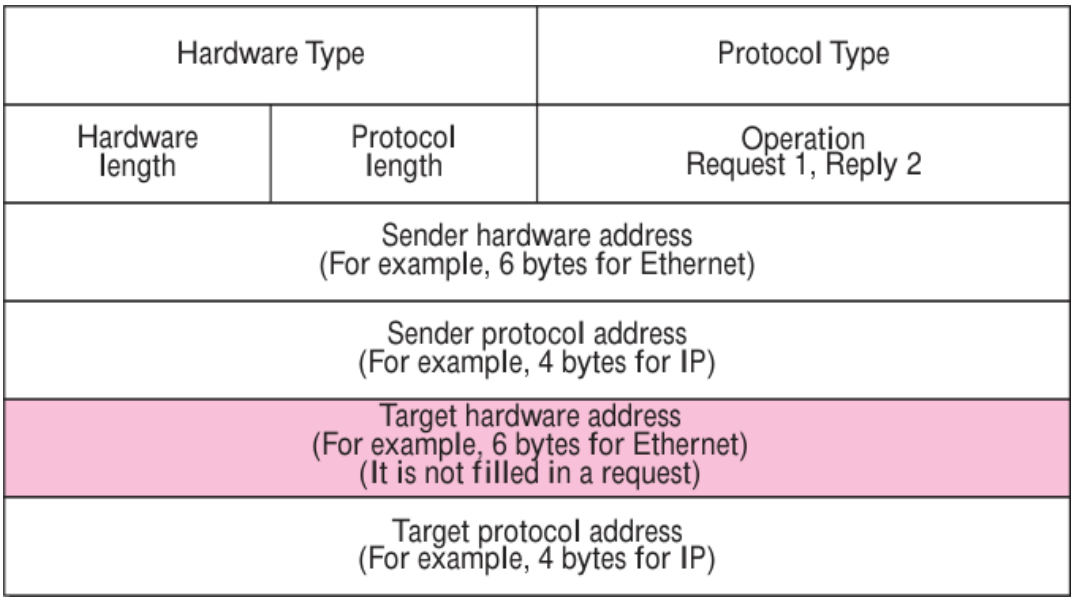


Slika 17. Izgled ARP zahtjeva koja je multicast (preuzeto iz [2])

Ako neki uređaj na mreži prepozna svoju IP adresu, taj uređaj šalje ARP odziv računalu koje je poslalo taj zahtjev. Taj odziv sadrži fizičku adresu, koje će se sad spremiti u *ARP Cache* tablicu, i svi sljedeći paketi za to računalo mogu sada koristiti poznatu fizičku adresu.



Slika 18. Izgled ARP odziva koji je unicast (preuzeto iz [2])



Slika 19. Izgled ARP paketa (preuzeto iz [2])

ARP paket se sastoji od sljedećih polja:

- Tip hardvera (eng. *Hardware Type*) – 16 bitno polje koje definira tip mreže na kojoj je ARP pokrenut. Svakoj lokalnoj mreži (eng. *LAN – Local Area Network*) se dodijeljuje broj ovisno o tipu mreže. Na primjer, Ethernetu je dan tip 1.
- Tip protokola (eng. *Protocol Type*) – 16 bitno polje koje definira protokol.
- Duljina hardvera (eng. *Hardware Length*) – 8 bitno polje koje definira duljinu fizičke adrese u bytovima. Npr., kod Ethernet-a, vrijednostje 6.
- Duljina protokola (eng. *Protocol Length*) – 8 bitno polje koje definira duljinu logičke adrese u byteovima. Npr., kod IPv4 protokola, vrijednost je 4.
- Operacija (eng. *Operation*) – 16 bitno polje koje definira tip paketa. Može imati dva tipa paketa: ARP zahtjev i ARP upit.
- Pošiljateljeva fizička adresa (eng. *Sender Hardware Address*) – polje varijabilne duljine koje definira fizičku adresu pošaljitelja.
- Pošiljateljeva protokol adresa (eng. *Sender Protocol Address*) – polje varijabilne duljine koje definira logičku adresu pošaljitelja.
- Ciljna fizička adresa (eng. *Target Hardware Address*) – polje varijabilne duljine koje definira fizičku adresu odredišta. Kod ARP zathjeva, ovo polje je puno nula jer pošiljatelj ne zna fizičku adresu odredišta
- Ciljna protokol adresa (eng. *Target Protocol Address*) – polje varijabilne duljine koje definira logičku adresu odredišta.

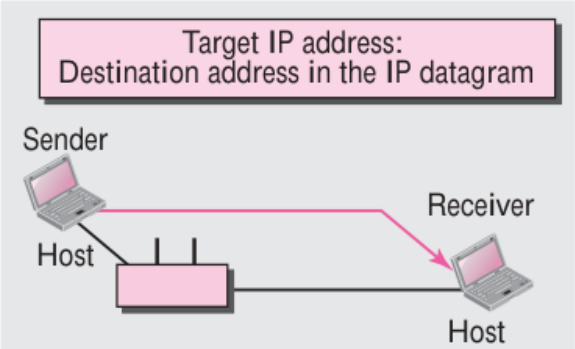
U ARP procesu imamo sedam koraka koji opisuju funkcije ARP-a na tipičnom internetu:

1. Pošaljitelj zna IP adresu odredišta.
2. Internetski protkol pita ARP da napravi ARP zahtjev u kojoj se nalazi pošiljateljeva fizička adresa, pošiljateljeva IP adresa i IP adresa odredišta. Polje sa pošaljiteljevom fizičkom adresom je napunjeno s nulama.
3. Poruka se šalje u sloj podatkovne veze, o kojem ćemo pričati kasnije, gdje se enkapsulira u okvir (eng. *Frame*) koristeći fizičku adresu pošiljatelja kao izvornu adresu i fizičku „*broadcast*“ adresu kao odredišnu adresu.
4. Svaki izvorni uređaj ili ruter dobiva okvir. Zato jer okvir sadrži *broadcast* adresu kao odredište, svi uređaji osim onog odredišnog odbacu paket jer će odredišni uređaj prepoznati IP adresu.
5. Odredišni uređaj odgovara sa ARP odgovorom koji sadrži njegovu fizičku adresu. Ovaj ARP zahtjev je *unicast*.
6. Pošiljatelj primi ARP odgovor i sada zna fizičku adresu odredišnog uređaja.
7. Paket, koji sadrži podatke za odredišni uređaj, je sad enkapsuliran u okvir i pomoću *unicast*-a se prenosi do odredišta.

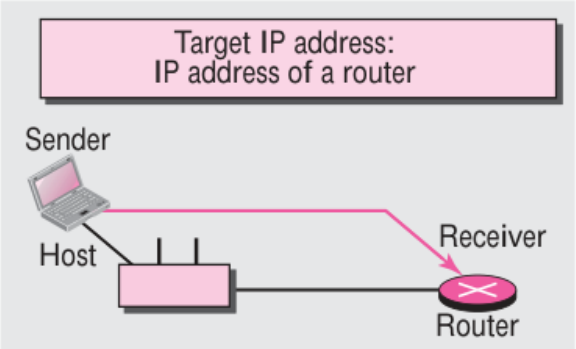
Imamo četiri različita slučaja za korištenje ARP-a:

- 1. Slučaj – Pošiljatelj je izvor koji želi poslati pakete drugom izvoru na istoj mreži. U ovom slučaju logička adresa koja se treba mapirati u fizičku adresu je ujedino i odredišna IP adresa u zaglavlju paketa.
- 2. Slučaj – Pošiljatelj je izvor koji želi poslati pakete drugom izvoru koji se nalazi na drugoj mreži. U ovom slučaju izvor pošiljatelj gleda u tablicu usmjeravanja i traži IP adresu od usmjerivača za odredište. Ako ju nema u tablici usmjeravanja, onda traži za IP adresu zadanog usmjerivača. IP adresa usmjerivača postaje logička adresa koja se mora mapirati u fizičku adresu.
- 3. Slučaj – Pošiljatelj je usmjerivač koji je primio paket koji predodređen izvoru na drugoj mreži. Provjerava tablicu usmjeravanja i traži IP adresu sljedećeg usmjerivača. IP adresa sljedećeg usmjerivača postaje logička adresa koja se mapira kao fizička adresa.
- 4. Slučaj – Pošiljatelj je usmjerivač koji je primio paket koji je predodređen izvoru na istoj mreži. Odredišna IP adresa paketa postaje logička adresa koja se treba mapirati kao fizička adresa.

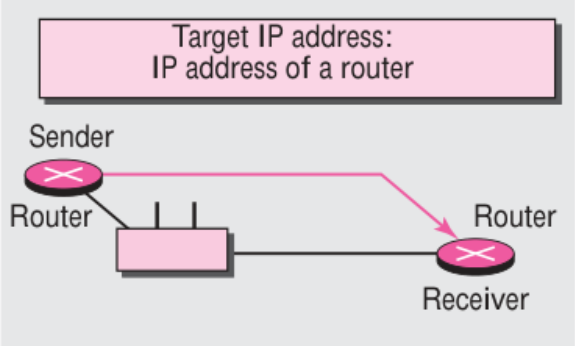
4. Slučaj



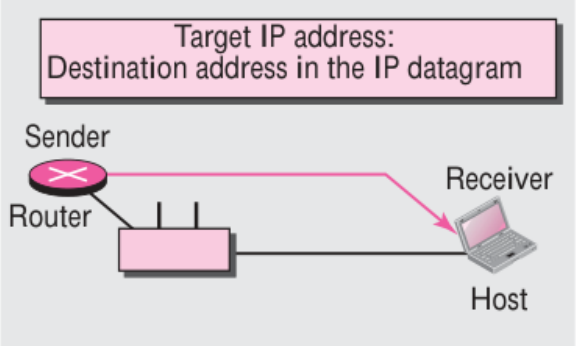
3. Slučaj



2. Slučaj



1. Slučaj



Slika 20. Svi slučajevi za korištenje ARP-a (preuzeto iz [2])

2. OSI model i TCP/IP paket protokola

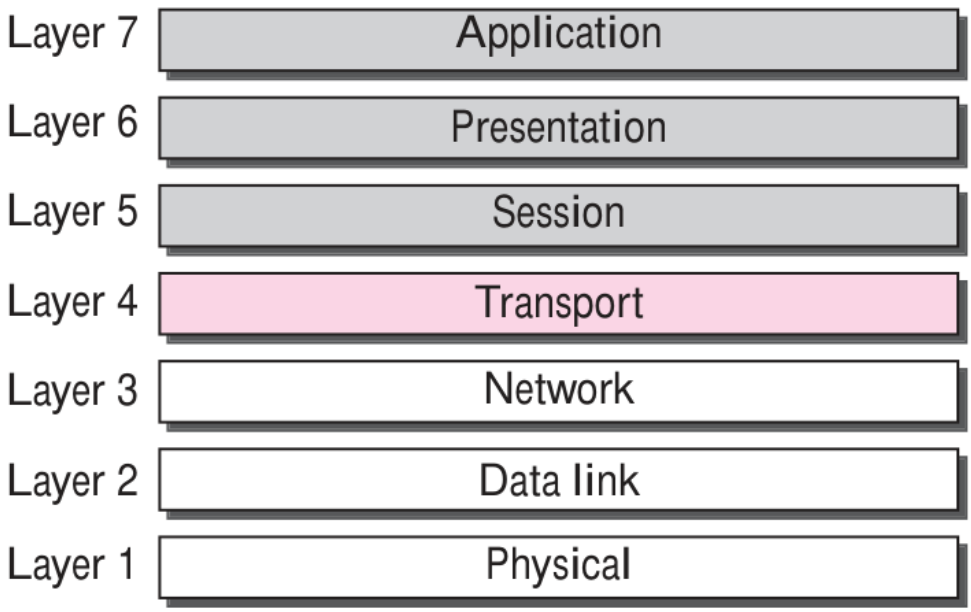
U prošlom poglavlju smo govorili o općenitom pojmu internetskog protokola te osnove kako internetski protokol radi i što sve koristi da bi uopće radio. Kada komunikacija između uređaja nije jednostavna, onda trebamo podijeliti kompleksne zadatke komunikacije na više slojeva.

U ovom poglavlju ćemo pokazati sve slojeve koji se koristu u komunikaciji na primjeru 2 najčešće korištena modela ili pakete protokola, a to su: OSI model i TCP/IP paket protokola. Prvo ćemo objasnit OSI model i njegove slojeve i onda se prebaciti na TCP/IP paket protokola i dva protokola koja ćemo koristiti u praktičnom dijelu ovog završnog rada, a to su: TCP i UDP.

2.1. OSI model

OSI model (eng. *Open Systems Interconnection*) je ISO standard koji pokriva sve aspekte internetske komunikacije [5]. OSI model predstavlja set protokola koji dozvoljavaju da bilo koja dva različita uređaja komuniciraju bez obzira na njihovu arhitekturu. Zadatak OSI modela je pokazati kako olakšati komunikaciju između različitih uređaja, a da ne moramo mijenjati logiku hardvera i softvera uređaja. OSI model je zamišljen da bude osnovna tvorevina u OSI stogu.

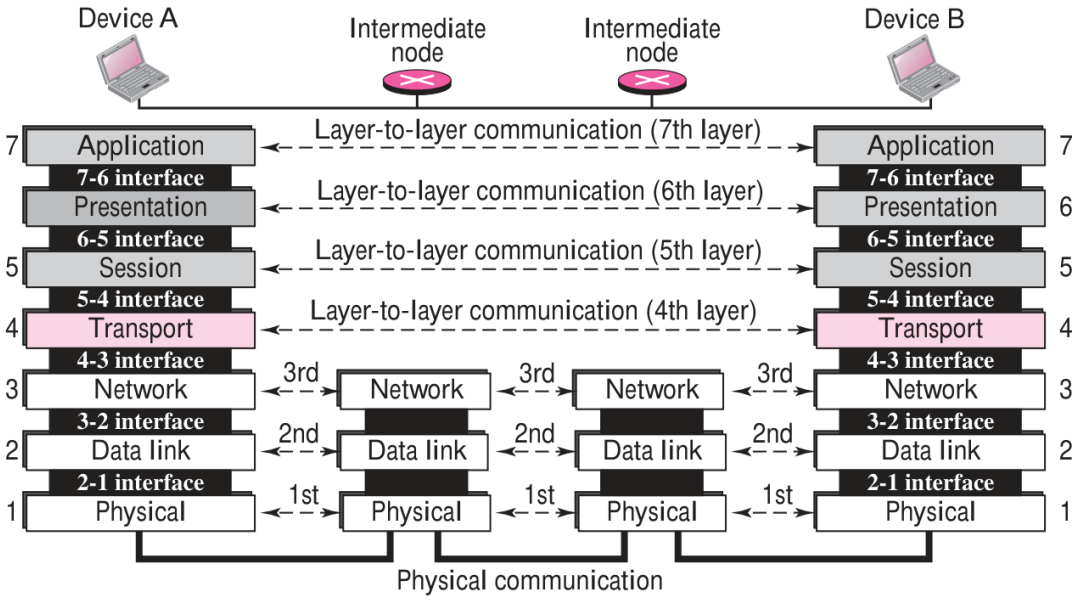
OSI model je slojeviti okvir (eng. *Layered Framework*) za dizajniranje mrežnih sustava koji dozvoljavaju komunikaciju između svih tipova računalnih sustava. Sadrži sedam različitih, ali povezanih slojeva, od kojih svaki predstavlja dio procesa koji prenosi informacije preko mreže.



