Osnovi računarskih mreža

Slavko Gajin

Univerzitet u Beogradu – Elektrotehnički fakultet
Akademska misao
Beograd, 2023.

Dr Slavko Gajin

OSNOVI RAČUNARSKIH MREŽA

Recenzenti
Dr Pavle Vuletić
Dr Dražen Drašković

Izdavači
Univerzitet u Beogradu – Elektrotehnički fakultet
Akademska misao, Beograd

Štampa
Grafoprint, Gornji Milanovac

Tiraž
300 primeraka

ISBN 978-86-7466-974-7

Sadržaj

Uvod

Potreba za računarskim komunikacijama nastala je neposredno nakon pojave prvih računara. U to doba računari su bili znatno drugačiji od današnjih, ne samo po primenjenoj tehnologiji i performansama, već i po svom fizičkom izgledu i veličini. Tadašnji računari su bili izuzetno glomazni, podeljeni u više fizičkih celina (tehničkih ormana) za procesor, memoriju, diskove, spoljne jedinice i njihove kontrolere. Ovi računari su se nazivali mejnfrejm računari (eng. *mainframe*) i zahtevali su instalaciju u posebnim prostorijama sa obezbeđenom klimatizacijom i drugim specifičnim uslovima. Tada izuzetno skupi, njihovo korišćenje je bilo deljeno od strane više korisnika, pristupajući im preko monitora i tastature, koji su mogli da budu u susednoj sobi ili na većim rastojanjima, udaljenim zgradama ili gradovima. Pritisak na tastaturi se binarno kodirao i direktnom fizičkom vezom prenosio do računara, tačnije kontrolera periferije, a odgovor se vraćao po istoj vezi za prikaz na monitoru. Za ovakvu komunikaciju su se koristile direktne veze (kablovi), a na većim rastojanjima postojeće i stalno uspostavljene telefonske veze. Za prenos informacija koristili su se uređaji koji se zovu modemi (eng. *modem*), a koji su binarne podatke pretvarali u zvuk, koji se prenosio preko telefonske veze, a na prijemnoj strani ponovo pretvarao u binarne podatke.

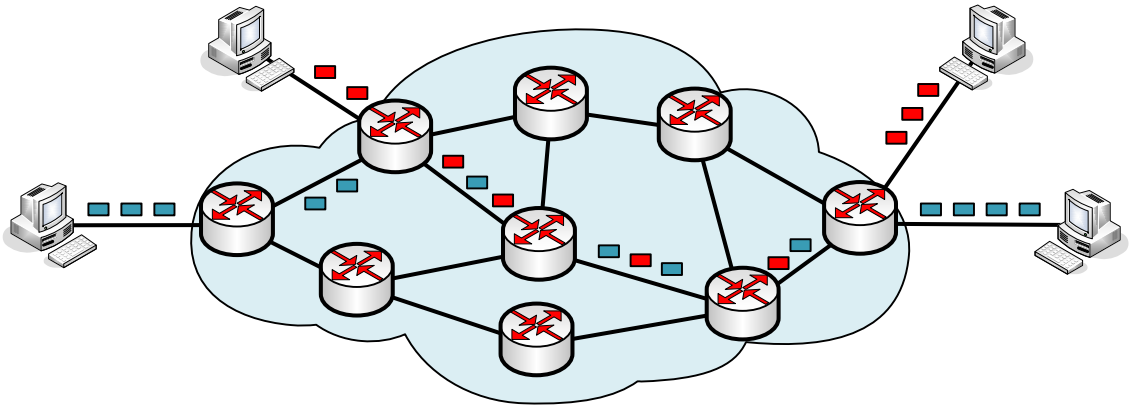
Iako je navedena tehnologija služila za prenos binarnih podataka, to ipak nisu bile računarske mreže, za koje je potrebno da postoji više računara koji međusobno komuniciraju. Računarske mreže se sastoje od fizičke infrastrukture koja povezuje računare i funkcionalne logike koja omogućava ispravnu razmenu podataka. Fizičku infrastrukturu čine fizičke veze (linkovi) i komunikacioni uređaji, dok je funkcionalna logika definisana skupom komunikacionih protokola koji uređuju mašinski prenos podataka. Materijal u ovom udžbeniku se u najvećoj meri odnosi na komunikacione protokole, budući da oni opisuju logiku funkcionisanja računarskih mreža.

1. Razvoj računarskih mreža

Prvobitni mejnfrejm računari su se povezivali tada raspoloživom tehnologijom, koja je podrazumevala modeme povezane preko telefonskih linija, što se nazivalo „komutacija kola“ nastalo iz prevoda engleskog termina *Circuit Switching*. Telefonske veze su mogle da se uspostavljaju po potrebi, ali je češći bio slučaj njihovog permanentnog zakupa samo za povezivanje krajnjih tačaka (eng. *leased line*). Podaci su se prenosili u manjim celinama, tzv. porukama, koje su bile posebno formatirane kako bi ih obe strane u komunikaciji mogle ispravno prepoznati. Prednost ovakvog načina povezivana se ogledala u dostupnosti i povezanosti telefonskih linija na globalnom nivou. Iako je tada predstavljao uobičajeni pristup, ekskluzivno korišćenje veze na celom putu između povezanih uređaja bilo je neekonomično i neskalabilno.

1.1 Paketski prenos podataka

Radikalno novi pristup se javio 1965. godine, koji su nezavisno predložili istraživači Donald Davies i Paul Baran [1]. Umesto direktnog povezivanja krajnjih tačaka, novi pristup je omogućavao ravnopravno povezivanje većeg broja učesnika u zajedničku komunikacionu infrastrukturu, po kojoj se podaci šalju u manjim celinama, tzv. paketima. Komunikaciona infrastruktura je time dobila novu logiku rada, koja je omogućavala da se paketi nezavisno prosleđuju do krajnjih odredišta. Ovu logiku je sprovodila mreža međusobno povezanih komunikacionih uređaja, što je nazvano **svičovanje paketa** (eng. *packet switching*). Prednost se ogledala u tome što je veći broj komunikacionih tokova mogao istovremeno da deli pojedinačne linkove, tako što su se njihovi paketi naizmenično prenosili u kratkim vremenskim intervalima (Slika 1.1). To je za posledicu imalo veliku fleksibilnost povezivanja i skalabilnost broja učesnika. Za veći intenzitet saobraćaja nije bilo potrebe uvoditi nove fizičke linkove, već samo povećavati kapacitet postojećim linkovima. Time je nastao koncept računarskih mreža u modernom smislu.



Slika 1.1. Mreža sa svičovanjem paketa

Dugačak je put od idejnog koncepta do tehničke realizacije, posebno kod ovako radikalnih promena. I sam koncept je naišao na otpor klasičnog telekomunikacionog sektora i postojeće industrije, pa je prva realizacija sprovedena kao istraživačko-eksperimentalni projekat finansiran od Agencije za napredna istraživanja pri Ministarstvu odbrane Sjedinjenih Američkih Država (eng. *Advanced Research Projects Agency, Department of Defence – ARPA*). Mreža

pod nazivom ARPANET je zaživela 1969. godine, povezujući četiri računara univerzitetskih centara u SAD. Pokretačku logiku rada mreže je činio mrežni protokol pod nazivom *Network Control Protocol* (NCP), koji predstavlja prvi protokol koji realizuje princip svičovanja paketa [2]. Narednih godina mreža je unapređivana, uz priključivanje sve većeg broja univerzitetskih i istraživačkih organizacija.

Razvoj mreže je ubrzo prevazišao istraživačke i eksperimentalne okvire, što je zahtevalo koordinirani rad, koji je organizovan u formi radne grupe (*Network Working Group*). Osim upravljačke uloge u mreži, radna grupa je usvajala i tehničku specifikaciju u vidu dokumenta pod nazivom *Request for Comments* (RFC). Inicijalno korišćeni kao neformalni dokumenti preporuka i tehničkih principa, što ukazuje i sam naziv, ubrzo su prerasli u formalne tehničke specifikacije protokola i servisa. Tako je nastala serija referentnih tehničkih dokumenata, koji su numerisani rednim brojevima, i danas aktuelni pod nazivom RFC dokumenti, u žargonu poznato kao „Internet standardi“ [3].

Koncept računarske mreže na bazi svičovanja paketa je bio potvrđen i primenjen u praksi, ali i dalje samo u akademskoj i istraživačkoj zajednici. Slične inicijative su praćene i u drugim zemljama, takođe kao istraživački projekti, poput *Cyclades* mreže u Francuskoj. Pojedini proizvođači računara su takođe realizovali svoje mrežne protokole na bazi svičovanja paketa, poput korporacije IBM koja je razvila SNA protokol (*System Network Architecture*), *Digital Equipment Corporation* (DEC) sa mrežnim protokolom DECnet i slično. Problem je bio što su svi ovi protokoli bili različiti, međusobno nekompatibilni i podržani samo od računara ovih proizvođača. Javila se potreba za standardizacijom koja bi uredila različita tehnološka rešenja i omogućila interoperabilnost opreme različitih proizvođača.

1.2 Standardizacija

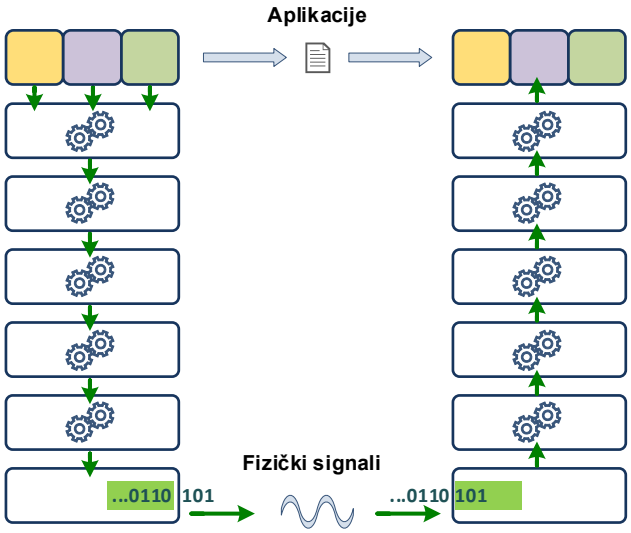
Radna grupa ARPANET mreže je prerasla u Međunarodnu radnu grupu (eng. *International Network Working Group* – INWG), koja se zalagala za promociju nove mrežne tehnologije na bazi svičovanja paketa. Oni su 1975. godine podneli predlog za standardizaciju novog protokola međunarodnoj organizaciji za standarde u telekomunikacijama – CCITT (eng. *International Telegraph and Telephone Consultative Committee*), danas pod nazivom *International Telecommunication Unit – Telecommunication Standardization Sector*, skraćeno ITU-T. Predlog je bio odbijen, pod formalnim obrazloženjem da je netestiran i riskantan, ali suštinski pod uticajem telekomunikacione industrije, koja se trudila da zadrži konzervativni monopolistički pristup na tržištu.

Proces standardizacije je 1977. godine prihvatila Međunarodna organizacija za standarde (eng. *International Organization for Standardization* – ISO), pod nazivom *Open System Interconnection* (OSI). Većina članova Međunarodne radne grupe INWG se pridružila ovom procesu, kao i veliki broj proizvođača opreme. Osnovni principi za donošenje novih standardizovanih protokola bili su:

- otvorenost, što podrazumeva da je komunikacija nezavisna od proizvođača uređaja i
- modularnost, što podrazumeva da je kompleksan problem podeljen u manje celine.

Na ovim principima uspostavljana je nova komunikaciona arhitektura koju uređaji treba da zadovolje, poznata kao OSI referentni model [4]. Ovaj model nastao je iz činjenice da postoji dosta velika razlika između interpretacije aplikativnih podataka na visokom logičkom nivou u odnosu na fizički prenos podataka kroz mrežu koji se odvija putem elektromagnetnih signala. U skladu sa principom modularnosti, proces transformacije aplikativnih podataka u prenosne

signale podeljen je u više nezavisnih celina, tzv. slojeva ili nivoa (eng. *layer*). Svaki sloj ima strogo definisanu ulogu i način komunikacije sa višim i nižim slojem, posredstvom interfejsa koji se naziva *Service Access Point* (SAP). Tom prilikom naziv „servis“ ukazuje da jedan sloj pruža usluge drugom sloju. Podaci se pri slanju između slojeva prenose naniže, dok se na prijemu sprovodi obrnuti proces rekonstrukcije podataka, od nižih slojeva do odredišne aplikacije, što ilustruje Slika 1.2.



Slika 1.2. Modularna arhitektura podeljena na slojeve

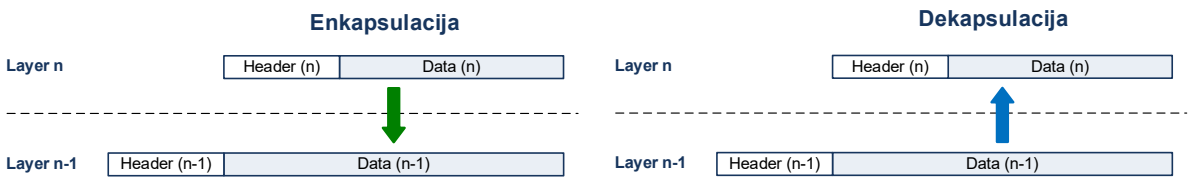
Konkretno, OSI referentni model uvodi sedam slojeva (nivoa), i to:

- L7, Aplikativni sloj (eng. *Layer 7*, *Application Layer*)
- L6, Prezentacioni sloj (eng. *Layer 6, Presentation Layer*)
- L5, Sloj sesije (eng. *Layer 5, Session Layer*)
- L4, Transportni sloj (eng. *Layer 4, Transport Layer*)
- L3, Mrežni sloj (eng. *Layer 3, Network Layer*)
- L2, Sloj veze podataka (eng. *Layer 2, Data-link Layer*)
- L1, Fizički sloj (eng. *Layer 1, Physical Layer*).

Na aplikativnom sloju uparene aplikacije međusobno logički komuniciraju razmenom struktuiranih podataka u vidu posebno formatiranih poruka. Spuštajući se naniže pri slanju, naredni niži sloj u celini preuzima ove poruke i posmatra ih kao monolitne podatke, bez poznavanja njihove strukture. Od ovih podataka kreira se nova poruka, postavljajući podatke sa višeg sloja u deo koji se naziva **telo poruke** (eng. *body*), dok se na početku poruke dodaje kontrolni podaci iz tekućeg sloja, u delu koji se naziva **zaglavlje** (eng. *header*). Ovako kreiranu poruku tekući sloj predaje nižem sloju, koji na svom nivou sprovodi isti postupak – preuzima poruku i postavlja je u telo svoje poruke, tzv. **enkapsulacija** (eng. *encapsulation*). Poruka isporučena fizičkom sloju se serijalizuje u niz bita, koji se kodiraju i transformišu u prenosne signale.

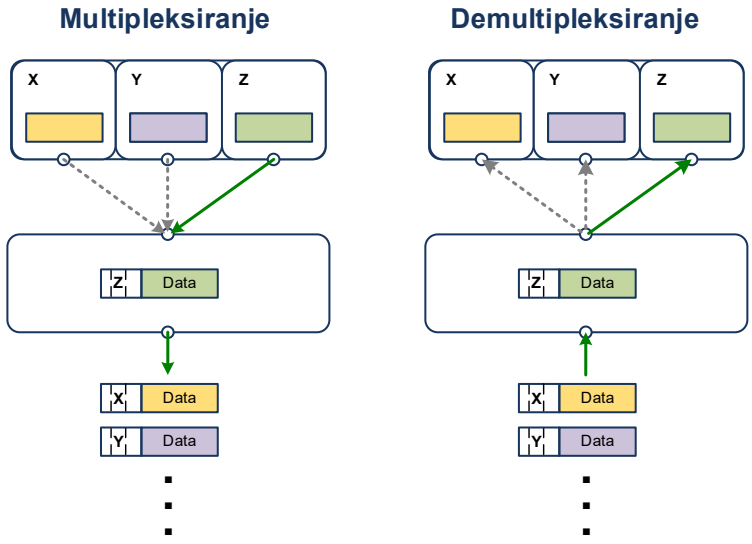
Na prijemu se u svakom sloju sprovodi inverzan proces postepenog raspakivanja podataka, tzv. **dekapsulacije** (eng. *decapsulation*). Svaki sloj dobija poruku u formatu koji je kreirao i poslao njemu upareni sloj na istom nivou. Iz zaglavlja se tumače poslati kontrolni podaci, i ako

je sve ispravno, telo poruke se u celini predaje višem sloju. Ovaj postupak se sprovodi sve do aplikativnog sloja, gde se rekonstruišu originalno poslati aplikativni podaci (Slika 1.3).



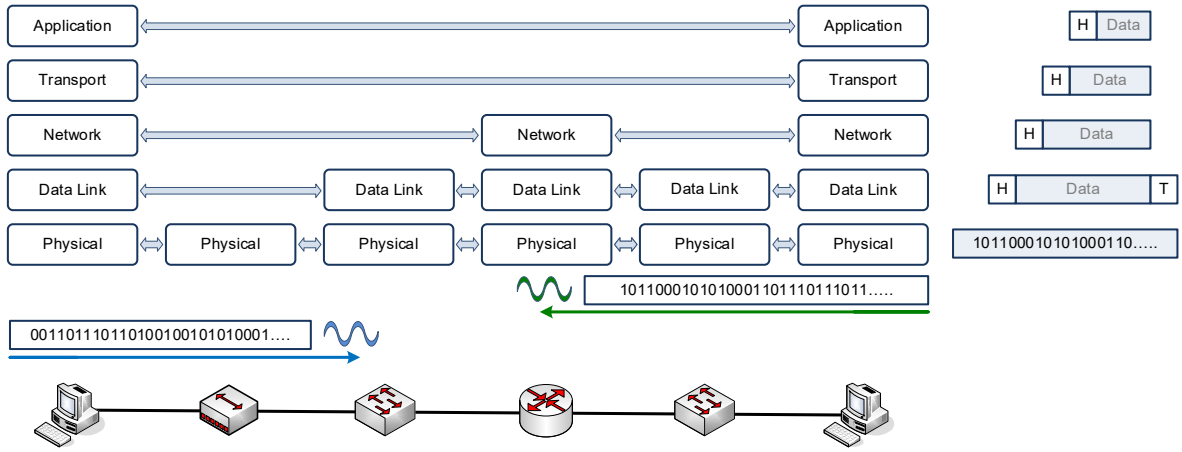
Slika 1.3. Proces enakapsulacije pri slanju i dekapsulacije pri prijemu podataka

Kao što postoji više aplikacija koje nezavisno funkcionišu na aplikativnom sloju, tako i na svakom nižem sloju mogu istovremeno funkcionisati više različitih i nezavisnih entiteta, odnosno implementacija različitih protokola. Poruke svih ovih protokola se na isti način preuzimaju i tretiraju od strane nižeg sloja, ali se one moraju posebno označiti kako bi se na prijemnoj strani adekvatno prepoznali i isporučili uparenom protokolu ili aplikaciji. Ovaj proces prenosa podataka iz različitih izvora na višem sloju kroz zajednički komunikacioni kanal koji obezbeđuje niži sloj, naziva se **mulitipleksiranje** (eng. *multiplexing*), dok se obrnuti proces na prijemnoj strani naziva **demultipleksiranje** (eng. *demultiplexing*), što ilustruje Slika 1.4.