Slika 21. OSI Model (preuzeto iz [2])

OSI model se sastoji od sedam poderanih slojeva:

1. Fizički sloj (eng. *Physical Layer*)
2. Podatkovni sloj (eng. *Data Link Layer*)
3. Mrežni sloj (eng. *Network Layer*)
4. Prijenosni sloj (eng. *Transport Layer*)
5. Sjednički sloj (eng. *Session Layer*)
6. Predodžbeni sloj (eng. *Presentation Layer*)
7. Aplikacijski sloj (eng. *Application Layer*)



Slika 22. Slojevi OSI modela (preuzeto iz [2])

Slika 22. prikazuje slojeve OSI modela i kako su oni uključeni u procesu prijenosa informacije koja je poslana od uređaja A do uređaja B. Na slici vidimo kako uređaj A šalje poruku uređaju B. Na mjestu slanja, od uređaja A, poruka se šalje kroz slojeve, od sloja 7 do sloja 1. Na fizičkom sloju cijeli paket se pretvara u formu koja se može poslati na odredište. Na odredištu se slična stvar događa kao i na izvoru, samo obrnuti redoslijed. Ovdje poruka, tj. paket ide od sloja 1 do sloja 7. Ovaj prijelaz podataka i informacija kroz slojeve izvornog uređaja i obrnutim redoslijedom na odredištu je moguće zbog sučelja (eng. *Interface*) između svakog para susjednih slojeva. Svako sučelje definira koju informaciju i servis sloj mora pružiti za sloj koji je iznad. Dobro profilirana sučelja i funckije slojeva pružaju modularnost mreži. Sve dok sloj pruža očekivane funkcije sloju iznad, specifična implementacija tih funkcija se može mijenjati ili u potpunosti zamijeniti bez potrebe za promjenama okolnih slojeva.

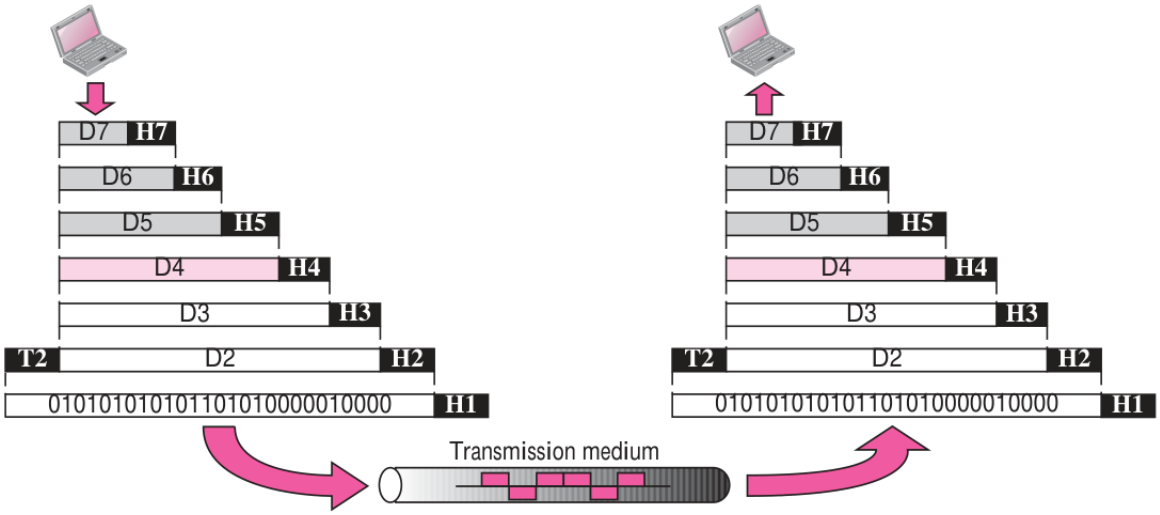
Ovih sedam slojeva možemo podijeliti u tri podgrupe.

Fizički, podatkovni i mrežni slojevi spadaju pod jednu podgrupu koja predstavlja mrežne potporne slojeve (eng. *Network Support Layers*) i oni se bave sa fizičkim aspektima prijenosa podataka sa jednog na drugi uređaj poput električnih specifikacija, fizičkih veza, fizičkog adresiranja, vrijeme transporta i pouzdanost.

Sjednički, predodžbeni i aplikacijski slojevi spadaju u drugu podgrupu koja predstavlja slojeve korisničke podrške (eng. *User Support Layers*) koji omogućavaju razmjenu informacija između računalnih sustava ili softvera.

Prijenosni sloj spada u svojoj podgrupi koja zapravo spaja ostale dvije podgrupe i osigurava da sve što su mrežni potporni slojevi prenjeli je u formi koju slojevi korisničke podrške mogu razumijeti i iskoristiti.

Slojevi korisničke podrške su skoro uvijek implementirani u softveru, dok su mrežni potporni slojevi implementirani u kombinaciji hardvera i softvera, osim fizičkog sloja koji je većinom implementiran u hardveru.



Slika 23. Razmjena podataka korištenjem OSI modela (preuzeto iz [2])

Na slici 23. koja nam daje ukupni pogled na slojeve OSI modela, sa slovom D su označeni podaci, a brojevi uz to slovo nam govoru na kojem se sloju trenutno nalazi. Tako npr., D7 znači da je podatak na aplikacijskom sloju, D6 da je podatak na predodžbenom sloju i tako za ostale. Kod svakog sloja imamo zaglavlje (eng. *Header*) koje se može dodati podatku, a na podatkovnom sloju možemo još dodati i „podnožje“ (eng. *Trailer*). Kada se formatirani podatak pošalje kroz fizički sloj, on se pretvara u elektromagnetski signal i prenosi preko fizičke veze do odredišta. Kada paket dođe do odredišta, signal prolazi kroz fizički sloj odredišta i pretvara se natrag u digitalnu formu. Podaci se prenosu kroz slojeve. Kako svaki blok podataka stigne do sljedećeg, višeg sloja, zaglavlje i podnožje koji su u prilogu sa cijelim paketom se u odgovarajućem sloju otklanja i onda se radu funkcije koje odgovaraju tom sloju. Kada dođe do zadnjeg, aplikacijskog sloja, poruka je opet u formi koja je prikladna za aplikacijski sloj i ona se prikazuje primatelju.

Ovo dodavanje zaglavlja i podnožja podatku se zove enkapsulacija. Paket na aplikacijskom sloju se enkapsulira u paket na predodžbenom sloju i tako za svaki sljedeći sloj. Enkapsulacija omogućava slanje podataka preko mreže i osigurava da ih primatelj može pravilno obraditi.

Drugim riječima, podatkovni dio paketa na sloju N nosi cijeli paket od sloja $N + 1$. Ovo se zove enkapsulacija jer sloj N nije svjestan koji dio enkapsuliranog paketa je podatak, a koji zaglavlje ili podnožje. Za sloj N, cijeli paket koji dolazi od levela $N + 1$ se tretira kao jedna sastavna jedinica.

2.1.1. Slojevi OSI modela

Fizički sloj

Fizički sloj koordinira funkcije koje su potrebne da se nosi tok bitova preko fizičkog medija [2]. Bavi se sa mehaničkim i električnim specifikacijama sučelja i prijenosnog medija. Usput još definira procedure i funkcije koje fizički uređaj i sučelje trebaju odraditi da bi se prijenos dogodio.

Fizički sloj se isto bavi sa:

- Fizičke karakteristike sučelja i medija – Fizički sloj definira karakteristike sučelja između uređaja i prijenosnog medija i definira tip prijenosnog medija.
- Reprezentacija bitova – Podaci fizičkog sloja se sastoji od toka bitova bez prekida. Da bi se prenjeli, bitovi se moraju kodirati u električne ili optičke signale. Fizički sloj definira tip kodiranja, tj. kako nule i jedinice se mijenjaju u signale.
- Brzina prijenosa – Broj bitova koji se šalje svakuu sekundu je određen fizičkim slojem. Drugim riječima, fizički sloj definira duljinu bita, koja govori koliko će trajati.
- Konfiguracija linije – Fizički sloj se bavi vezom uređaja. U konfiguraciji od točke do točke (eng. *Point-To-Point Configuration*), dva uređaja su spojena zajedno preko posebne veze, a kod multipoint konfiguracije (eng. *Multipoint Configuration*), veza je dijeljena između više uređaja
- Fizička topologija – Definira kako su uređaji spojeni da stvaraju mrežu. Uređaji se mogu povezati korištenjem mrežaste topologije (eng. *Mesh Topology*) u kojoj je svaki uređaj spojen sa svakim drugim uređajem, zvjezdasta topologija (eng. *Star Topology*) u kojoj su uređaji spojeni preko centralnog uređaja, prstenasta topologija (eng. *Ring Topology*) gdje se svaki uređaj spaja sa sljedećim i tako formiraju prsten ili preko autobusne topologije (eng. *Bus Topology*) gdje je svaki uređaj povezan preko zajedničke poveznice.
- Prijenosni čvor – Fizički sloj definira prijenos između dva uređaja. Imamo jednostavni način rada (eng. *Simplex Node*) gdje samo jedan uređaj može slati, a drugi može samo primati podatke, jednosmjerni način rada (eng. *Half-Duplex Node*) gdje dva uređaja mogu slati i primati podatke, ali ne u isto vrijeme. Na kraju imamo dvosmjerni način rada (eng. *Full-Duplex mode*) u kojem dva uređaja mogu slati i primati podatke u isto vrijeme.

Podatkovni sloj

Podatkovni sloj pretvara fizički sloj u pouzdanu vezu [2]. To znači da napravi da fizički sloj izgleda bez grešaka drugim, višim slojevima. Ostali zadaci podatkovnog sloja su:

- Framiranje (eng. *Framing*) – Podatkovni sloj dijeli tok bitova dobivenih od mrežnog sloja u upravljive podatkovne jedince koje se zovu okviri (eng. *Frames*).
- Fizičko adresiranje (eng. *Physical Addressing*) – Ako su okviri distributirani različitim uređajima na mreži, podatkovni sloj dodaje zaglavlje okviru koji definira pošaljitelja ili primatelja tog okvira. Ako je okvir namijenjen za sistem izvan pošaljiteljeve mreže, primateljeva adresa je adresa povezanog uređaja koji povezuje mrežu do sljedeće mreže.
- Kontrola protoka (eng. *Flow Control*) – Ako je stopa kojom primatelj prima podatke manja od stope kojom se podaci šalju od pošaljitelja, podatkovni sloj će nametnuti mehanizma kontrole protoka koji će spriječiti da se primatelj zatrpa podacima.
- Kontrola grešaka (eng. *Error Control*) – Podatkovni sloj dodaje pouzdanost fizičkom sloju tako da doda mehanizme koji detektiraju i reemitiraju izgubljene i oštećene okvire. Usput koristi mehanizam da prepozna duplicirane okvire. Kontrola greške se obično postiže preko podnožja koje se doda na kraj okvira.
- Kontrola pristupa (eng. *Access Control*) – Kada dva ili više uređaja su spojena na istu vezu, protokoli podatkovnog sloja su potrebni da odredu koji uređaj ima kontrolu veze u svakom trenutku.

Mrežni sloj

Mrežni sloj je zadužen za dostavu paketa od izvora do odredišta, čak i preko više mreža [2]. Gdje podatkovni sloj nadzire dostavu paketa između dva sustava na istoj mreži, mrežni sloj osigurava da svaki paket stigne od izvora do odredišta. Ako dva sustava su povezana na istoj mreži, tada nema potrebe za mrežni sloj. Mrežni sloj ima zadatke:

- Logičko adresiranje (eng. *Logical Addressing*) – Ako paket prođe kroz mrežnu granicu, treebamo drugi adresni sistem koji će pomoći razlikovati izvorni i odredišni sustav. Mrežni sloj dodaje zaglavlje paketu koje dolazi od višeg sloja koje uključuje logičku adresu pošaljitelja i primatelja.
- Usmjeravanje (eng. *Routing*) – Kada su nezavisne mreže zajedno spojene da kreiraju međumreže (eng. *Internetworks*) ili veće mreže, usmjerivači usmjeravaju pakete do odredišta, a ovaj mehanizam omogućava jedna od funkcija mrežnog sloja.

Prijenosni sloj

Prijenosni sloj je odgovoran za isporuku od procesa do procesa (eng. *Process-To-Process Delivery*) cijele poruke [2]. Proces je program koji se trenutno izvršava. Dok mrežni sloj nadzire isporuku od izvora do odredišta pojedinih paketa, on ne prepoznaje nikakav odnos između tih paketa. On tretira svaki odvojeno kao da svaki dio spada zasebnoj poruci. Prijenosni sloj osigurava da cijela poruka dospije do odredišta čitava i u ispravnom rasporedu, pritom nadzireći kontrolu grešaka i kontrolu protoka na takozvanom izvor do odredišta levelu (eng. *Source-To-Destination Level*). Ovaj sloj se još bavi sa:

- Adresiranje na razini točke usluge (eng. *Service-Point Addressing*) – Uređaji općenito izvršavaju više programa u isto vrijeme. Zbog ovoga, isporuka od izvora do odredišta znači isporuka ne samo od jednog uređaja do sljedećeg uređaja, ali od specifičnog procesa na jednom uređaju do specifičnog procesa na drugom. Zaglavlje prijenosnog sloja zato mora sadržavati tip adrese koji se još zove port adrese. Mrežni sloj dostavlja svaki paket do točnog uređaja, a prijenosni sloj cijelu poruku do točnog procesa na tom uređaju.
- Segmentacija i ponovno sastavljanje (eng. *Segmentation and reassembly*) – Poruka se dijeli u prenesive segmente, od kojih svaki segment sadrži slijedni broj. Ovi broji omogućavaju da prijenosni sloj točno sastavi poruku prilikom dolaska do odredišta i da identificira i zamijeni pakete koji su se izgubili tijekom prijenosa.
- Kontrola veze (eng. *Connection Control*) – Prijenosni sloj može biti ili bez veze (eng. *Connectionless*) ili orijentiran na vezu (eng. *Connection-Orientend*). Kod prijenosa bez veze, svaki segment kao nezavisni paket i dostavlja ih prijenosnom sloju na odredištu. Kod prijenosa orijentiranog na vezu, prijenosni sloj prvo uspostavi vezu sa prijenosnim slojem na odredištu prije nego što pošalje pakete i onda nakon prekine tu vezu.
- Kontrola protoka (eng. *Flow Control*) – Kao i kod podatkovnog sloja, prijenosni sloj je zadužen za kontrol protoka, ali kontrola protoka kod ovog sloja se izvršava od kraja do kraja (eng. *End To End*).
- Kontrola grešaka (eng. *Error Control*) – Slično kao i kod podatkovnog sloja, ali ovdje se kontrola grešaka izvršava od procesa do procesa. Pošaljiteljev prijenosni sloj osigurava da će poruka doći na primateljev prijenosni sloj bez grešaka, a ako bude bilo grešaka, onda će se ponovno poslati sve dok ne bude bilo grešaka.

Sjednički sloj

Sjednički sloj predstavlja NDC (eng. *Network Dialog Controller*). Ovaj sloj uspostavlja, održava i sinkronizira interakciju između sustava koji komuniciraju [2]. On se još bavi:

- Kontrola dijaloga (eng. *Dialog Control*) – Sjednički sloj omogućava da dva sustava uđu u dijalog i omogućava komunikaciju između dva procesa da bude ili jednosmjerno ili dvosmjerno.
- Sinkronizacija (eng. *Synchronization*) – Pomoću sinkronizacije usklađujemo komunikaciju između procesa na različitim uređajima. Sjednički sloj omogućava procesu da doda točke sinkronizacije (eng. *Synchronization Points*) u tok podataka pomoću kojih se provodi sama sinkronizacija.

Predodžbeni sloj

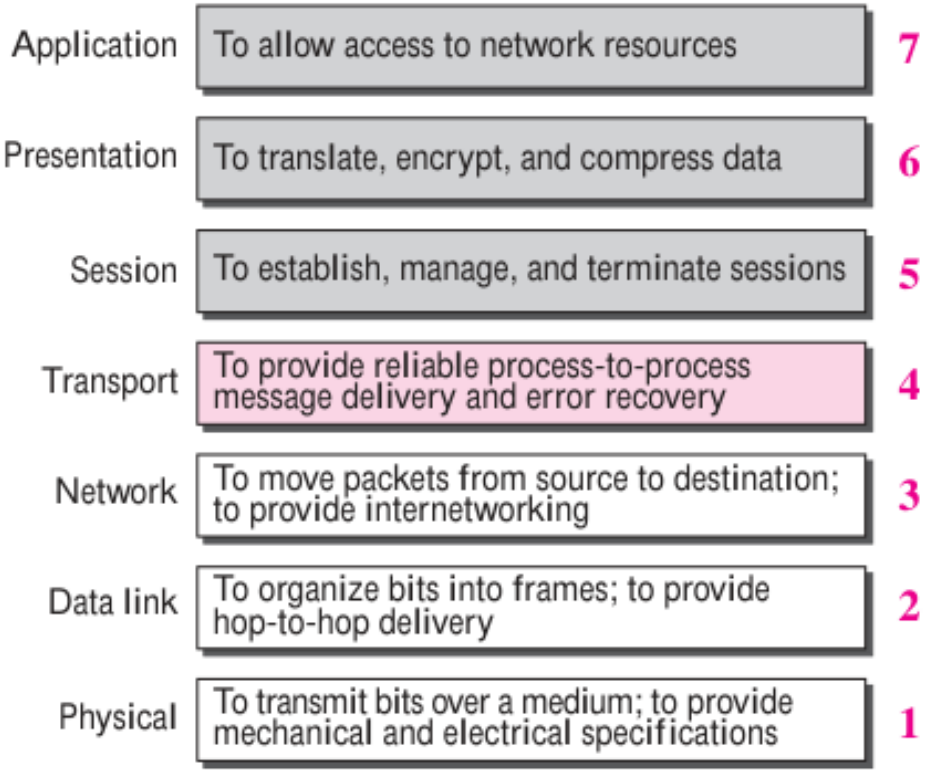
Predodžbeni sloj se bavi sa sintaksom i semantikom informacija razmijenjenih između dva sistema [2]. On se još bavi:

- Prijevodom (eng. *Translation*) – Procesi u dva sustava obično razmjenjuju informacije kao stringove, znakove, brojeve i slično. Informacije se trebaju promijeniti u tokove bitova prije nego što se prenesu. Zbog toga što različiti uređaji koristu različite sisteme kodiranja, predodžbeni sloj je zadužen za prijevod ovih kodiranja u neki zajednički format.
- Enkripcija (eng. *Encryption*) – Enkripcijom se originalna informacija pretvara u drugu formu i šalje se preko mreže, a dekripcijom se informacija pretvara u originalnu formu.
- Kompresija (eng. *Compression*) – Kompresijom se smanjuje broj bitova sadržanih u informaciji i tako trebamo slati manje bitova.

Aplikacijski sloj

Aplikacijski sloj omogućuje korisniku da pristupi mreži. Omogućava korisničko sučelje i podršku za servise kao elektronička pošta, udaljeni prijenos datoteka i ostale tipove distribucijskih informacijskih servisa [2]. Od specifičnih servisa aplikacijskog sloja imamo:

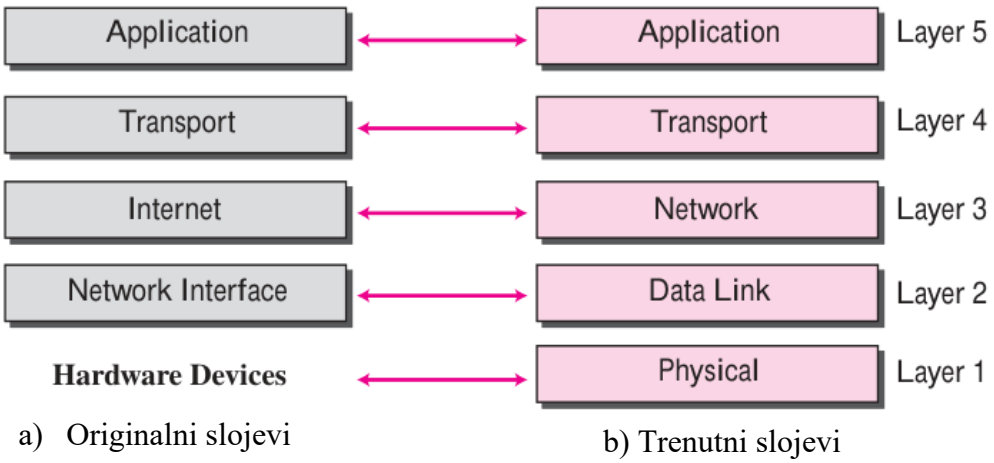
- Mrežni virtualni terminal (eng. *Network Virtual Terminal*) – Ovo je softverska verzija fizičkog terminala koja omogućava korisniku da se prijavi na udaljeni uređaj. Da bi se ovo moglo napraviti, aplikacija kreira softversku emulaciju terminala na udaljenom uređaju i korisnički uređaj priča sa tim softverskim terminalom koji priča sa udaljenim uređajom i obrnuto. Udaljeni uređaj tako misli da priča sa svojim terminalom i tako dopušta korisniku da se prijavi na njega i da ga koristi.
- FTAM (eng. *File Transfer, Access and Management*) – Ova aplikacija omogućava korisniku prijenos i dohvat datoteka na udaljenom uređaju, te kontrolu datoteka koje se nalazu na udaljenom uređaju.
- Servisi elektroničke pošte (eng. *E-mail services*) – Ova aplikacija pruža osnovu prosljeđivanja elektroničke pošte i njeno spremanje.



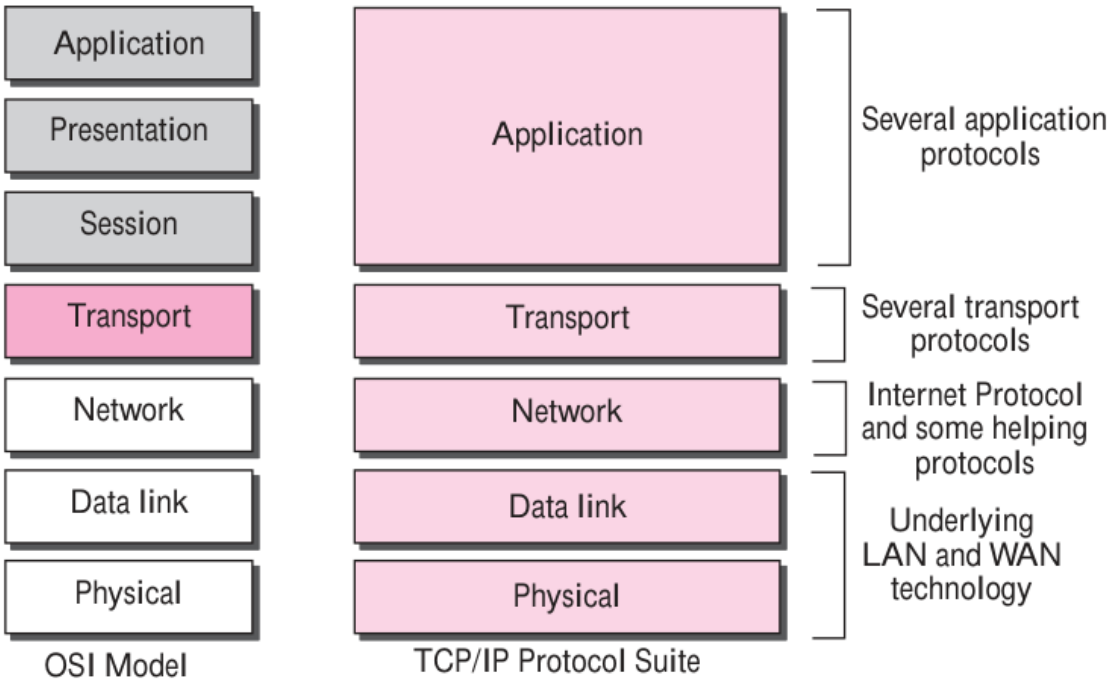
Slika 24. Kratki opis slojeva OSI modela (preuzeto iz [2])

2.1. TCP/IP paket protokola

TCP/IP paket protokola (eng. *TCP/IP Protocol Suite*) je skup protokola koji čine temelj za komunikaciju u mrežama temeljenim na internetu [1]. On je pravljen prije OSI modela i zbog toga se slojevi ne poduduraju točno sa onima u OSI modelu. Originalni TCP/IP paket protokola je bio zamišljen kao četiri softverska sloja izgrađena na hardveru. Danas, TCP/IP paket protokola je zamišljen kao model sa pet slojeva sa slojevima slično imenovanim kao oni u OSI modelu.



Slika 25. Slojevi TCP/IP paketa protokola (preuzeto iz [2])



Slika 26. Usporedba TCP/IP paketa protokola i OSI modela (preuzeto iz [2])

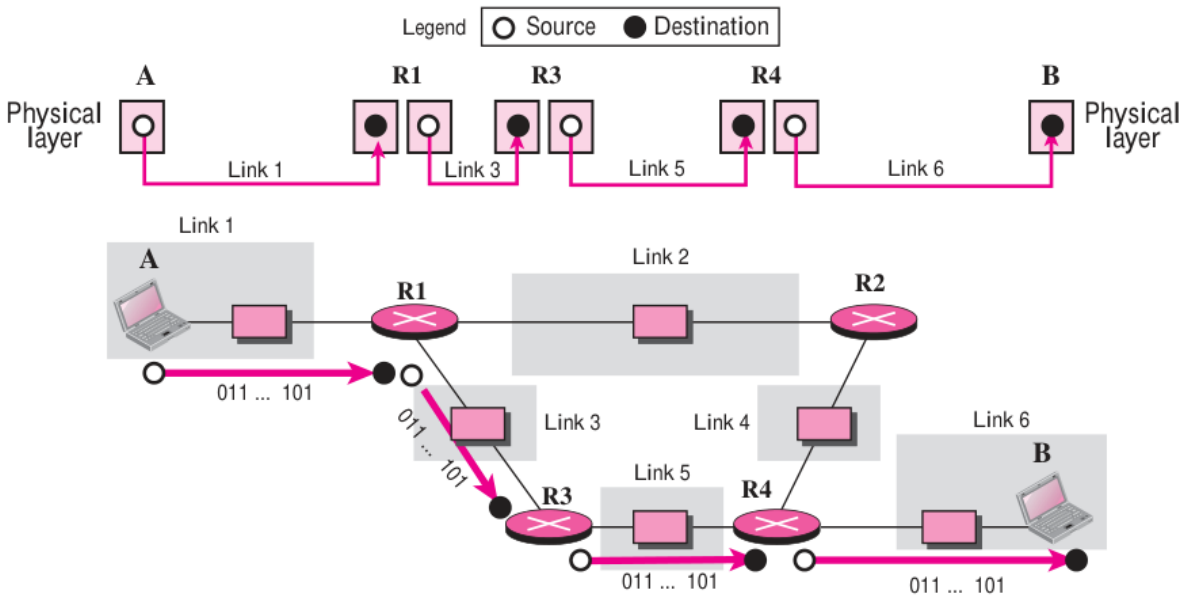
Kada usporedimo ova dva modela, vidimo da u TCP/IP paketu protokola nema dva sloja, sjednički i predodžbeni sloj. Aplikacijski sloj u TCP/IP paketu se obično razmatra kao kombinacija sjedničkog, predodžbenog i aplikacijskog sloja kod OSI modela. Ovo je napravljeno iz dva razloga:

- TCP/IP ima više od jednog prijenosnog sloja i neke funkcionalnosti sjedničkog sloja su dostupne u prijenosnom sloju.
- Aplikacijski sloj nije jedini softver. Puno aplikacija se može napraviti uz pomoć ovog sloja. Ako su neke funkcionalnosti, koje se nalaze u sjedničkom i predodžbenom sloju, potrebne za neku posebnu aplikaciju, one se mogu uključiti u razvoju za taj dio tog softvera.

TCP/IP paket protokola je hijerarhijski protokol što znači da viši slojevi protokola su podržani jednim ili više nižih protokola, a sastoji se od interaktivnih modula od kojih svaki pruža posebne funkcionalnosti, ali ti moduli nisu nužno međusobno ovisni. Dok OSI model pruža spefičine funkcionalnosti koje pripadaju svakom od slojeva, slojevi kod TCP/IP paketa protokola sadržu relativno neovisne protokole koji se mogu miješati i usklađivati, zavisno od toga što sustav treba.