Slika 1.4. Proces multipleksiranja pri slanju i demultipleksiranja pri prijemu podataka

Imajući ovo u vidu, može se smatrati da se na logičkom nivou sprovodi komunikacija putem poruka između slojeva istog nivoa na udaljenim uređajima. Podaci se na krajnjim uređajima fizički prenose između slojeva „po vertikali“, dok se na logičkom nivou komunikacija odvija između slojeva istog nivoa. Na ovom mestu treba istaći da se na putu između učesnika u mreži može javiti više međutačaka, odnosno komunikacioni uređaji koji implementiraju slojeve na nižim nivoima, prihvataju poruke i prosleđuju ih dalje prema odredištu, što ilustruje Slika 1.5.

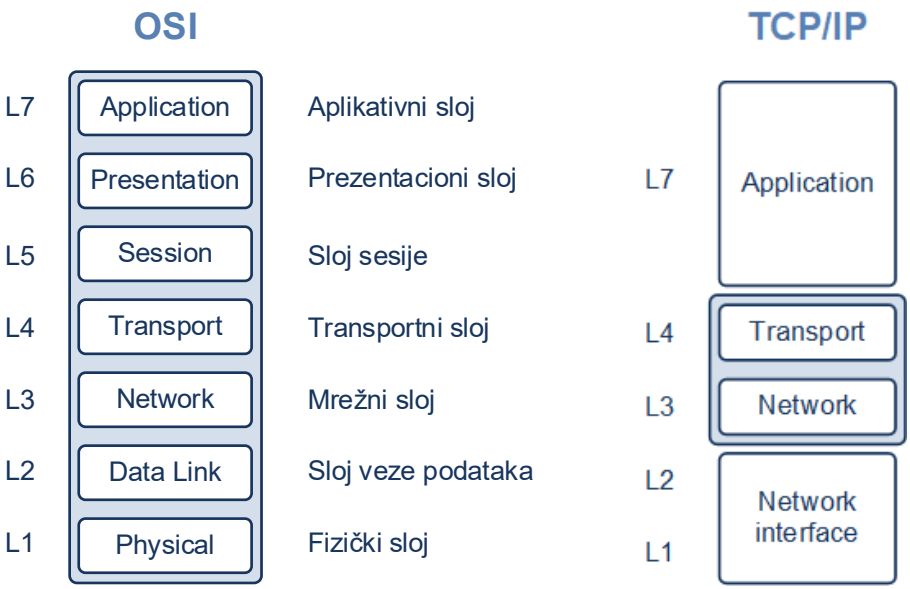


Slika 1.5. Fizički prenos podataka i logička komunikacija između uparenih slojeva

Nakon usvajanja OSI referentnog modela, proces standardizacije se fokusirao na pojedinačne protokole, pre svega na slojevima nižih nivoa, koji implementiraju mrežne funkcionalnosti i na krajnjim uređajima i na komunikacionim uređajima. Ovaj proces je ipak bio dosta spor, opterećen birokratskim procedurama, velikim brojem učesnika, često sa suprotstavljenim stavovima, ali i preambicioznim zahtevima. Protokoli koji su se razvili i usvojili bili su dosta robusni, komplikovani, skupi za realizaciju.

Nasuprot tome, razvoj u okviru ARPANET mreže je bio manje formalan, mnogo agilniji i fleksibilniji, uz efikasno testiranje i demonstraciju koncepta u praksi. To je dovelo do arhitekture koja je takođe bila bazirana na OSI referentnom modulu, ali u pojednostavljenoj varijanti, sa posebnim fokusom samo na mrežni sloj (L3) i transportni sloj (L4). Zbog velike implementacione zavisnosti dva najniža sloja, fizičkog i sloja veze podataka, oni se obično tretiraju objedinjeno, a u to vreme su bili nezavisno razvijani od strane različitih proizvođača, a standardizovani od strane organizacije *Institute of Electrical and Electronics Engineers*, daleko popularnije po skraćenici IEEE (izgovara se „aj-tripl-i“, eng. „*I-triple-E*“). Sa druge strane, tri najviša sloja suštinski su bliži softverskim implementacijama nego mrežnim tehnologijama, pa su objedinjeni sa aplikativnim slojem.

Rezultat ovakvog pristupa je bio jedan jednostavan protokol na mrežnom nivou, pod nazivom ***Internet Protocol*** (**IP**), dva protokola na transportnom nivou, i to ***Transmission Control Protocol*** (**TCP**) i znatno jednostavnija varijanta pod nazivom *User Datagram Protocol* (UDP), sa još nekoliko pratećih pomoćnih protokola koji su upotpunjavali pojedine nedostajuće funkcije. Celokupni skup ovih protokola poznat je pod nazivom **TCP/IP**.



Slika 1.6. OSI i TCP/IP arhitektura podele na slojeve

TCP/IP familija protokola je predstavljala ono što se danas u razvoju softvera po agilnim metodologijama zove „minimalno održivo rešenje“ (*Minimum Viable Product* – MVP). Rešenje doneto i realizovano u minimalnom, ali funkcionalnom obliku, i primenjeno u praksi, ali samo u okviru istraživačke i univerzitetske zajednice, gde je ostvarila veliku popularnost. Za to vreme, telekomunikaciona i računarska industrija na strani proizvođača i ostali sektori na strani korisnika, sledili su konvencionalni pristup vođen ISO standardizacijom. Bilo je jasno da je tehnologija na prekretnici, što je označeno kao rat između OSI i TCP/IP tehnologija. Ovaj „tehnološki rat“ je bio nepravedan[1] – OSI pristup imao podršku svih relevantnih faktora (vlada, kapitala, industrije, komercijalnih korisnika), ali je TCP/IP model, iako podržan samo od akademske zajednice, imao ključnu prednost – bio je jednostavan, a uz to je i radio.

Bitni datumi koji ilustruju razvoj događaja su sledeći:

- **1.1.1983**: Ministarstvo odbrane SAD (eng. *U.S. Department of Defence*), donosi odluku o prelasku sa *Network Control Protocol* na TCP/IP protokol u okviru ARPANET mreže.
- **Maj 1983**: ISO OSI referentni model je zvanično objavljen kao standard [4].
- **1985**: Nacionalni istraživački komitet SAD (eng. *U.S. National Research Council*) preporučuje da ARPANET postepeno pređe sa TCP/IP na OSI standard.
- **1988**: Ministarstvo trgovine SAD (eng. *U.S. Department of Commerce*) naložio je vladinim ustanovama prelazak na OSI standard do avgusta 1990. godine.
- **Kraj 80-tih**: Naznake neostvarenih očekivanja u OSI redovima
  - Brian Carpenter, “Is OSI Too Late?” [5]
  - Louis Pouzin, „Ten Years of OSI — Maturity or Infancy?” [6]
- **1990**: ARPANET projekat je zvanično završen, ali je mreža nastavila da funkcioniše i dalje se razvija finansirana od strane Nacionalne naučne fondacije SAD (eng. *U.S. National*

[1] A koji rat nije nepravedan?

Science Foundation – NSF) kroz program pod nazivom *National Science Foundation Network* (NSFNET). Mreža je prihvaćena pod novim neformalnim imenom – Internet.

- **1991:** Nastanak veb servisa, *World Wide Web* - WWW, Tim Berners-Lee.
- **1992:** NSF dozvoljava povezivanje komercijalnih korisnika, što je dovelo do formiranja Interneta u modernom smislu.

Time je ISO OSI definitivno izgubio rat od strane TCP/IP protokola, ali se i u narednim godinama smatralo da je TCP/IP suviše jednostavan za tada narastajuće multimedijalne potrebe. Bilo je pokušaja razvoja novih naprednih protokola, a upečatljiv je primer dugo razvijanog, ali brzo zaboravljenog, ATM protokola (eng. *Asynchronous Transfer Mode*). TCP/IP se nije mnogo menjao, a nedostajuće funkcije su upotpunjene dodatnim protokolima, kao što je sigurnost i podrška za video saobraćaj u realnom vremenu. Internet mreža se eksplozivno razvijala i u novi milenijum je ušla sasvim zrela, čak punoletna, budući da je za „rođenje“ Interneta (eng. „*the birth of the Internet*”) izabran 1.1.1983. godine kao datum zvaničnog uvođenja TCP/IP protokola u tadašnju ARPANET mrežu.