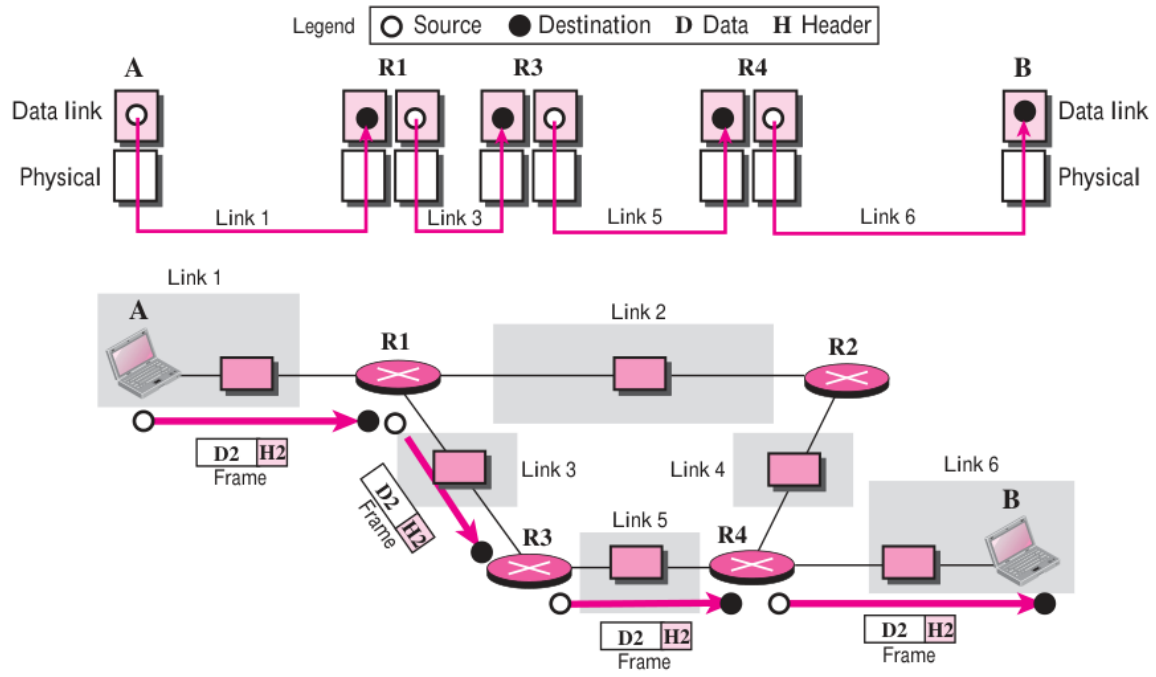
2.1.1. Slojevi TCP/IP paketa protokola

Fizički sloj



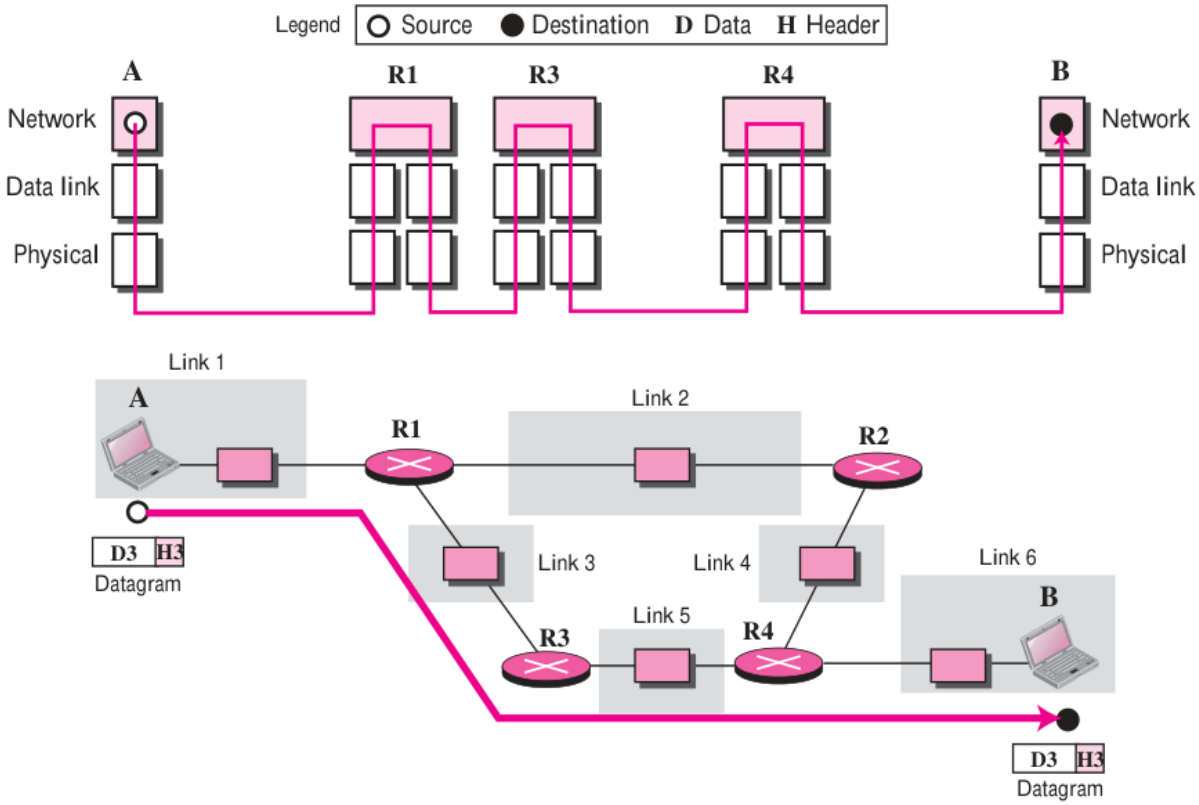
Slika 27. Komunikacija na fizičkom sloju (preuzeto iz [2])

TCP/IP ne definira neki specifični protokol za fizički sloj. Na ovom sloju imamo komunikaciju između dva čvora, usmjerivača ili računala. Kada se uspostavi komunikacija između čvorova, tok bitova će proteći između njih, ali će ih fizički sloj tretirati odvojeno. Na slici 27. vidimo put bitova između računala A i računala B. Računalo A šalje svaki bit usmjerivaču R1 u formatu korištenom za tu vezu. Usmjerivač R1 šalje svaki bit usmjerivaču R3 u formatu koji se koristi za tu vezu i tako dalje.



Slika 28. Komunikacija na podatkovnom sloju (preuzeto iz [2])

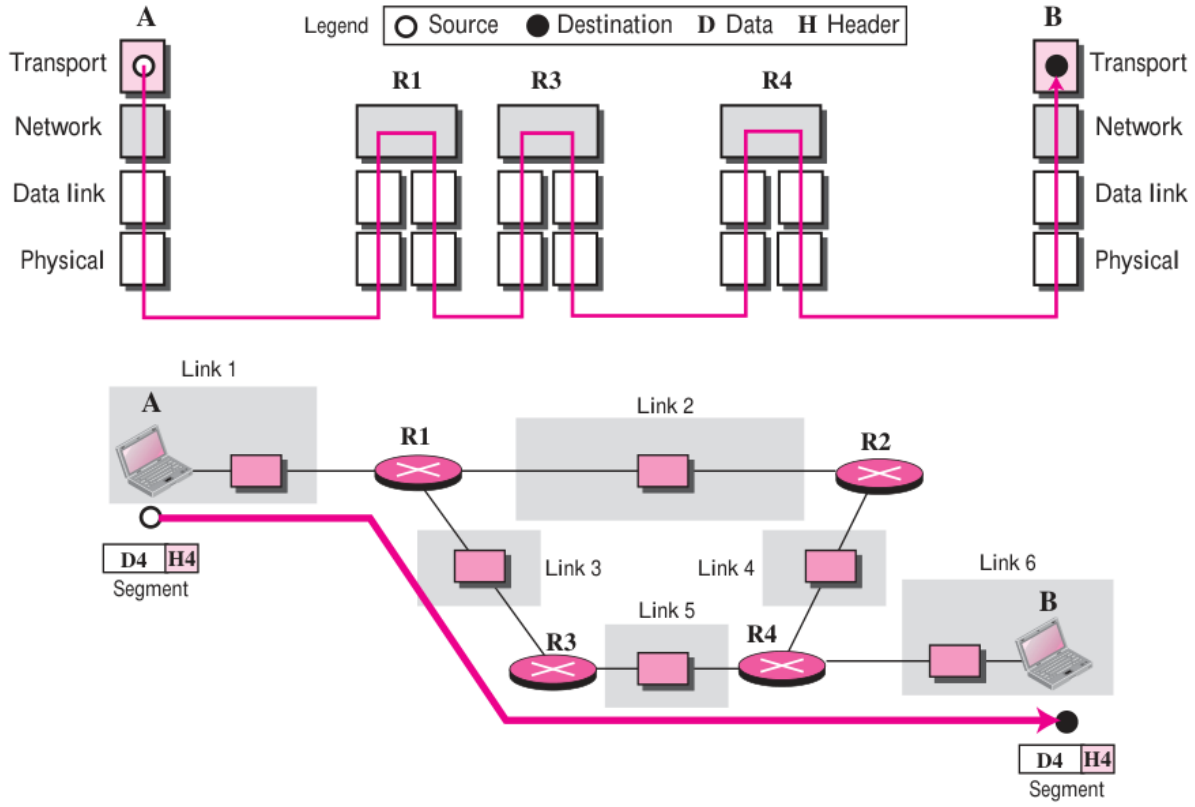
Slično kao i kod fizičkog sloja, TCP/IP ne definira nikakav specifični protokol za podatkovni sloj i na ovom sloju je komunikacija između čvorova, ali za razliku od fizičkog sloja, kod kojih se koristu bitovi, ovdje se koristu okviri (eng. *Frame*). Okvir predstavlja paket koji enkapsulira podatke poslane od mrežnog sloja sa dodanim zaglavljem i ponekad podnožjem. Zaglavlje, osim informacija za komunikaciju, još sadrži i izvor i odredište od okvira. Odredišna adresa je potrebna da bi se identificirao odgovarajući primatelj okvira, a izvorna adresa je potrebna kao identifikacija kod nekih protokola. Slika 28. prikazuje komunikaciju na podatkovnom sloju. Okvir koji putuje od računala A do usmjerivača R1 može biti drugačiji od onog koji putuje između usmjerivača R1 i usmjerivača R3. Kada okvir dospije do usmjerivača R1, ovaj usmjerivač šalje dospjeli okvir do podatkovnog sloja, gdje se okvir otvara i podaci se vadu iz okvira. Podaci se onda šalju drugom podatkovnom sloju koji pravi novi okvir koji će se poslati usmjerivaču R3. Razlog ovome je zbog toga što veze mogu koristiti drugačije protokole i zahtjeva drugačije formate okvira. Jedna stvar koja se može primjetiti na slici je da nema fizičkog prijenosa okvira. To je zbog toga što se fizički prijenos odvija samo u fizičkom sloju, dok se u podatkovnom sloju odvija samo logička komunikacija. Drugim riječima, podatkovni sloj u usmjerivaču R1 samo razmišlja o okviru koji se direktno poslao od podatkovnog sloja računala A.



Slika 29. Komunikacija na mrežnom sloju (preuzeto iz [2])

Na mrežnom sloju, TCP/IP podržava internetski protokol (IP) i pomoću internetskog protokola se vrši sami prijenos. Na slici 29. se vidi komunikacija na mrežnom sloju. Glavna razlika između komunikacije na mrežnom sloju i komunikacije na podatkovnom ili na fizičkom sloju je što je komunikacija na mrežnom sloju od kraja do kraja (eng. *End To End*), dok je u druga dva sloja komunikacija od čvora do čvora.

Prijenosni sloj



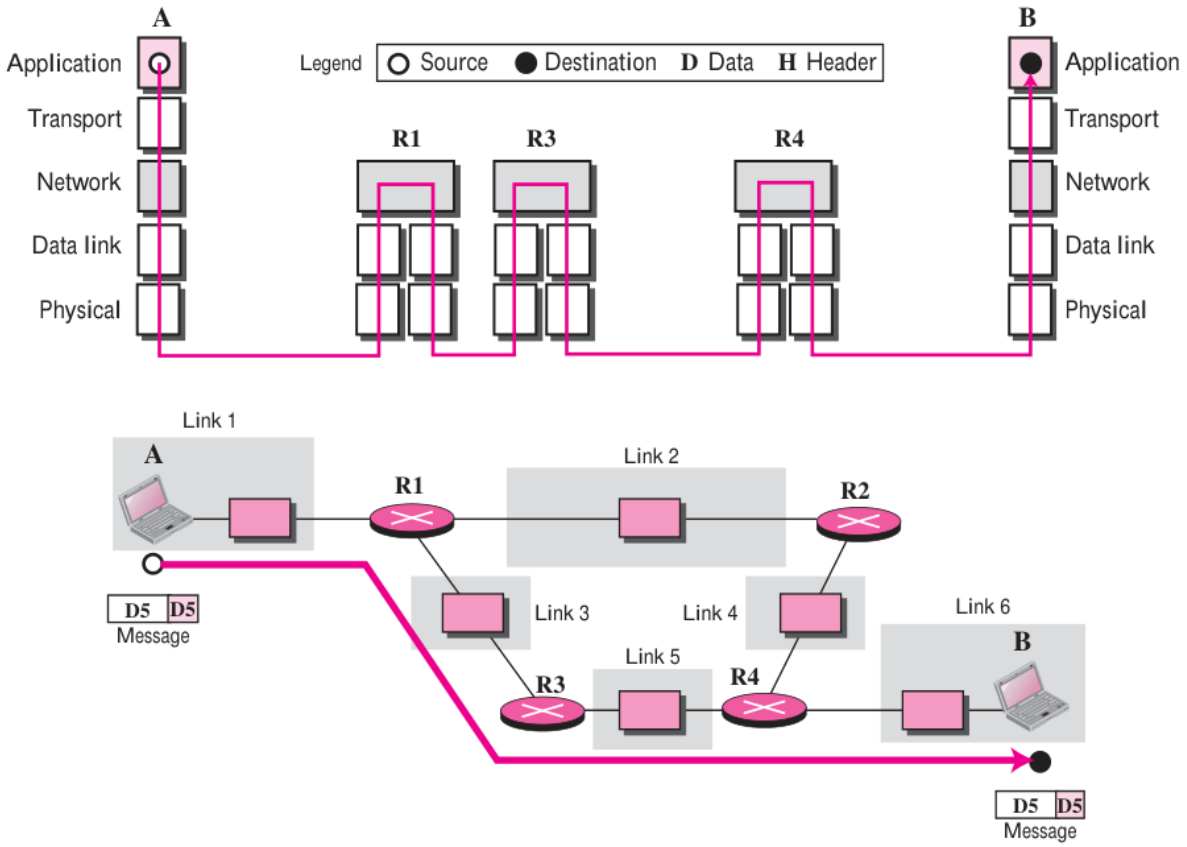
Slika 30. Komunikacija na prijenosnom sloju (preuzeto iz [2])

Glavna razlika između prijenosnog i mrežnog sloja je u tome što mrežnim slojem se šalju pojedini datagrami od računala A do računala B, dok je prijenosni sloj odgovoran za dostavljanje cijele poruke, segmenta ili korisničkog datagrama, od A do B. A segment se može sastojati od nekoliko desetina datagrama. Segmenti se trebaju razbiti u datagrame i onda se svaki datagram treba dostaviti do mrežnog sloja za prijenos, a prijenosni sloj na odredištu treba sačekati da se svi datagrami dostave i ponovno sastavu u segment.

Trebamo znati da dva prijenosna sloja misle da komuniciraju jedan sa drugim preko segmenta, ali se komunikacija vrši preko fizičkog sloja i razmjenom bitova.

Prijenosni sloj se prikazuje pomoću dva protokola: TCP i UDP, o kojima ćemo kasnije pričati.

Aplikacijski sloj



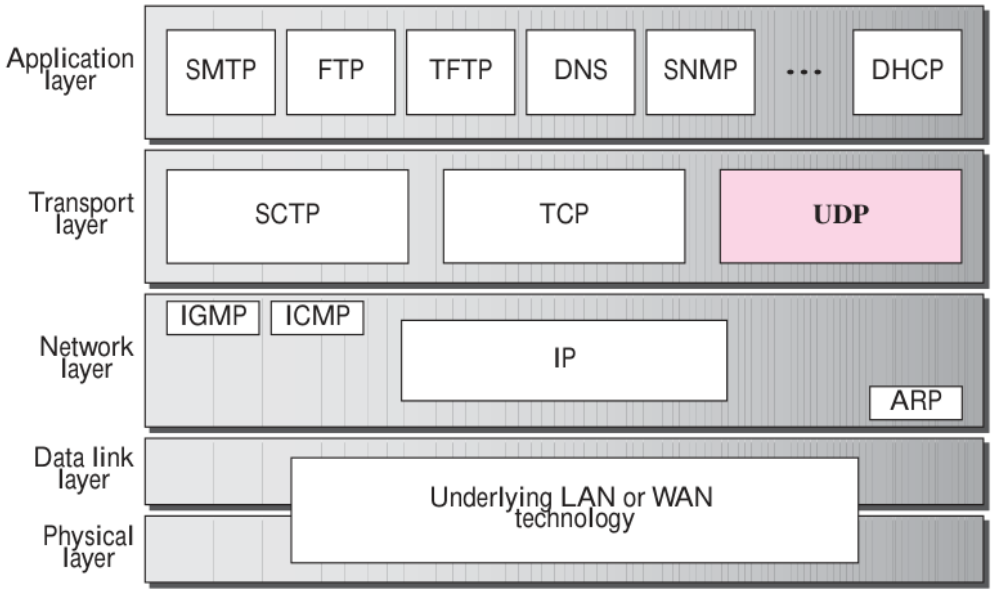
Slika 31. Komunikacija na aplikacijskom sloju (preuzeto iz [2])

Aplikacijski sloj u TCP/IP-u je jednak kombinaciji sjedničkog, predodžbenog i aplikacijskog sloja u OSI modelu. Aplikacijski sloj omogućava korisniku pristup servisima privatnog i javnog interneta. Mnogo protokola je definirano u ovom sloju da bi pružili servise kao elektronička pošta, transfer datoteka i slično. Imajte na umu da je komunikacija na aplikacijskom sloju slična onoj na prijenosnom sloju.

2.1.2. Prijenosni protokoli TCP/IP paketa protokola

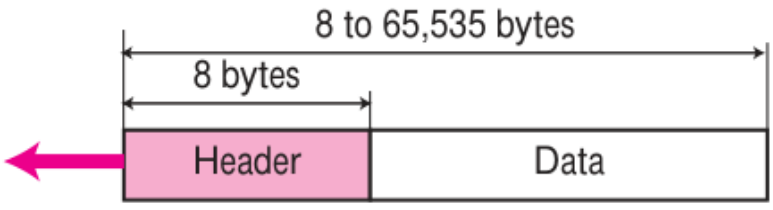
Imamo dva glavna protokola o kojima ćemo detaljno pričati: UDP i TCP. Isto imamo i SCTP (eng. *Stream Control Transmission Protocol*) koji je novi protokol, ali nije tema ovog završnog.

UDP Protokol



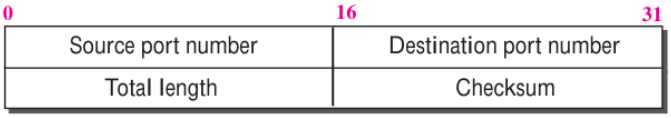
Slika 32. Položaj UDP-a u TCP/IP paketu protokola (preuzeto iz [2])

UDP Protokol (eng. *User Datagram Protocol*) je nepouzdani prijenosni protokol bez spajanja veze [3]. Ne dodanje ništa novo funkcijama internetskog protokola. Jedino omogućava komunikaciju od procesa do procesa umjesto od jednog izvora do drugog izvora. UDP je veoma jednostavni protokol pomoću kojeg se šalju male poruke i pošaljitelja nije briga za pouzdanost. Slanjem malih poruka korištenjem UDP-a uzima manje međudjelovanja između pošaljitelja i primatelja.



Slika 33. Format korisničkog datagrama (preuzeto iz [2])

UDP paketi ili korisnički datagrami imaju fiksnu duljinu zaglavlja od 8 bytova nakon kojeg slijedu podaci.



Slika 34. Format zaglavlja korisničkog datagrama (preuzeto iz [2])

Zaglavlje korisničkog datagrama se sastoji od:

- Izvornog broja porta (eng. *Source Port Number*) – broj porta kojeg procesi koristu na izvoru. Veličine je 16 bita, što znači da ovaj broj može biti od 0 do 65535. Ako je izvorni uređaj klijent koji šalje zahtjev, broj porta je u većini slučaja kratkotrajni broj porta zatražen od strane procesa i izabran UDP softverom koji je pokrenut na izvornom uređaju. Ako je izvorni uređaj server, broj porta je u većini slučaja dobro poznati broj.
- Odredišnog broja porta (eng. Destination Port Number) – broj porta kojeg procesi koristu na odredištu. Isto je veličine 16 bita. Ako je odredišni uređaj server, broj porta je u većini slučaja dobro poznati broj, a ako je klijent, onda je broj porta u većini slučaja kratkotrajni broj porta.
- Duljina (eng. *Total Length*) – 16 bitno polje koje definira cijelu duljinu korisničkog datagrama, odnosno zaglavlja i podataka. Iako duljina može biti od 0 do 65535, ona mora biti puno manja od toga, jer se korisnički datagram sprema u IP datagramu koji ima veličinu 65535 bytova.
- Polja za provjeru (eng. *Checksum*) – ovo polje se koristi za otkrivanje grešaka u korisničkom datagramu.

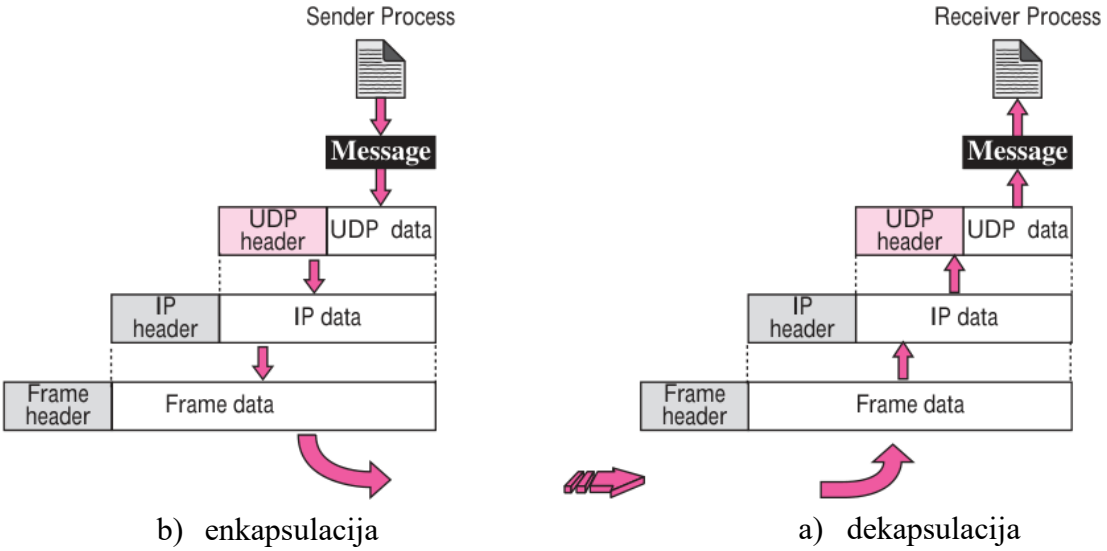
UDP protokol omogućava komunikaciju od procesa do procesa koristeći kombinaciju IP adresa i portova.

Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
53	Domain	Domain Name Service (DNS)
67	Bootps	Server port to download bootstrap information
68	Bootpc	Client port to download bootstrap information
69	TFTP	Trivial File Transfer Protocol
111	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol
162	SNMP	Simple Network Management Protocol (trap)

Tablica 1. Tablica poznatih portova korištenih u UDP protokolu (preuzeto iz [2])

Kao što smo prije naveli, UDP protokol koristi vezu bez spajanja, što znači da se svaki datagram šalje kao zasebni datagram. Nema veza između različitih datagrama čak i ako dolaze od istog izvornog procesa i idu na isto odredište i datagrami nisu numerirani. Isto tako nije uspostavljena nikakva veza što znači da svaki korisnički datagram može putovati različitim putem. Jedna od posljedica veze bez spajanja je što proces koji koristi UDP ne može poslati tok podataka na UDP i očekivati da će ih UDP podijeliti na različite povezane korisničke datagrame. Zbog toga svaki zahtjev treba biti dovoljno mali da stane u jedan korisnički datagram i zato se UDP koristi za slanje malih poruka koje su manje od 65535 byteova, odnosno 65507 bytova, jer se treba uzet u obzir veličina zaglavlja i zaglavlje internetskog protokola.

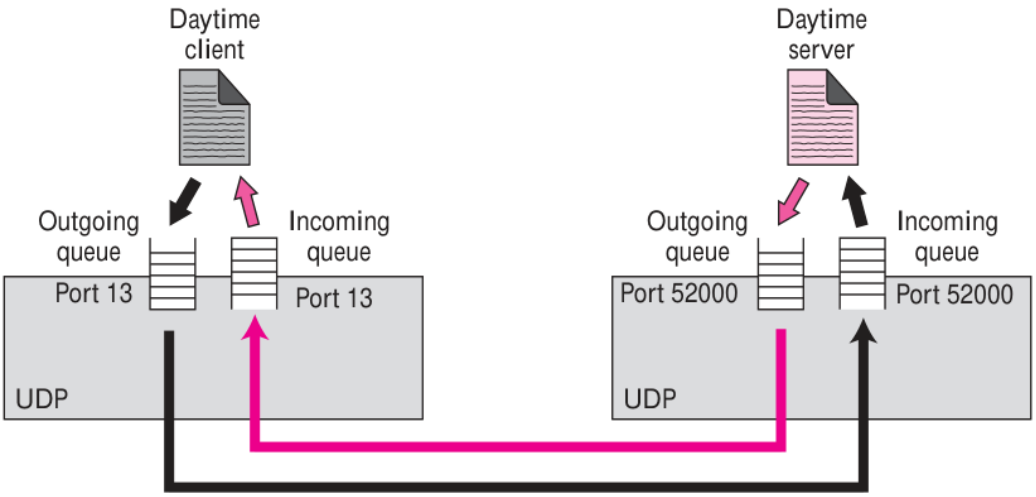
Zbog toga što je UDP jednostavan protokol, nema kontrole protoka, što znači da se primatelj može „preplaviti“ sa dolaznim porukama. Isto tako nema ni kontrole grešaka osim kod polja za provjeru, što znači da pošaljitelj ne može znati je li se poruka izgubila ili se duplicirala. Kada primatelj otkrije grešku preko polja za provjeru, korisnički datagram se potiho odbaci. Isto tako, zbog jednostavnosti UDP-a, nema ni kontrole zagušenja jer UDP misli da su paketi dovoljno mali i da je razmak između slanja paketa dovoljno velik da neće doći do zagušenja u mreži.



Slika 35. Enkapsulacija i dekapsulacija poruke (preuzeto iz [2])

Da bi se poslala poruka od jednog procesa do drugog, UDP protokol ju enkapsulira na izvoru i onda ju dekapsulira na odredištu.

Kao što vidimo na slici 35. kada proces želi poslati poruku kroz UDP protokol, on ju šalje kroz UDP uz par adrese i duljine podatka. UDP primi podatak i doda UDP zaglavlje i onda šalje korisnički zahtjev internetskom protokolu koji doda svoje zaglavlje i doda vrijednost 17 u polje protokola tako da se zna da je podatak došao od UDP protokola. Zatim se IP datagram šalje podatkovnom sloju, koji primi datagram i doda svoje zaglavlje i ponekad podnožje ako je potrebno i pošalje ga u fizički sloj koji dekodira bitove u električne i optičke signale i šalje ih na odredište. Kada poruka dođe do odredišta, fizički sloj dekodira signale u bitove i šalje ih na podatkovni sloj koji koristi zaglavlje i podnožje ako postoji da provjeri podatke. Ako nema greške, zaglavlje i podnožje se odbacuju i datagram se šalje internetskom protokolu koji ga opet provjerava i ako nema greške, odbacuje zaglavlje i datagram se šalje UDP protokolu sa pošaljiteljevom i primateljevom IP adresom. UDP koristi polje za provjeru da provjeri cijeli korisnički datagram i ako nema grešaka, zaglavlje se odbacuje i podaci se šalju procesu.



Slika 36. Primjer reda u UDP protokolu (preuzeto iz [2])

Redovi u UDP protokolu su povezani sa portovima. Na strani klijenta, kada se proces pokrene, on zatraži broj porta od operacijskog sustava. Neke implementacije operacijskog sustava kreiraju dolazne i odlazne redove čekanja (eng. *Queue*) koji su povezani sa svaki procesom, a neke kreiraju samo dolazne redove za svaki proces.

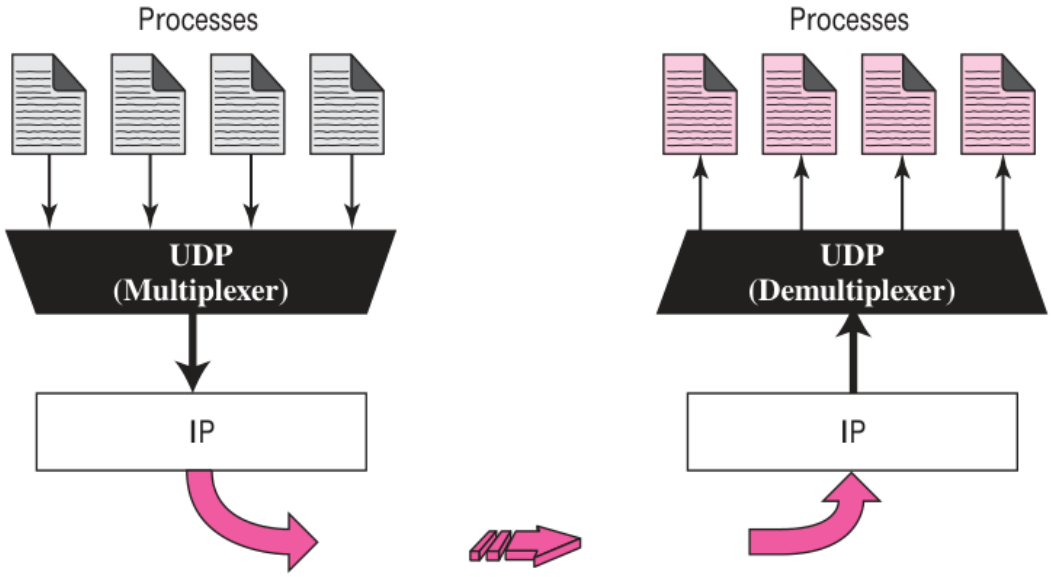
Čak i ako proces želi komunicirati sa više procesa, on može dobiti samo jedan portni broj i jedan dolazni i jedan odlazni red čekanja. Redovi čekanjna koji su otvoreni sa klientove strane su identificirani sa kratkotrajnim brojem porta. Redovi čekanja su aktivni sve dok je proces pokrenut. Čim se proces zaustavi, redovi čekanja se uništavaju.