1.3 Predmet udžbenika

Predmet ovog udžbenika se odnosi na opis tehnologija i protokola, prateći slojeve od nižih ka višim, podeljeno u tri celine. Prvi deo je posvećen sloju podataka veze (eng. *Data link layer*), koji je vezan za fizički sloj kroz funkciju pristupa medijumu. Detaljno se opisuje danas dominantna tehnologija ove vrste koja se primarno koristi za lokalno povezivanje manjeg broja korisnika, tzv. Eternet mreže (eng. *Ethernet*). Materija prati postepeni razvoj tehnologije, od ranih potreba i praktičnih realizacija, opisom postepenih unapređenja i naknadno razvijanih protokola i tehničkih poboljšanja do oblika u kome se koristi danas.

Drugi deo udžbenika u potpunosti je posvećen Internet protokolu (IP) koji realizuje funkcije mrežnog sloja (L3), zajedno sa pratećim protokolima koji predstavljaju jezgro rada savremenih računarskih mreža, uključujući i globalnu Internet mrežu.

Treći i poslednji deo je posvećen višim slojevima, odnosno transportnom sloju i aplikativnom sloju. Detaljno su opisani TCP i UDP transportni protokoli, koji realizuju mrežne komunikacije između krajnjih uređaja. Na aplikativnom nivou se prikazuje princip rada aplikacija u klijent-server režimu, sa primerima najčešćih aplikativnih servisa. Zbog svoje važnosti za funkcionisanje Interneta, posebno je izdvojen servis obezbeđivanja imenovanja (eng. *Domain Name System*, skr. DNS), čiji se princip rada detaljno opisuje.

Prvi deo:

Niži slojevi

2. Kontrola pristupa medijumu

Podaci koji se prenose u računarskim mrežama predstavljaju niz bajtova određene veličine, tzv. paket, koji ima svoju definisanu strukturu kako bi se na prijemu podaci mogli ispravno rekonstruisati. Paket se zatim serijalizuje u nestruktuirani niz bitova, što i dalje predstavlja logičku interpretaciju podataka u obliku diskretnih vrednosti. Da bi se ove apstraktne informacije (nule i jedinice) prenele između učesnika u komunikaciji, one se moraju transformisati u određeni fizički signal koji se prenosi kroz njemu odgovarajući fizički medijum. To mogu biti nivoi električnog napona u bakarnim provodnicima, radio talasi u određenom opsegu frekvencija u bežičnim komunikacijama, svetlost u optičkim kablovima i slično. Shodno tome, način interpretacije jedinica i nula je različit. U optičkim komunikacijama se bitovi predstavljaju prisustvom i odsustvo svetlosti (fotona) u određenom (jako kratkom) vremenskom intervalu. U slučaju elektromagnetnih signala, što predstavljaju i radio-talasi u bežičnim komunikacijama i električni signali u bakarnim provodnicima, radi efikasnosti i prenosa veće količine informacija, više bitova se koduje u određenu vrednost, koja se pretvara u jačinu (amplitudu) signala. Tom prilikom, medijum se zauzima na određeni vremenski interval, ne samo da se svi bitovi iz paketa pretvore u vremenski promenljive signale, već i da ovi signali stignu i do najudaljenijih učesnika. Iako se signali prenose jako velikim brzinama, one su ipak konačne: elektromagnetni talasi radio komunikacija se kreću brzinom svetlosti, fotoni u optičkim kablovima nešto sporije (jer se ne prenose kroz vakuum), dok se elektro-magnetni talasi u bakarnim provodnicima prenose brzinom koja iznosi oko 2/3 brzine svetlosti. Naravno, svaki ovakav signal mora da se na prijemnoj strani jednoznačno transformiše u originalni izvorni niz bitova, da se prepozna struktura bajtova, i na osnovu toga da se ispravno izdvoje podaci koji se prenose.

Komunikacija u računarskim mrežama se sprovodi između većeg broja učesnika i svi oni pristupaju komunikacionom medijumu radi slanja i prijema paketa. Ako je medijum na ovaj način deljen od strane više učesnika, problem nastaje ako više njih šalju svoje paketa u istom ili bliskom vremenskom trenutku. Tada će paketi na medijumu da se „preklope“ u vremenu, odnosno njihovi signali će da se pomešaju (interferiraju) i međusobno unište tako da se informacije iz originalnih paketa ne mogu više rekonstruisati. Ova pojava se naziva kolizija.

Komunikacione tehnologije moraju da definišu sve detalje transformacije podataka u signale, njihov fizički prenos po medijumu i rekonstrukciju u originalne podatke. U opštem slučaju medijum se deli između više učesnika, a način njihovog dogovora oko pristupa i deljanja medijuma, uključujući i način tretiranja kolizije, naziva se kontrola pristupa medijumu (eng. *Media Access Control,* skr. *MAC*).

Svaki medijum i tehnologije prenosa podataka, ima svoj maksimalni kapacitet, odnosno propusni opseg (eng. *bandwidth*) izraženo u jedinici „biti u sekundi“ (eng. *bits per second*, skr. *bps*). Prilikom osmišljavanja određene komunikacione tehnologije postavlja se sledeće pitanje: kako na optimalan način *N* učesnika mogu da dele medijum kapaciteta *B* bps? U idealnom slučaju, ako samo jedan učesnik ima potrebu da šalje podatke, onda on treba da iskoristi maksimalni kapacitet od B bps, dok u slučaju više (N) učesnika kapacitet treba da se ravnomerno podeli da svaki učesnik ostvari protok od B/N bps. Takođe, potrebno je odrediti i način kako više učesnika deli medijum. Ipak, kao što ćemo u kasnijim primerima videti, možda i najbitniji zahtev je da tehnološko rešenje bude što jednostavnije za realizaciju, kako bi bilo ekonomski prihvatljivo za masovno korišćenje i dugoročni opstanak tehnologije.

Osnovne tehnike kontrole pristupa medijumu su sledeće:

- Podela na kanale (eng. *Channel Partitioning*)
- Pristup sa dodelom dozvole (eng. *Taking Turns*)
- Slučajni pristup (eng. *Random Access*)

2.1 Podela na kanale

Najjednostavnije rešenje kontrole pristupa je podela medijuma na više nezavisnih delova, tzv. kanala (eng. *Channel Partitioning*), i njihova dodela pojedinačnim učesnicima na ekskluzivno korišćenje. Frekvencijski domen za prenos radio signala se može podeliti na više frekvencijskih opsega koji nezavisno i istovremeno prenose signale (eng. *FDMA - Frequency Division Multiple Access*). Ovaj princip se već decenijama koristi pri emitovanju klasičnih radio i TV signala. U slučaju jedinstvenog fizičkog medijuma (npr. bakarnih provodnika), korišćenje medijuma se može podeliti u vremenu (eng. *TDMA - Time Division Multiple Access*). Tom prilikom svaki učesnik dobija fiksni vremenski interval (tzv. slot) za privremeno korišćenje medijuma po unapred utvrđenom redosledu.

Dva nedostatka su zajednička za obe tehnike. Prvo, broj učesnika je ograničen na maksimalni broj kanala koji je unapred fiksiran. Drugo, svaki učesnik može da iskoristi protok koji odgovara jednom kanalu (*B/N bps*), a ne i celom prenosnom medijumu (*B bps*). Ovo za posledicu ima da se medijum optimalno koristi jedino ako svi učesnici sprovode prenos podataka. U suprotnom i češćem slučaju, ako neko od učesnika nema podatke za slanje, njegov kanal tada ostaje neiskorišćen, što efektivno smanjuje iskorišćenost celog medijuma.

2.2 Pristup sa dodelom dozvole

Pristup medijumu sa dodelom dozvole podrazumeva dinamičko određivanje ko od učesnika i kada može da koristi medijum. Ovi se sprovodi kroz međusobnu komunikaciju učesnika, što može da bude centralno ili distribuirano.

U slučaju centralizovane kontrole, jedan učesnik, tzv. *master*, ima centralu ulogu tako što „proziva“ (eng. *polling*) ostale učesnike (eng. *slaves*) i daje im dozvolu korišćenja medijuma za slanje paketa. Ovaj mehanizam prozivanja zahteva posebnu kontrolnu komunikaciju između *master* i *slave* učesnika, koja je daleko od jednostavnog. Prvo, *master* mora da zna za sve ostale učesnike u mreži, koji zbog toga na neki način moraju da se prijave *master* uređaju. Dalje, *master* ne zna unapred da li pojedinačni učesnici imaju podatke za slanje, ali im naizmenično dodeljuje medijum na korišćenje. Ako uređaj poseduje spremne podatke za slanje, on ima na raspolaganju određeni vremenski interval za korišćenje medijuma, nakon čega ga oslobađa. U suprotnom slučaju, ako uređaj kome je dodeljen medijum u tom trenutku nema spremne podatke za slanje, on obaveštava *master* uređaj da odustaje od korišćenja medijuma. Medijum se tada oslobađa i po istom principu dodeljuje drugom učesniku, ali je određeni vremenski period ipak izgubljen, što smanjuje ukupnu efikasnost prenosa podataka. Osim ovog problema, kao i komplikovane realizacije i neravnomernog opterećenja *master* uređaja u odnosnu na ostale uređaje, možda i najveći praktičan problem je što će prestanak rada *master* uređaja da izazove prekid svih komunikacija u mreži. To ga čini tzv. jedinstvenom tačkom otkaza (eng. *single point of failure*). Ovo se može rešiti mehanizmom izbora novog master uređaja, što dodatno usložnjava logiku rada i njenu implementaciju na uređajima, što konsekventno dovodi do povećanja krajnje cene primene tehnologije.