Klijentni proces može poslati poruke odlaznim redovima čekanja koristeći izvorni broj porta specificiran u zahtjevu. UDP će otkloniti poruke jednu po jednu, dodavajući zaglavlje i dostavljajući ih internetskom protokolu. Odlazni red čekanja se može poplaviti. Ako se ovo dogodi, operacijski sustav će zatražiti da klijentni proces sačeka sa slanjem daljnih poruka dok se ovo ne popravi.

Kada poruka dospije do klijenta, UDP provjerava da li se neki dolazni red čekanja kreirao na portu specificiranom na odredišnom portu od korisničkog datagrama. Ako postoji takav red čekanja, UDP će poslati taj korisnički datagram na kraj tog reda čekanja, a ako taj red čekanja ne postoji, UDP će odbaciti korisnički datagram i zatražiti ICMP protokol da pošalje poruku serveru da je taj port nedostupan. Sve dolazne poruke od jednog klijentnog procesa, bez obzira da li dolazu od istog ili različitog servera, se šalju na isti red čekanja. Dolazni red čekanja se isto može poplaviti porukama, a ako se ovo dogodi, UDP će odbaciti korisničke datagrame i zatražiti da se pošalje poruka serveru da je taj port nedostupan.
Na strani servera, mehanizam redova čekanja je drugačiji. U najjednostavnijoj formi, server pita za dolazne i odlazne redove čekanja koristeći neki poznati port kada se pokrene. Redovi čekanja će ostati otvoreni dok god je server upaljen.

Kada poruka stigne do servera, UDP provjerava da li se dolazni red čekanja stvorio za taj port specificiran u polju korisničkog datagrama. Ako postoji takav red čekanja, UDP šalje taj korisnički datagram na kraj tog reda, a ako ne postoji, UDP će odbaciti taj korisnički datagram i pitati ICMP protokol da pošalje poruku klijentu da je taj port nedostupan. Sve dolazne poruke jednog servera, bez obzira da li doalzu od istog ili različitog klijenta, se šalju u isti red čekanja. Dolazni red čekanja se može poplaviti, a ako dođe do toga, UDP će odbaciti korisnički datagram i zamoliti da se pošalje poruka klijentu koja govori da je taj port nedostupan.
Kada server želi odgovoriti klijentu, on pošalje poruku odlaznom redu čekanja koristeći port koji je specificiran u zahtjevu. UDP će uklanjati poruke, jednu po jednu, i dodavat im svoje zaglavlje i slati ih internetskom protokolu. Odlazni red čekanja se može preplaviti, a ako se to dogodi, operacijski sustav će zatražiti od servera da sačeka sa slanjem novih poruka.

Na jednom sustavu imamo samo jedan UDP protokol i to može predstavljati problem kada više procesa na jednom sustavu želi koristiti funkcije i servise UDP protokola. Da bi se ovo riješilo, UDP protokol se multipleksira i demultipleksira.

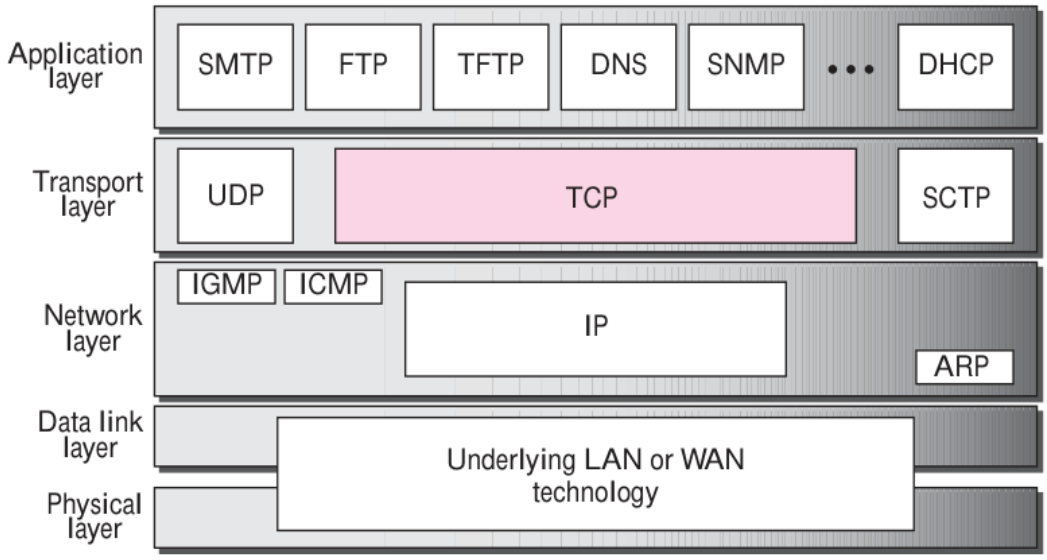


Slika 37. Multipleksiranje i demultipleksiranje UDP protokola (preuzeto iz [2])

Multipleksiranje UDP protokola (eng. *Multiplexing*) je postupak kombiniranja više tokova podataka na jedan UDP protokol, koje se razlikuju pomoću njihovog porta. Nakon što UDP protokol doda UDP zaglavlje, on šalje taj datagram internetskom protokolu.

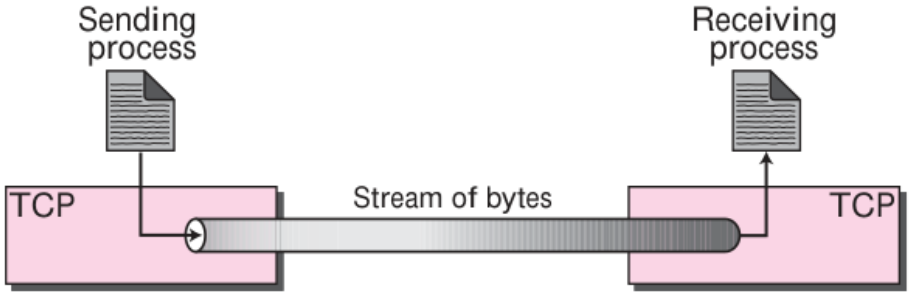
Demultipleksiranje UDP protokola (eng. *Demultiplexing*) je postupak razdvajanja kombiniranih tokova podataka koji su pristigli na jedan UDP protokol. UDP protokol će primiti korisničke datagrame i provjeriti imaju li ikakve greške. Ako nemaju grešaka, UDP protokol će odbaciti zaglavlja i poslati poruke procesu kojemu pripada pomoću broja porta.

TCP Protokol



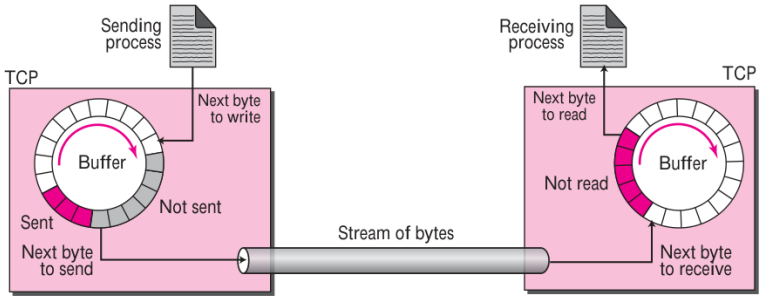
Slika 38. Položaj TCP-a u TCP/IP paketu protokola (preuzeto iz [2])

TCP protokol (eng. *Transmission Control Protocol*) je za razliku od UDP protokola konekcijski orijentiran protokol. To znači da omogućava procesu da pošalje podatke kao tok bitova i dozvoljava primajućem procesu da primi podatke kao tok bitova [3]. TCP protokol kreira okruženje u kojem su dva procesa komuniciraju kao da su povezani „cijevima“ (eng. *Tube*) koje prenosu njihove bitove, kao što vidimo na slici 39.



Slika 39. Isporuka toka bitova (preuzeto iz [2])

Zbog slanja i primanja toka bitova, procesi ne moraju nužno pisati i čitati podatke u isto vrijeme, TCP protokol treba međuspremnik (eng. *Buffer*) za njihovu pohranu. TCP protokol ima 2 međuspremnika: pošaljiteljev međuspremnik i primateljev međuspremnik, jedan za svaku stranu.

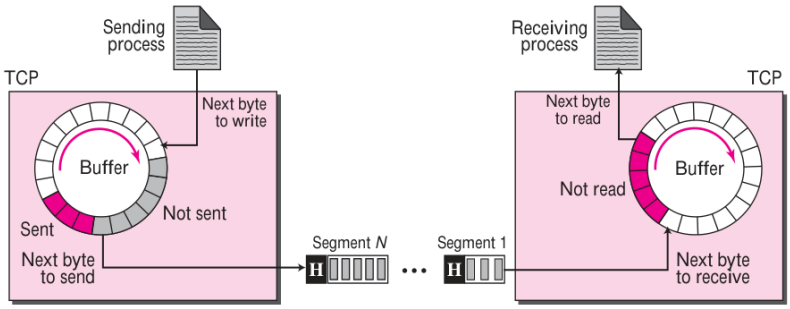


Slika 40. Slanje i primanje međuspremnika (preuzeto iz [2])

Slika 40. pokazuje kretanje podataka u jednom smjeru. Radi jednostavnosti ovdje imamo dva međuspremnika od 20 byte-ova svaki. Obično međuspremnici imaju stotine ili tisuće byte-ova, ovisno o implementaciji. Isto tako pokazujemo međuspremnike iste veličine, što nije uvijek slučaj.

Kod pošaljitelja, međuspremnik ima tri vrste komora (eng. *Chambers*). Bijeli dio sadrži prazne komore koje se mogu napuniti. Obojani dio sadrži byte-ove koji su se poslali, ali još nisu priznati jer pošaljiteljev TCP ih čuva u međuspremniku dok ne primi potvrdu od primatelja da su došli. Sivi dio sadrži byte-ove koji se još trebaju poslati. Kod primatelja su operacije sa međuspremnicima jednostavnije. Međuspremnik je podijeljen u dva dijela: obojeni i bijeli dio. Bijeli dio sadrži prazne komore koje čekaju da se napunu byte-ovima, a obojeni dio sadrži primljene byte-ove koji proces može pročitati. Kada se byte pročita, komora će se reciklirat i pridružiti praznim komorama.

Iako privremeno spremanje rješava nerazmjernost između brzine procesa kreiranja i procesa čitanja, još imamo jedan korak prije nego što možemo poslati podatke. IP sloj treba poslati pakete, a ne tok bitova. Zbog toga TCP protokol će grupirati nekoliko byteova u paket kojega zovemo segment. TCP dodaje zaglavlje svakom segmentu radi kontrole i šalje taj segment u IP sloj za prijenos. Segment će se enkapsulirati u IP datagram i prenjenti. Cijela ova operacija je transparentna procesu primanja.



Slika 41. Prijenos segmenata (preuzeto iz [2])

TCP protokol omogućava dvosmjerni način rada (eng. *Full-Duplex Service*), gdje se podaci mogu prenositi u oba smjera u isto vrijeme. Tada svaki krajnja točka TCP protokola ima svoj međuspremnik za slanje i primanje [2].

Kao i UDP, i TCP protokol radi multipleksiranje kod pošaljitelja i demultipleksiranje kod primatelja, ali, pošto je TCP konekcijski orijentiran protokol, veza se treba uspostaviti za svaki par procesa.

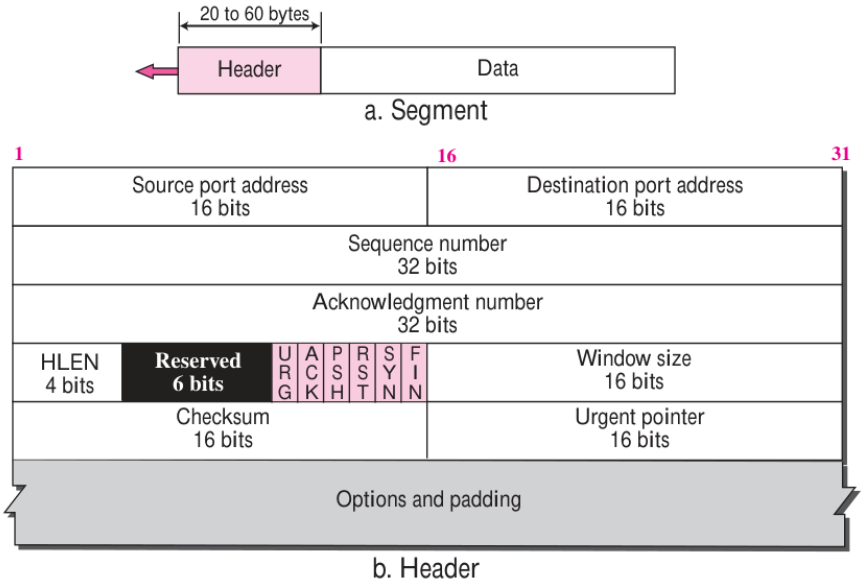
Kada proces želi poslati i primiti podatke od drugog procesa na drugoj lokaciji, javit će se:

1. Dva TCP protokola će uspostaviti „virtualnu vezu“ između njih
2. Podaci će se razmijeniti
3. Veza će se prekiniti

Ovdje nema fizičke veze. Pošto se segment TCP protokola šalje u IP datagramu, on se može poslati u krivom redoslijedu ili izgubiti ili se može koruptirati, pa se onda ponovno šalje. TCP protokol će napraviti okruženje u kojem će on prihvatiti odgovornost za dostavljanje byte-ova u određenom redoslijedu na drugu lokaciju.

TCP protokol numerira sve blokove podataka koji se prenosu. Numeriranje je zasebno za svaki smjer. Kada primi byte-ove podataka od procesa, TCP ih spremi u međuspremnik i onda ih numerira. Numeriranje ne mora nužno počet od nule, nego TCP izabire neki proizvoljni broj između 0 i 2^{32}. Nakon što su se svi byte-ovi numerirali, TCP protokol dodjeluje redni broj svakom segmentu koji se trebaju poslati.

TCP protokol za razliku od UDP protokola, ima kontrolu protoka. Ovo je implementirano da bi se spriječilo da se primatelj preplavni podacima. Isto tako, TCP protokol ima kontrolu grešaka i kontrolu zagušenja.

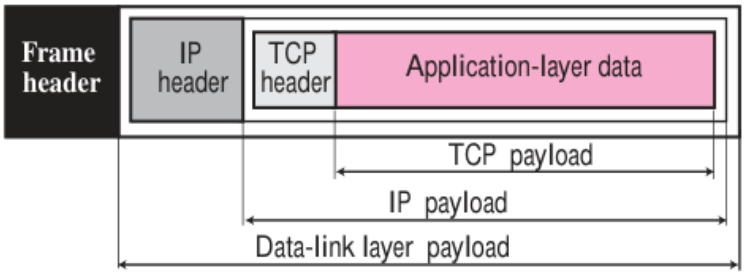


Slika 42. Format TCP segmenta i njegovog zaglavlja (preuzeto iz [2])

Segment TCP protokola se sastoji od zaglavlja, koje je veličine $20 - 60$ byte-ova i podataka. Zaglavlje se sastoji od:

- Izvornog broja usluge (eng. *Source Port Address*) – 16 bitno polje koje definira port procesa koji šalje segment.
- Odredišnog broja usluge (eng. *Destination Port Address*) – 16 bitno polje koje definira port procesa koji prima segment.
- Rednog broja (eng. *Sequence Number*) – 32 bitno polje koje definira broj koji je dodijeljen prvom byteu podataka u segmentu. On govori odredištu koji byte u ovoj sekvenci je prvi byte u segmentu.
- Broja potvrde (eng. *Acknowledgment Number*) – 32 bitno polje koje definira broj byte-a kojeg primatelj segmenta očekuje primiti od pošaljitelja.
- Dužine zaglavlja (eng. *Header Length*) – 4 bitno polje koje indicira broj od četiri bytne riječi u zaglavlju TCP protokola. Dužina može biti $20 - 60$ byte-a.
- Zastavice (eng. *Control*) – sadrži se od 6 jednobitnih zastavica koje imaju svoju funkciju:
    - URG (eng. *Urgent Pointer*) – za hitno slanje
    - ACK (eng. Acknowledgement Number) – potvrda primanja
    - PSH (eng. *Pushed Dana*) – pražnjenje spremnika
    - RST (eng. *Reset*) – resetiranje veze
    - SYN (eng. *Synchronization*) – sinkronizacija
    - FIN (eng. *Final*) – oslobađanje veze

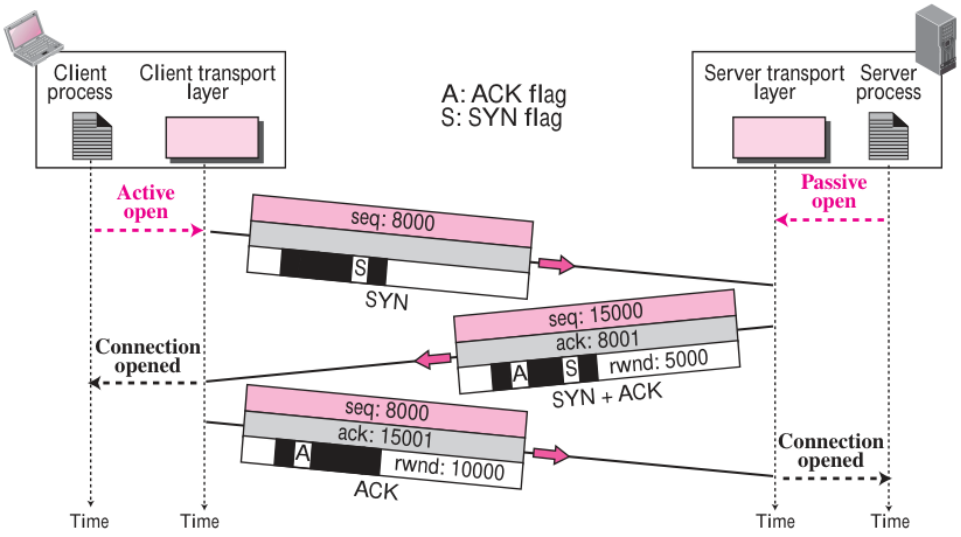
- Veličine prozora (eng. *Window Size*) – ovo polje definira veličinu prozora za slanje byteova. Ovo je 16 bitno polje, što znači da je maksimalna veličina prozora 65535 byte-ova.
- Polja za provjeru (eng. *Checksum*) – 16 bitno polje za provjeru koje provjerava ispravnost zaglavlja i podataka koji slijedu
- Pokazatelja na važne podatke (eng. *Urgent Pointer*) – 16 bitno polje pokazivača hitnosti koji pokazuje na važne podatke.
- Opcija (eng. *Options*) – polje u kojem može biti do 40 byteova dodatnih mogućnosti kojih inače nema u zaglavlju.



Slika 43. Enkapsulacija segmenta (preuzeto iz [2])

TCP segment enkapsulira podatke koje prima i onda se enkapsulira u IP datagram, nakon čega se enkapsulira u okvir u podatkovnom sloju.

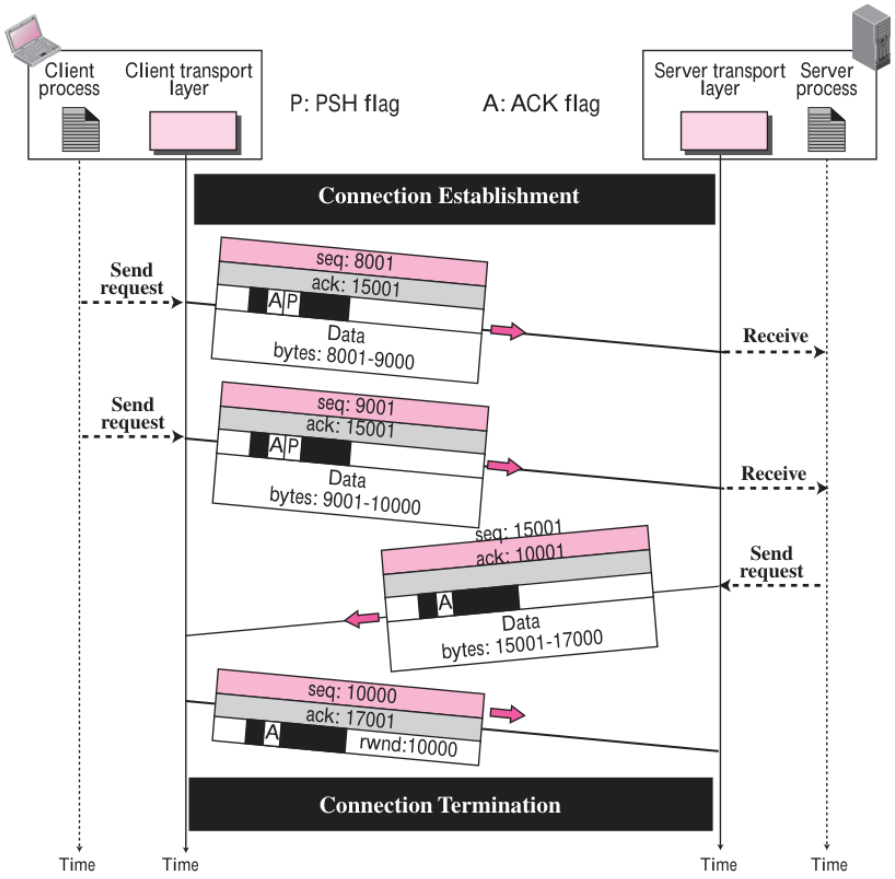
Uspostavljena veza u TCP protokolu se zove trostruko rukovanje (eng. *Three-Way Handshaking*), tj. dvije strane se dogovaraju o uspostavi veze izmjenom tri segmenta sa upravljačkim porukama.



Slika 44. Trostruko rukovanje (preuzeto iz [2])

Trostruko rukovanje se izvršava sljedećim koracima:

1. Klijent prvo pošalje segment koji se zove SYN segment, koji je segment kod kojeg je SYN bit zastavice postavljen u 1. Ovim segmentom klijent obavještava drugu stranu da želi sa njom komunicirati i šalje slijedni broj kojim pošiljatelj označava slijed svojih segmenata i koristi se da bi se sačuvao redoslijed podataka. SYN segment ne može nositi podatke, on samo nosi slijedni broj.
2. Server šalje drugi segment koji se zove SYN + ACK segment kod kojeg dva bita zastavice postavljeni u 1: SYN i ACK. Ovim segmentom pozvana strana potvrđuje prijem poziva i kaže pozivajućoj strani od kojeg broja će ona početi brojati svoje segmente i onda će server poslati segment kojim potvrđuje primanje segmenta od klijenta i šalje svoje podatke. SYN + ACK segment ne može nositi podatke.
3. Klijent šalje treći segment koji se zove ACK segment kojim priznaje potvrdu od servera.
4. Kada se prenesu svi podaci, dva uređaja trostrukim rukovanjem razmjenjuju segmente sa upravljačkim informacijama koje sadrže FIN bit kojom se veza prekida.



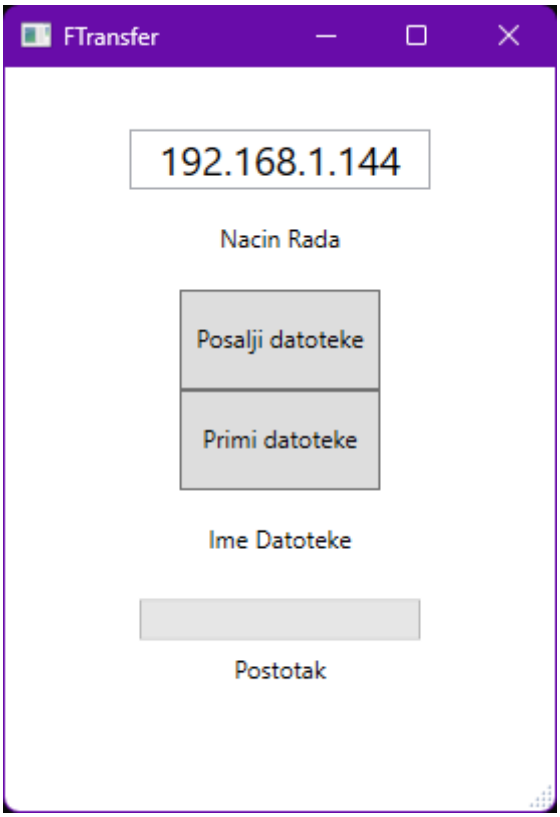
Slika 45. Prijenos podataka pomoću trostrukog rukovanja (preuzeto iz [2])

3. Praktični dio

3.1. *FTransfer* aplikacija

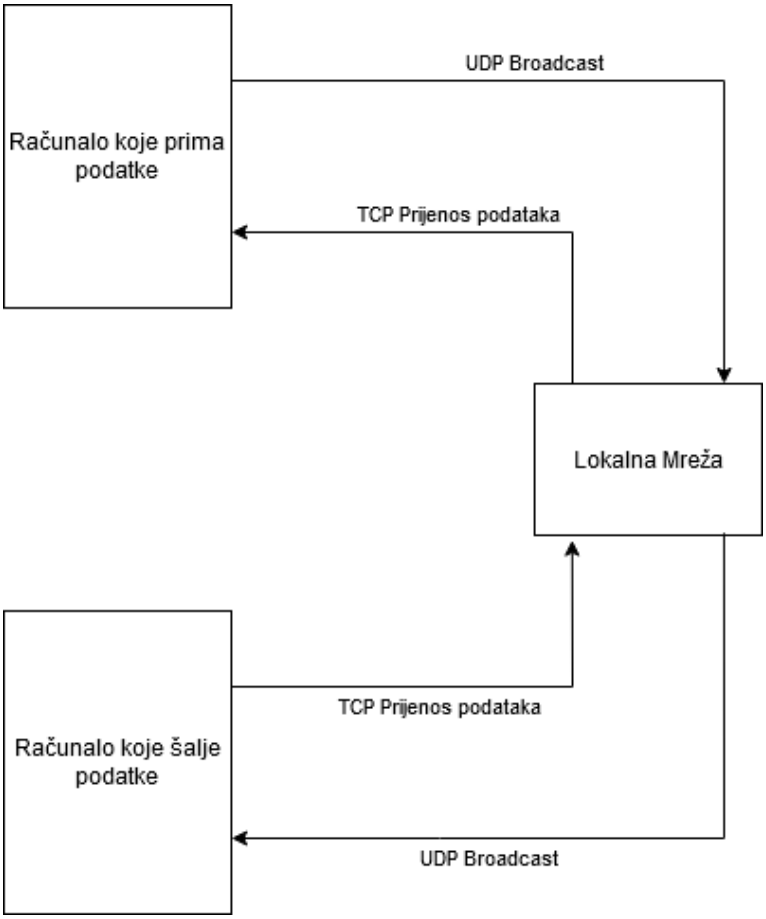
FTransfer je naziv aplikacije koju smo izradili u okviru ovog projekta kao primjer korištenja TCP/IP paketa protokola za prijenos podataka sa jednog računala na drugi. Pošto je ovo aplikacija napravljena u C# .NET Framework-u, radi samo na Windows računalima, ali se ovo može napraviti u drugim jezicima sa podrškom za sve moguće uređaje. Kada se aplikacija pokrene, na početnom prozoru se prikazuje lokalna IP adresa koje računalo koristi, dugmad kojima određujemo hoće li ovo računalo primati ili slati datoteke drugom računalu i još par stvari koje prikazuju koja se datoteka šalje, postotak slanja i u kojem načinu rada aplikacija trenutno radi, tj. da li ovo računalo šalje ili prima datoteke.

U ovom dijelu rada prikazat ćemo kako se šalju i primaju datoteke preko aplikacije i zašto je aplikacija pravljena u C#. Slike koje će prikazivati korištenje aplikacije su vlastito snimljene iz projekta, te zbog toga nemaju reference.



Slika 46. Početni prozor aplikacije

3.2. Općeniti rad aplikacije



Slika 47. Apstraktni primjer rada aplikacije

Na slici 47. je prikazan primjer rada aplikacije iz perspektive korisnika bez puno detalja i pomoću ove slike ćemo objasniti kako ova aplikacija radi.

Prvo što korisnik radi je na računalo sa kojeg želi poslati podatke postavlja u način rada „Slanje datoteka“ pritiskom na dugme „Pošalji datoteke“ prikazano na slici 46. Ovo postavlja aplikaciju da u način slušanja za UDP Broadcast. Ovo smo napravili zato jer trenutno računalo sa kojeg šaljemo podatke nema odredište gdje da ih šalje. Pomoću UDP-a ćemo mi zapravo poslati IP adresu odredišta na izvorno računalo, odnosno računalo koje bi trebalo slati podatke.

Ali da bi dobili taj UDP broadcast, moramo na odredišnom računalu aktivirati taj UDP Broadcast, a to se radi tako što pritisnemo dugme „Primi datoteke“. Ovo dugme šalje UDP Broadcast na Broadcast adresu i tako javljamo cijeloj lokalnoj mreži našu IP adresu i kada to naše izvorno računalo vidi, spremit će tu IP adresu i onda možemo poslati podatke na odredište.