Kod distribuirane dodele medijuma svi učesnici su ravnopravni i međusobno komuniciraju radi dogovora po pitanju korišćenja medijuma. Najpoznatija tehnologija ove vrste se bazira na mehanizmu kruženja posebnog kontrolnog paketa, tzv. žetona (eng. *Token Passing*). Ovaj paket sa kontrolnim podacima se prenosi u kružnom redosledu između svih učesnika, i preko njega se razmenjuju određene kontrolne informacije o korišćenju medijuma. Kružna komunikacija se obično rešava fizičkim povezivanjem učesnika u prstenastu topologiju (eng. *ring*). Uprošćeno govoreći, svaki učesnik kada dobije žeton, može da šalje podatke ako ih ima, nakon čega predaje žeton narednom učesniku čime mu se prepušta mogućnost korišćenja medijuma. Kao i žeton, paketi sa podacima se takođe prenose u kružnom redosledu do uređaja kome su namenjeni.

Najpoznatija tehnologija ove vrste je bio tzv. *TokenRing*, realizovana od strane proizvođača IBM. Ova tehnologija se dosta koristila krajem osamdesetih i početkom devedesetih godina dvadesetog veka, ali daleko od toga da je bila jednostavna i jeftina[2]. Budući da je mreža bazirana na prstenastoj topologiji, prekid rada jednog uređaja potencijalno može da prekine rad cele mreže. Zato su se računari na *TokenRing* mrežu povezivali preko posebnih pristupnih uređaja, tzv. *Media Access Unit* (skr. MAU). Prestankom rada povezanog računara, MAU uređaj je u bio u stanju da „prespoji“ prekinutu prstenastu vezu i omogući kontinuitet rada ostalih uređaja. Iako je *TokenRing* mreža omogućavala povezivanje i bakarnim i optičkim kablovima, brzine prenosa podataka su bile fiksne, ograničene na svega 4 Mbps, sa mogućnošću povećanja na 16 Mbps. Tehnologija se pokazala neskalabilna po pitanju povećanja brzine prenosa, što je uskoro postalo i najveće ograničenje.

U to vreme novija generacija tehnologije na bazi mehanizma kruženja žetona je bila tzv. FDDI (eng. *Fiber Distributed Data Interface*). Kao što ime ukazuje, prvenstveno je bila namenjena za prenos po optičkim vlaknima (eng. *fiber optics*), takođe po prstenastoj topologiji, ali kruženjem podataka u oba smera brzinama od 100 Mpbs. Jedna od specifičnosti je bila mogućnost otpornosti na fizički prekid prstena, tako što se ostatak dvosmernog prstena rekonfiguriše u jednosmerni prsten. Iako se od FDDI tehnologije dosta očekivalo, njen životni vek je bio još kraći. Osim ove, u to vreme egzotične osobine rekonfiguracije, malo šta je od FDDI tehnologije ostalo vredno za pamćenje.

2.3 Slučajni pristup

Metod slučajnog pristupa medijumu (eng. *Random Access*) je nastao upravo iz potrebe da se na što jednostavniji način omogući komunikacija većeg broja učesnika, bez složenog mehanizma dogovora oko korišćenja medijuma. Da bi sagledali kontekst u kome je nastala ova metoda, kao i njene specifičnosti, vratimo se još ranije u prošlost. Krajem šezdesetih godina prošlog veka Univerzitet na Havajima je imao potrebu da poveže svoje objekte koji su se nalazili na različitim ostrvima u radijusu od oko 300 km. Tačnije, postojao je samo jedan centralni računar, a bila je potreba da se sa njime omogući komunikacija velikog broja udaljenih pasivnih terminala (monitor i tastatura). Za komunikacioni medijum su izabrane radio veze i to sa dve frekvencije: jedna za komunikaciju u smeru od centralnog računara prema terminalima (eng. *downstream*), a druga u obrnutom smeru, od terminala do centralnog računara (eng. *upstream*). U oba slučaja se radi o deljenom medijumu (izabrani opseg frekvencija radio talasa), ali u smeru od centra ka periferiji samo jedan učesnik (centralni računar) može da šalje podatke, pa nema potrebe za posebnim mehanizmom dodele medijuma. Za razliku od toga, u

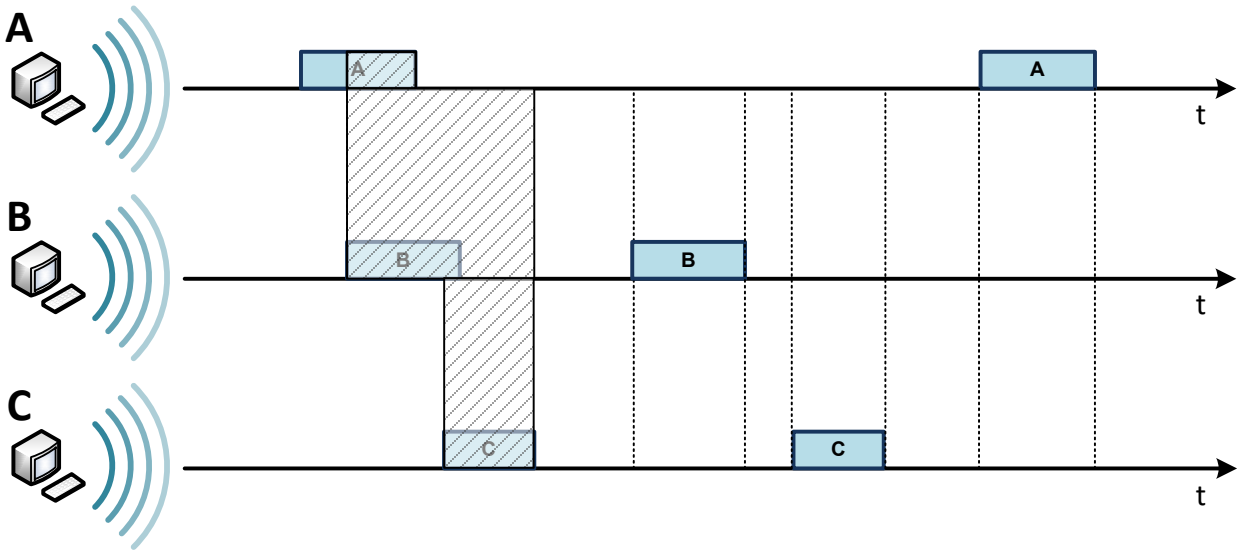
[2] IBM je u to vreme bio poznat po izuzetno kvalitetnim, ali skupim rešenjima.

suprotnom smeru više terminala može da šalje podatke ne deljeni medijum, pa je bilo potrebno rešiti problem pristupa medijumu. Podela na kanale po frekvenciji ili vremenu je odbačena, jer se predviđao porast broja učesnika, kao i potreba za prenosom sve veće količine podataka. Upravo zbog očekivanog velikog broja krajnjih terminala i tada skupih hardverskih resursa[3], posebno bitan uslov je bila jednostavnost same tehnologije i niska cena njene implementacije.

Uz ova ograničenja, nametnulo se rešenje koje je bilo neuobičajeno, ali ipak inventivno, što će se kasnije pokazati i kao revolucionarno – svakome se dozvoljava pristup medijumu u bilo kom trenutku kada postoje raspoloživi podaci za slanje, kao da nema drugih terminala koji takođe mogu da šalju svoje podatke. Budući da na ovaj način može nastati kolizija, za svaki paket koji se šalje od terminala očekivala se potvrda o njegovom uspešnom prijemu od strane centralnog računara. U slučaju da ovaj paket potvrde izostane, smatra se da je došlo do kolizije i paket će ponovo da se pošalje. Činjenica da je došlo do kolizije ukazuje na to da je i paket nekog drugog učesnika takođe uništen i da i on mora ponovo da se pošalje. Da bi se u ovakvim slučajevima izbegla ponovna kolizija neposredno nakon oslobađanja medijuma, pre ponovnog slanja istog paketa potrebno je sačekati i dodatni vremenski interval promenljivog trajanja, a koji se određuje na slučajan način, tako što se generiše pseudo-slučajan broj koji se množi fiksnim jediničnim intervalom. Na taj način se ipak ne garantuje uspešan prenos paketa, ali se značajno smanjuje verovatnoća ponovnog nastanka kolizije. Štaviše, prilikom svakog slanja se može proceniti verovatnoća nastanka kolizije u zavisnosti od opterećenja medijuma, ali se kroz određeni broj pokušaja očekuje da se paket ipak uspešno pošalje.

Slika 2.1 ilustruje primer slanja paketa od strane tri uređaja prema ovom principu. Čak iako su paketi manjim delom preklopljeni u vremenu nastaće kolizija i svi paketi će da budu u celini uništeni. Ukupno vreme od početka slanja prvog paketa do kraja slanja trećeg paketa će da bude izgubljeno, kao i vreme dodatnog čekanja na retransmisiju, što sve zajedno smanjuje rezultujuću efektivnost prenosa podataka. Redosled slanja paketa takođe nije određen, pa može da se desi da uređaji koji su inicijalno kasnije započeli slanje podataka, u narednim pokušajima posle kolizije to mogu uspešno da završe i pre drugih uređaja.

[3] Cena memorijskog bafera od 88 bajta za prijem i slanje paketa je tada koštala oko 300 dolara [7].



Slika 2.1. Primer kolizije i retransmisije pri slanju paketa od strane tri različita uređaja

Implementacija navedenog protokola podrazumevala je izradu novih uređaja preko kojih su se terminale povezivali na radio mrežu, a koji su nazvani *Terminal Control Unit* (skr. TCU). U junu mesecu 1971. godine prvi terminal je povezan na centralni računar brzinom od 9.600 kbps (Slika 2.2). Povezivanjem većeg broja terminala koje je usledilo nakon toga, koncept je u praksi uspešno verifikovan, a cela mreža je nazvana ALOHAnet.

Potrebno je imati u vidu da su tada terminali prenosili samo kratke tekstualne poruke zapakovane u male pakete, pa i pored, za današnja merila, skromnih brzina prenosa i mogućnosti nastanka kolizije, ovaj vid komunikacije je uspešno funkcionisao i za veći broj povezanih uređaja. Novi princip pristupa medijumu nazvan je slučajni pristup (*Random Access*), a ovaj konkretan metod je nazvan *Pure ALOHA*.

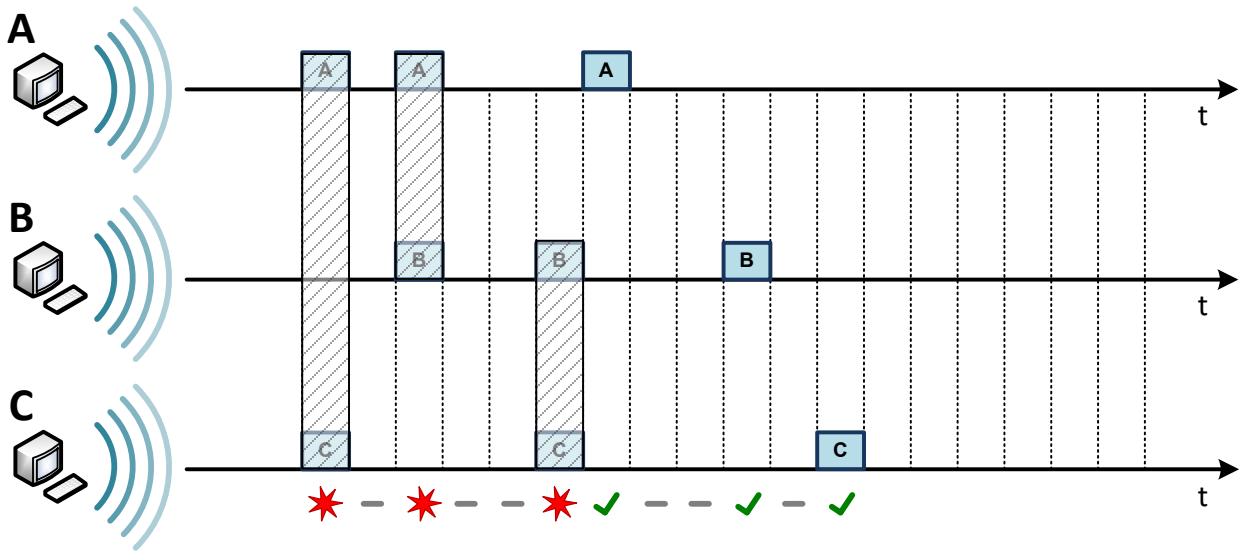
Slika 2.2. ALOHAnet, korisnički terminal povezan na TCU (eng. Terminal Control Unit), Havaji, 1971. godine [7].

Ipak, intuicija nam govori da povećanjem opterećenja mreže, kroz veći broj učesnika i njihovu intenzivniju komunikaciju, raste verovatnoća nastanka kolizije i retransmisije paketa. Time se smanjuje efikasnost prenosa, mereno kao količina uspešno prenetih bita u odnosu na raspoloživi kapacitet. To potvrđuje i matematički proračun, koji predviđa da sa povećavanjem opterećenja raste iskorišćenje mreže, ali samo do određenog nivoa, kada se ulazi u zasićenje, a koje iznosi svega 18.4% ukupnog kapaciteta. Daljim povećavanjem opterećenja, javljaju se

učestalije kolizije, čime se iskorišćenost naglo smanjuje. Drugim rečima, ako veći broj korisnika koristi ukupni kapacitet od 9.600 kbps, može se očekivati da se postiže ukupno iskorišćenje od svega 1.766 kbps. Budući da je broj povezanih terminala u ALOHAnet mreži bio sve veći, ovo se ubrzo pokazalo kao ozbiljno ograničenje.

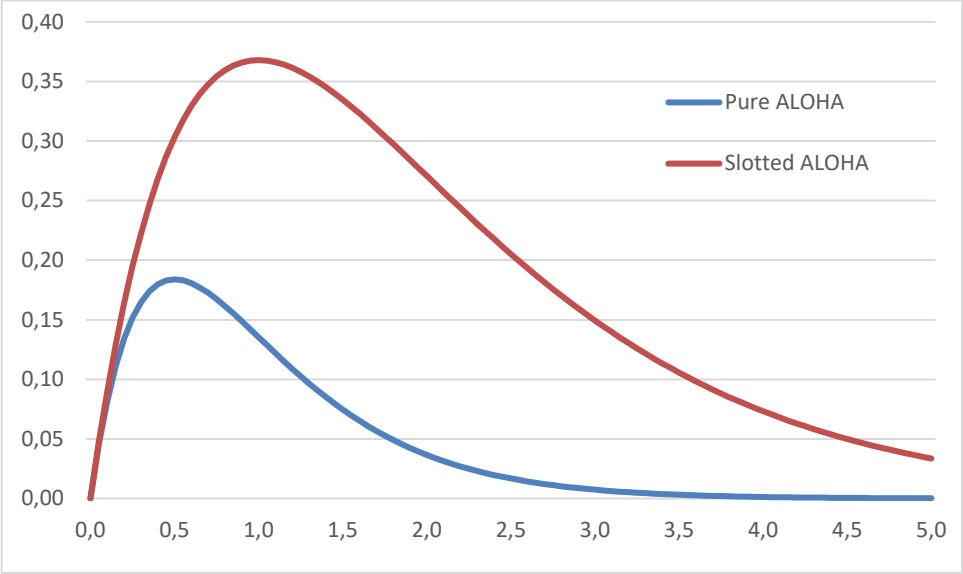
Posebno nepovoljna okolnost kod *Pure ALOHA* metode se odnosila na mogućnost „nadovezivanja“ kolizije od strane naknadno poslatih paketa, kao što ilustruje Slika 2.1. Da bi se trajanje kolizije kontrolisalo i ograničilo na fiksni interval, predloženo je da paketi budu ograničene veličine i da se mogu slati samo u određenim vremenskim periodima, tzv. vremenskim slotovima (eng. *time slot*). Tokom trajanja jednog slota, ako postoji samo jedan paket spreman za slanje, on će se uspešno preneti. U suprotnom slučaju, ako postoji više paketa, doći će do kolizije, ali će ona trajati najduže jedan slot vremena, bez obzira koliko paketa u tome učestvuje.

Ovaj metod je nazvan *Slotted ALOHA* (Slika 2.3), a matematički proračun pokazuje da se time duplira maksimalni protok u odnosu na *Pure ALOHA* prenos, čime se postiže maksimalno iskorišćenje od 36.6% ukupnog kapaciteta (Slika 2.4). Udvostručavanje protoka je predstavljalo značajno unapređenje, ali ipak po cenu nešto komplikovanijeg dizajna – bilo je potrebno da svi uređaji budu vremenski sinhronizovani, kako bi znali tačne trenutke početka slotova. Nakon implementacije ovog rešenja, *Slotted ALOHA* je uspešno puštena u rad 1973. godine.



Slika 2.3. Princip slanja paketa kod Slotted ALOHA mreže

Osim što je u ALOHAnet mreži prvi put primenjen metod slučajnog pristupa medijumu, ona će ostati upamćena i kao prva bežična mreža paketskog prenosa podataka, kao preteča današnjih bežičnih mreža (eng. *wireless*). Još značajnije, ALOHAnet je inspirisala razvoj nove tehnologije prenosa u lokalnim računarskim mrežama, tako što je princip slučajnog pristupa medijumu primenjen na bakarnim provodnicima koji su omogućili daleko veće brzine prenosa.

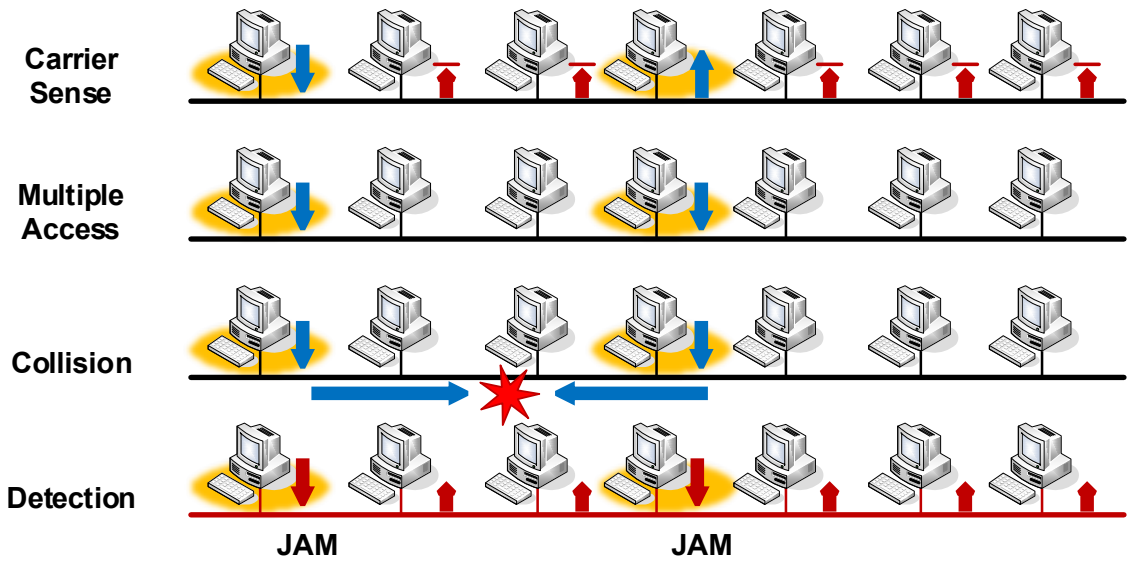


Slika 2.4. Iskorišćenost propusnog opsega za Pure ALOHA i Slotted ALOHA metod slučajnog pristupa medijumu