3.2.1. Slanje podataka unutar programa

```
NacinRada.Content = "Slanje Datoteke...";
// Slusanje Broadcasta
UdpClient udpClient = new UdpClient(UDPPort);
udpClient.EnableBroadcast = true;
IPEndPoint serverEndPoint = null;
while (serverEndPoint == null)
{
    Console.WriteLine("Cekanje na broadcast");
    byte[] serverBroadcastBytes = udpClient.Receive(ref serverEndPoint);
    string serverBroadcast = Encoding.UTF8.GetString(serverBroadcastBytes);
    Console.WriteLine($"Broadcast: {serverBroadcast}");
    string[] tempSplit = serverBroadcast.Split(':');
    OtherPCIP = tempSplit[0];
    Console.WriteLine(OtherPCIP);
}
```

Slika 48. Primanje UDP Broadcasta unutar programa

Na slici 48. se vidi kako izvorno računalo prima UDP Broadcast o kojem smo ranije pričali. Pošto se šalje IP adresa sa portom, onda ih trebamo rastaviti jer port je već poznat, tako da nam nije potreban.

```
List<string> filePaths = new List<string>();
OpenFileDialog openFileDialog = new OpenFileDialog();
openFileDialog.Multiselect = true;
openFileDialog.Filter = "All Files (*.*)|*.*";
if (openFileDialog.ShowDialog() == true)
{
    string[] tempPaths = openFileDialog.FileNames;
    foreach (string path in tempPaths)
    {
        filePaths.Add(path);
    }
}
```

Slika 49. Odabir datoteka za slanje na odredište unutar programa

Zatim kad izađemo iz ove *while* petlje, tj. kada primimo IP adresu odredišta pomoću UDP protokola, otvara se File Dialog u kojem odabiremo datoteke koje želimo poslati na drugo računalo.

```
TcpClient client = new TcpClient(OtherPCIP, TCPPort);
NetworkStream stream = client.GetStream();

// Posalji koliko se datoteka treba poslat
byte[] fileCountBytes = BitConverter.GetBytes(filePaths.Count);
await stream.WriteAsync(fileCountBytes, 0, fileCountBytes.Length);
```

Slika 50. Slanje broja datoteka unutar programa

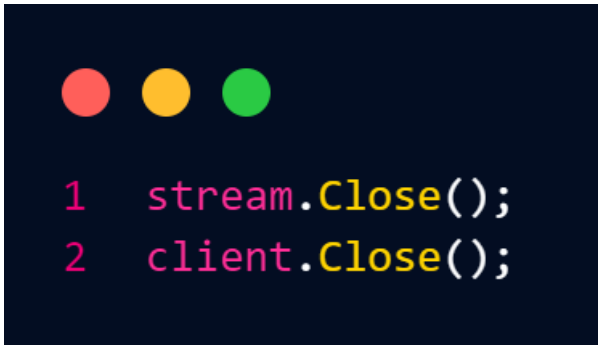
Nakon što smo odabrali sve datoteke za prijenos, pokrećemo TCP client i onda dohvaćamo tok (eng. *Stream*) od tog TCP clienta i prvo što šaljemo je broj datoteka koje ćemo poslati odredištu tako što taj broj datoteka pretvorimo u niz Byte-ova i onda pomoću funkcije *stream.WriteAsync()* šaljemo taj broj datoteka. Funkcija *stream.WriteAsync()* prima sam taj niz Byte-ova broja datoteka, zatim početni indeks u nizu Byte-ova (eng. *offset*) i na kraju sam broj elemenata u nizu Byte-ova (slično kao *for()* petlja).

```
// Za svaku datoteku iz liste se ponavlja isto
foreach (string path in filePaths)
{
    // Prvo se salje ime datoteke
    string fileName = Path.GetFileName(path);
    ImeDatoteke.Content = fileName;
    byte[] fileNameBytes = Encoding.UTF8.GetBytes(fileName);
    byte[] fileNameSize = BitConverter.GetBytes(fileNameBytes.Length);
    await stream.WriteAsync(fileNameSize, 0, fileNameSize.Length);
    await stream.WriteAsync(fileNameBytes, 0, fileNameBytes.Length);

    // Citaju se svi bytovi od datoteke i zapisuju se u niz bytova
    byte[] fileBytes = File.ReadAllBytes(path);

    // Posalji velicinu datoteke
    byte[] fileSize = BitConverter.GetBytes(fileBytes.Length);
    await stream.WriteAsync(fileSize, 0, fileSize.Length);

    // Posalji sadrzaj datoteke
    await stream.WriteAsync(fileBytes, 0, fileBytes.Length);

    // Kada se datoteka posalje, ocisti buffer
    await stream.FlushAsync();
}
```

Slika 51. Slanje datoteka u programu

Sada kada smo poslali broj datoteka koje trebamo poslati možemo poslati datoteke. Sve radimo na isti princip, samo je različito to što šaljemo. Prvo šaljemo duljinu imena datoteke koja se prvo pretvara u niz Byte-ova, i onda samo ime datoteke koje smo pretvorili u niz Byte-ova. Zatim čitamo sve Byte-ove od datoteke i zapisujemo ih u niz i onda šaljemo prvo veličinu datoteke i onda sami sadržaj datoteke. Kada smo sve poslali onda čistimo *buffer*

odnosno memoriju pomoću *stream.FlushAsync()* funkcije, inače bi datoteke ostale u radnoj memoriji i onda bi brzo ostali bez radne memorije kada bi slali velike datoteke i operacijski sustav bi postao nestabilan.



Slika 52. Gašenje TCP clienta unutar programa

Kada *forEach()* petlja završi sa radom, potrebno je da zatvorimo TCP protokol, inače bi taj TCP protokol ostao otvoren i kada bi sljedeći put slali datoteke trebali bi promijeniti port za TCP protokol, inače nebi mogli koristiti aplikaciju.

3.2.2. Primanje podataka unutar programa

```
// Kreiraj udpClient za Broadcast
UdpClient udpClient = new UdpClient();
udpClient.EnableBroadcast = true;

// Posalji IP adresu i port ovog racunala preko Broadcasta
byte[] broadcastBytes = Encoding.UTF8.GetBytes($"{ThisPCIP}:{TCPPort}");
IPEndPoint broadcastEndPoint = new IPEndPoint(IPAddress.Broadcast, UDPPort);
udpClient.Send(broadcastBytes, broadcastBytes.Length, broadcastEndPoint);
udpClient.Close();
```

Slika 53. Kreiranje i izvršavanje UDP broadcasta unutar programa

Na slici 53. vidimo samo kreiranje i slanje UDP Broadcasta unutar lokalne mreže. IP adresu i port pretvaramo u niz Byte-ova koje šaljemo na Broadcast.

```
TcpListener listener = new TcpListener(IPAddress.Parse(ThisPCIP), TCPPort);
listener.Start();

// Prihvati TCP od racunala koje salje datoteke
TcpClient client = listener.AcceptTcpClient();
NetworkStream stream = client.GetStream();
```

Slika 54. Pokretanje TCP protokola i čekanje na poziv od strane odredišnog računala

Pokretanje TCP protokola na odredišnom računalu je malo drugačije od pokretanja TCP protokola na izvornom računalu. Na odredišnom računalu zapravo pokrećemo takozvani „*TCPListener*“ koji čeka na TCP poziv od strane drugog računala. Kada dobijemo poziv, onda će *TCPListener* prihvatiti taj TCP poziv i tako ćemo otvoriti TCPClient na odredištu.

```
byte[] fileCountBytes = new byte[sizeof(int)];
await stream.ReadAsync(fileCountBytes, 0, sizeof(int));
int fileCount = BitConverter.ToInt32(fileCountBytes, 0);
```

Slika 55. Primanje broja datoteka unutar programa

Prvo što dobijemo od izvornog računala je broj datoteka koje će se poslati. Ovo nam je bitno kod primanja datoteka, ali o tome ćemo kasnije. Prvo radimo novi niz Byte-ova koji je veličine *int*-a (eng. *integer*) jer smo poslali broj kao *int* i onda koristimo funkciju *stream.ReadAsync()* za čitanje broja datoteka tako što zapisujemo primljene Byte-ove u niz kojeg smo stvorili veličine *int*-a i onda taj niz pretvaramo natrag u *int*.

```
for (int i = 0; i < fileCount; i++)
{
    // Citanje velicinu imena datoteke
    byte[] fileNameSize = new byte[sizeof(int)]; // Salje se 32-bitni int kao 8bitni niz bytova
    await stream.ReadAsync(fileNameSize, 0, sizeof(int));
    int size = BitConverter.ToInt32(fileNameSize, 0);
    byte[] buffer = new byte[size];
    int bytesRead = 0;

    // Citanje imena datoteke u buffer
    while (bytesRead < size)
    {
        bytesRead += await stream.ReadAsync(buffer, bytesRead, size - bytesRead);
    }
    string fileName = Encoding.UTF8.GetString(buffer); // Pretvori niz bytova u ime datoteke
    ImeDatoteke.Content = fileName;
    // Dohvati velicinu datoteke
    byte[] fileSize = new byte[sizeof(int)];
    await stream.ReadAsync(fileSize, 0, sizeof(int));

    // Citanje sadrzaja datoteke
    size = BitConverter.ToInt32(fileSize, 0);
    buffer = new byte[size];
    bytesRead = 0;
    while (bytesRead < size)
    {
        bytesRead += await stream.ReadAsync(buffer, bytesRead, size - bytesRead);
    }

    await File.WriteAllBytesAsync(Environment.CurrentDirectory + @"\" + fileName, buffer); // Spremi datoteku
    await stream.FlushAsync(); // Ocisti buffer od streama

    Console.WriteLine("Datoteka " + fileName + " primljena!");
}
```

Slika 56. Primanje datoteka unutar programa

Sada koristimo broj datoteka kojeg smo prvog primili jer znamo koliko će se datoteka poslati na odredište i možemo koristiti *for* petlju i znamo da će uvijek raditi. Ponavljamo korake kao i u slanju datoteka, samo što ovdje čitamo, a ne šaljemo. Čitamo veličinu datoteke slično kao i broj datoteka, ali kod čitanja imena datoteke je drugačije. Prvo pretvaramo veličinu datoteke u niz Byte-ova koji je dug kao veličina imena datoteke i onda deklariramo novu varijablu tipa *int* i postavljamo njenu vrijednost 0.

Zatim u *while* petlji, dok god je nova varijabla manja od veličine imena datoteke izvršavamo čitanje Byte-ova. Kada izađemo iz *while* petlje, pretvaramo niz dobivenih Byte-ova u ime datoteke. Ovaj proces ponavljamo za sadržaj datoteke: prvo čitamo veličinu datoteke, pa onda sadržaj datoteke i nakon toga pišemo datoteku na računalo. Na kraju, kao i kod slanja datoteka, pomoću funkcije *stream.FlushAsync()* čistimo memoriju.

Ovaj proces se ponavlja koliko se datoteka šalje. Kada se *for()* petlja izvrši, kao i kod slanja, zatvarmo TCP protokol pomoću naredbe *client.close()* i zatvaramo „*TCPListener*“ pomoću naredbe *listener.close()*.

4. Zaključak

Kada se sve uzme u obzir, mogu se izdvojiti prednosti i mane korištenja TCP protokola za prijenos datoteka. Neke od prednosti su: relativno brz prijenos (Ovisi o brzini lokalne mreže, ali je pretežito dovoljno brzo), nije potreban prijenosni disk/USB stick za prijenos datoteka, nego se sav prijenos vrši od jednog računala na drugo, veličina datoteke nije određena (Testirano je za datoteke do 10GB).

S druge strane, postoji nekoliko nedostataka. Kao prvo, od korisnika se očekuje da ima lokalnu mrežu, što danas nije toliki problem jer skoro svi imamo kućni uređaj kojeg smo dobili od operatera, ali ponekad ga treba podesiti da radi onako kako mi želimo. Još jedan od problema je što brzina zna biti nestabilna i Windows zna ovu brzinu odjednom samo uništiti i smanjiti do 1MB/s i onda će brzina ostati takva sve dok se Windows ne resetira u potpunosti.

Literatura

[1] A. Burilović i M. Lovričević, »Računalne Mreže,« 2022.

[2] B. A. Forouzan, TCP/IP Protocol Suite 4th Edition, New York: McGraw-Hill Education, 2009.

[3] C. M. Kozierok, The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference, Portland: No Starch Press, 2005.

[4] T. Rooney, IP Address Management: Principles and Practice, New Jersey: Wiley-IEEE Press, 2011.

[5] K. S. W. Fall, TCP/IP Illustrated, Volume 1: The Protocols (Addison-Wesley Professional Computing Series), New Jersey: Pearson Education, Inc, 2011.

Sadržaj slika

Slika 1.Koncept sučelja i razina 2
Slika 2. Dostava paketa preko internetskog protokola 4
Slika 3. Format IPv4 datagrama 5
Slika 4. Format IPv6 datagrama 7
Slika 5. Primjer IPv4 adrese u binarnom i decimalnom prikazu 10
Slika 6. Podjela IPv4 adresa po klasama 10
Slika 7. Format mrežne maske po klasi 11
Slika 8. Primjer IPv6 adrese u binarnom zapisu 12
Slika 9. Primjer IPv6 adrese u heksadekadskom zapisu 12
Slika 10. Podjela IPv6 adresa u odjeljke 13
Slika 11. Strategije za prijelaz na IPv6 protokol 13
Slika 12. Dual Stack strategija 14
Slika 13. Strategija tuneliranja 14
Slika 14. Strategija sa prevoditeljom zaglavlja 15
Slika 15. Tipovi ICMP poruka 16
Slika 16. Općeniti format ICMP poruka 16
Slika 17. Izgled ARP zahtjeva koja je multicast 17
Slika 18. Izgled ARP odziva koji je unicast 17
Slika 19. Izgled ARP paketa 18
Slika 20. Svi slučajevi za korištenje ARP-a 20
Slika 21. OSI Model 22
Slika 22. Slojevi OSI modela 23
Slika 23. Razmjena podataka korištenjem OSI modela 24
Slika 24. Kratki opis slojeva OSI modela 30
Slika 25. Slojevi TCP/IP paketa protokola 31
Slika 26. Usporedba TCP/IP paketa protokola i OSI modela 31
Slika 27. Komunikacija na fizičkom sloju 33
Slika 28. Komunikacija na podatkovnom sloju 34
Slika 29. Komunikacija na mrežnom sloju 35
Slika 30. Komunikacija na prijenosnom sloju 36
Slika 31. Komunikacija na aplikacijskom sloju 37
Slika 32. Položaj UDP-a u TCP/IP paketu protokola 38

Slika 33. Format korisničkog datagrama 38
Slika 34. Format zaglavlja korisničkog datagrama 39
Slika 35. Enkapsulacija i dekapsulacija poruke 40
Slika 36. Primjer reda u UDP protokolu 41
Slika 37. Multipleksiranje i demultipleksiranje UDP protokola 43
Slika 38. Položaj TCP-a u TCP/IP paketu protokola 44
Slika 39. Isporuka toka bitova 44
Slika 40. Slanje i primanje međuspremnika 45
Slika 41. Prijenos segmenata 46
Slika 42. Format TCP segmenta i njegovog zaglavlja 47
Slika 43. Enkapsulacija segmenta 48
Slika 44. Trostruko rukovanje 48
Slika 45. Prijenos podataka pomoću trostrukog rukovanja 49

Sadržaj tablica

Tablica 1. Tablica poznatih portova korištenih u UDP protokolu 39