2.4 CSMA/CD

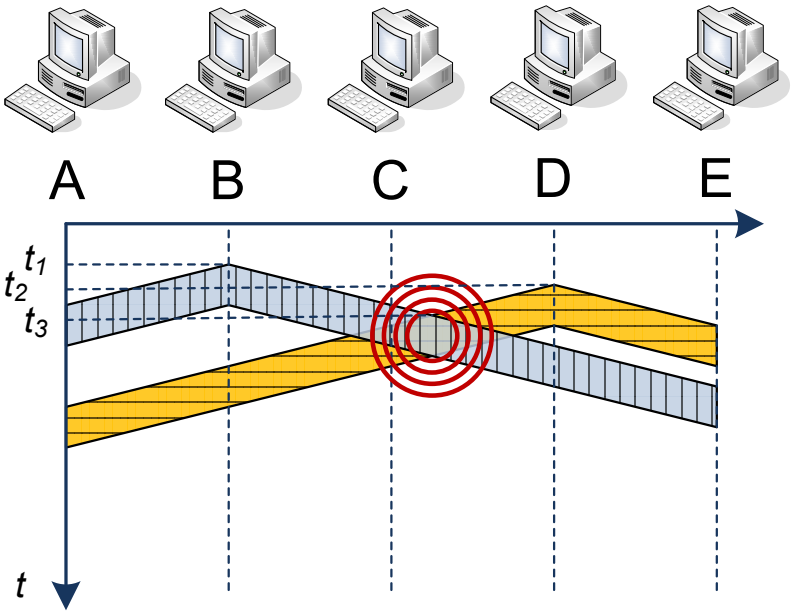
U tehnologijama koje se primenjuju u računarskim mrežama, pa tako i u ovoj knjizi, dosta se koriste različite skraćenice i akronimi. Neki od njih su zvučni i jednostavni za izgovaranja, poput do sada spominjanih „LAN“ i „MAC“ pojmova, dok su mnoge skraćenice čak i nezgrapne. Ipak, učestalim korišćenjem u tehničkoj literaturi i inženjerskoj praksi, ovi pojmovi se vremenom odomaće čak i u neengleskom govornom području. To je slučaj i sa pojmom CSMA/CD, koji se u ovom obliku ipak lakše piše i izgovara nego puni naziv tehnologije – „*Carrier Sense Multiple Access with Collision Detection*“.

CSMA/CD metod pristupa medijumu je nastao adaptacijom *Pure ALOHA* i *Slotted ALOHA* metoda primenjenih na električnom provodniku (inicijalno na koaksijalnom kablu). Pojam „*Multiple Access”* odnosi se na pristup većeg broja učesnika uz mogućnost nastanka kolizije, što smo imali i kod ALOHAnet mreže, ali se uvode i dva nova koncepta: „*Carrier Sense”* i „*Collision Detection”*. Prenosni medijum („*Carrier”*) se najpre osluškuje da li ga neko trenutno koristi („*Sense”*), tako da se paketi mogu poslati jedino ako je medijum slobodan. Drugi koncept se odnosi na eksplicitni način detekcije kolizije (eng. *Collision Detection*), koji se umesto obavezne potvrde prijema za svaki uspešno preneti paket sada bazira na aktivnom praćenju saobraćaja na medijumu (Slika 2.5).



Slika 2.5. CSMA/CD kontrola pristupa medijumu

Slično principu *Pure ALOHA*, i kod CSMA/CD metode paketi se mogu poslati u bilo kom trenutku, bez vremenske sinhronizacije, ali samo ako je medijum slobodan. Ako je medijum zauzet od strane drugog učesnika koji šalje podatke, posmatrani uređaji će najpre da sačeka da se medijum oslobodi pre nego što pošalje svoj paket. Ali čak i kada je medijum slobodan u nekom trenutku, zbog međusobne udaljenosti računara u mreži i ograničene brzine propagacije signala, dva ili više uređaja mogu gotovo istovremeno da pošalju svoje pakete, što će dovesti do kolizije. Ovaj slučaj ilustruje Slika 2.6, gde su računari prostorno raspoređeni po X osi, dok je vreme prikazano po Y osi. U trenutku t_1 računar B konstatuje da je medijum slobodan i šalje svoj paket, koji nastavlja da propagira po mreži. Neposredno nakon toga, u trenutku t_2, ovaj paket još nije stigao do računara D, koji konstatuje da je medijum i dalje slobodan, pa stoga i on šalje svoj paket. Čak iako računari B i D završe slanje paketa, oni će po koaksijalnom kablu i dalje da propagiraju i u nekom delu mreže će ubrzo da se sretnu i izazovu koliziju (trenutak t_3). Neki uređaji će da prime ispravne pakete, poput računara A koji prima ispravan paket od računara B, i računar E koji prima ispravan paket od računara D. Ipak, nakon nastanka kolizije, oba paketa su uništena, a rezultujući signal će da nastavi da propagira kroz mrežu. Primetimo da i ovaj interferirani signal ima određeni oblik i intenzitet, koji se takođe može interpretirati kao niz nula i jedinica, ali su ovi podaci ipak nevalidni.



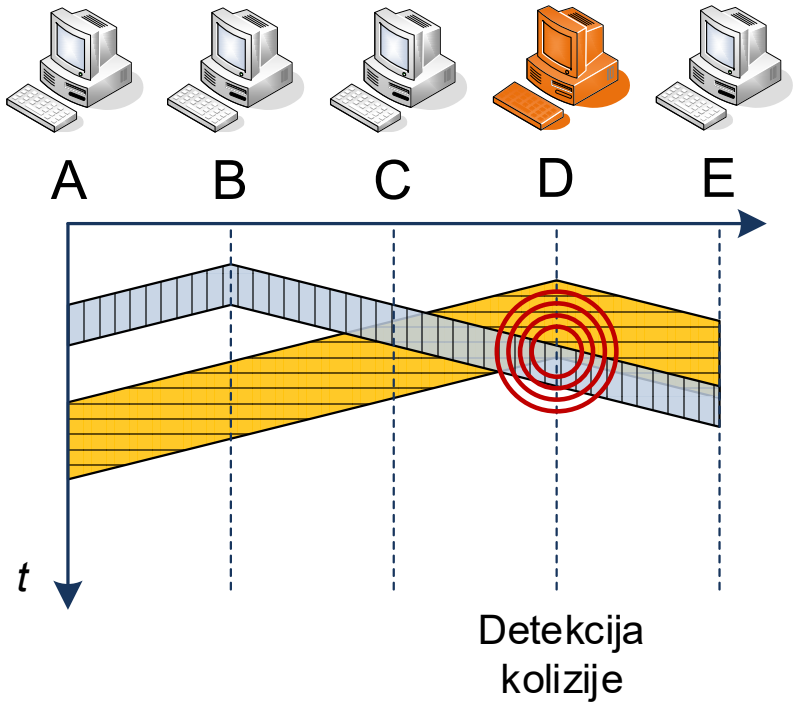
Slika 2.6. Ilustracija nastanka kolizije u mreži

CSMA/CD metod rešava problem kolizije tako što svaki učesniku u mreži detektuje koliziju, a uništeni paketi su odbacuju i ponovo šalju. Tom prilikom, javljaju se dva osnovna problema:

- Kako uređaj koji šalje paket prepoznaje da je njegovi paket uništen u koliziji, a da bi mogao da ga ponovo pošalje?
- Kako ostali uređaji koji primaju pakete prepoznaju da je paket nevalidan, jer je uništen u koliziji?

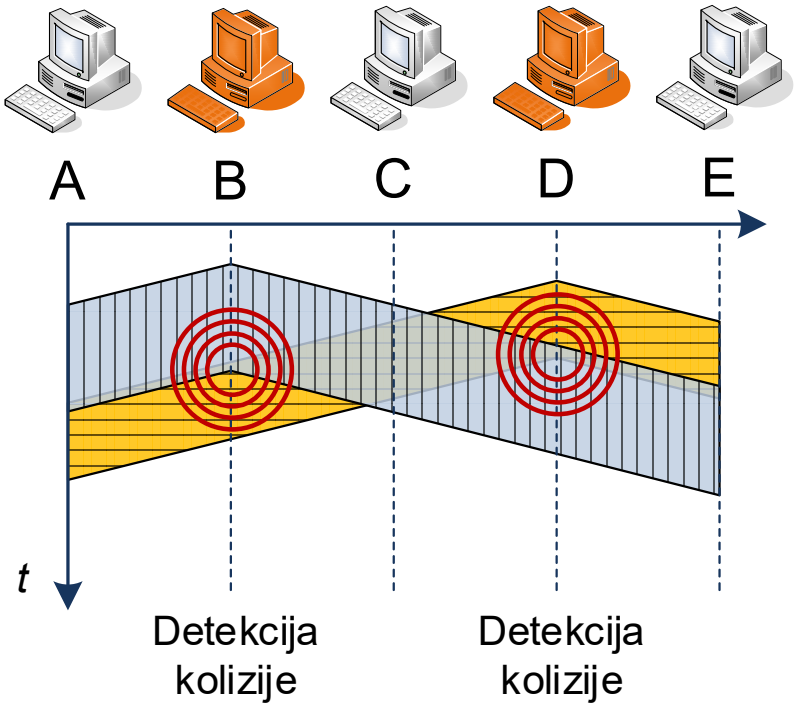
Rešenje prvog problema omogućavaju električne karakteristike prenosnog provodnika (kabla) kao i mrežnih interfejsa koji uređaje povezuje na kabl. Uređaj dok šalje paket na mrežu, istovremeno osluškuje i prima sa mreže trenutno prisutne signale. Ako nema kolizije, binarni podaci koji se šalju na mrežu u potpunosti će odgovarati podacima koji se istovremeno i primaju sa mreže. U slučaju da se ovi podaci ipak razlikuju, čak i za jedan bit, smatra se da je došlo do kolizije, kao što Slika 2.7 ilustruje za računar D.

Potrebno je posebno istaći da uređaj na ovaj način može da detektuje koliziju jedino tokom slanja svog paketa i to na mestu u mreži gde je on povezan. U navedenom primeru, računar B na ovaj način ipak neće detektovati koliziju svog paketa, budući da je slanje završio pre nego što je do njega stigao paket od računara D, koji takođe učestvuje u koliziji.



Slika 2.7. Detekcija kolizije od strane računara D

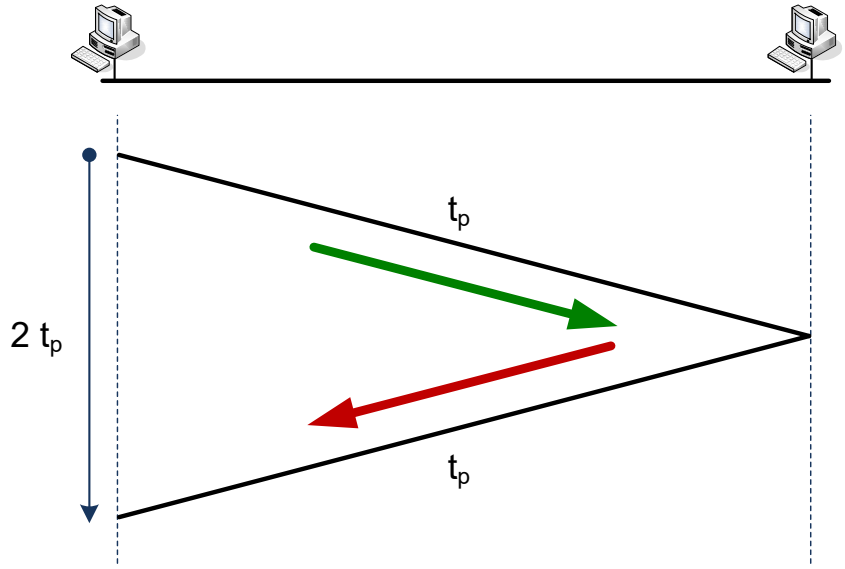
Stoga i računar B mora da paket šalje dovoljno dugo da omogući drugom paketu da dođe do njega i time detektuje da paket koji on šalje izaziva koliziju (Slika 2.8).



Slika 2.8. Detekcija kolizije od strane oba računara čiji paketi učestvuju u koliziji (B i D)

Da bi se omogućila detekcija kolizije u svim slučajevima, potrebno je obezbediti da vreme slanja jednog paketa bude dovoljno veliko da se omogući da drugi paket sa kojim nastaje kolizija u udaljenom delu mreže dođe do posmatranog uređaja, dok se prvi paket i dalje šalje.

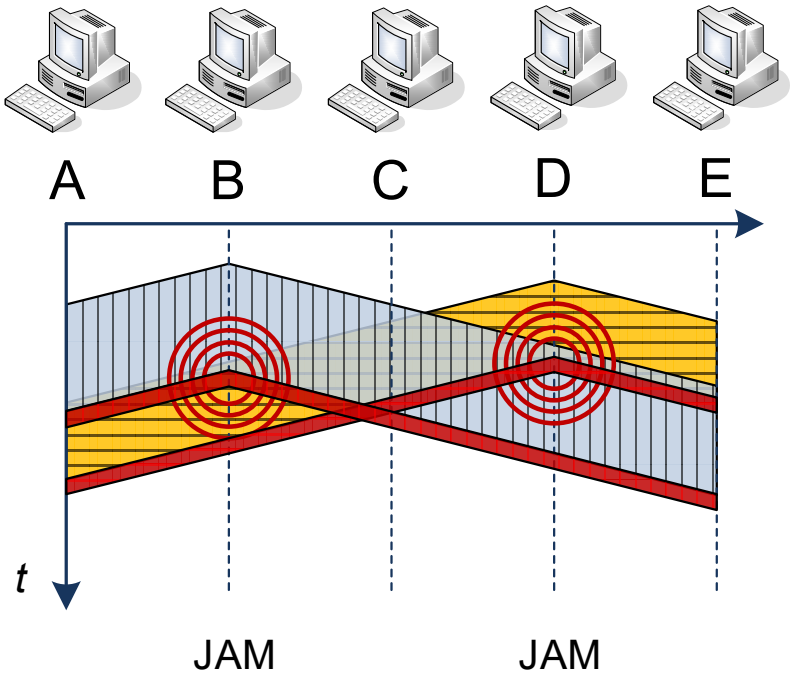
Ovo dalje zavisi do dva parametra: vremena izlaska celog paketa na mrežu i vremena propagacija paketa u mreži. Dalje, vreme za koje će paket da izađe na mrežu zavisi od veličine paketa i brzine kojom se paket šalje. Brzina slanja paketa je fiksna za određenu tehnologiju i odgovara propusnom opsegu mreže, izraženu u bitima u sekundi – što je veća brzina slanja, ceo paket će brže da izađe na mrežu. Vreme propagacija paketa zavisi od brzine prenosa signala, koja je takođe fiksna i iznosi oko 2/3 brzine svetlosti, i rastojanja u mreži. Veličina paketa i rastojanje u mreži su u opštem slučaju varijabilni parametri, pa moraju da se usaglase i ograniče na vrednosti koje će garantovati detekciju kolizije i u najnepovoljnijem slučaju.



Slika 2.9. Detekcija kolizije od strane oba računara čiji paketi učestvuju u koliziji (B i D)

Najnepovoljniji slučaj nastaje kada se posmatraju dva najudaljenija uređaja u mreži (računari A i B, Slika 2.9), između kojih je vreme propagacije paketa i najveće (t_p). Ako je računar A poslao svoj paket, najkasniji trenutak nastanka kolizije u delu mreže kod računara B je kada on pokuša slanje paketa neposredno pred dolazak paketa od računara A, dakle posle vremena t_p. Da bi informacija o koliziji stigla i do računara A, potrebno je da paket od uređaja B pređe isti put, ali u suprotnom smeru, za šta je potrebno dodatno vreme propagacije t_p. Da bi računar A mogao da detektuje da je njegov paket u koliziji, on mora da i dalje šalje paket na mrežu. Imajući u vidu fiksnu brzinu izlaska paketa na mrežu, kao i fiksnu brzinu propagacije elektromagnetnih signala po bakarnim provodnicima, uvodi se minimalna veličina paketa i maksimalno dozvoljeno rastojanje u mreži, kako bi slanje paketa na mrežu trajalo duže od dvostrukog najvećeg vremena propagacije paketa ($2t_p$).

Drugi problem se odnosi na to kako ostali uređaji koji ne šalju pakete detektuju koliziju, odnosno kako prepoznaju da je paket koji primaju zapravo uništen u koliziji. Da bi se to obezbedilo, uređaji koji neposredno detektuju koliziju tokom slanja svojih paketa, prekidaju slanje i šalju poseban signal na mrežu koji služi kao informacija za ostale uređaje da je nastupila kolizija i da je prethodno primljene podatke potrebno odbaciti. Ovaj signal se naziva JAM (izgovara se „*džem*“), i predstavlja niz od 32 bita naizmeničnih nula i jedinica. Iako će u početku i JAM signal da bude u koliziji sa zaostalim delovima paketa koji su izazvali koliziju, ubrzo će ovi signali samostalno da propagiraju u svim delovima mreže, kao što ilustruje Slika 2.10.



Slika 2.10. Obaveštavanje ostalih uređaja da je došlo do kolizije – JAM signal

I konačno, kada uređaj detektuje da je njegov paket izazvao koliziju, zaključuje se da slanje nije uspelo i da se paket mora ponovo poslati. Budući da je moguće da i drugi učesnici čekaju na oslobađanje medijuma za slanje svojih paketa, da bi se izbegla kolizija neposredno nakon oslobađanja medijuma, svaki uređaj će da sačeka i dodatni vremenski interval promenljivog trajanja. Ovaj vremenski interval se određuje na slučajan način, tako što se generiše pseudo-slučajan broj koji se množi sa fiksnim jediničnim intervalom. Ovaj proces zadržavanja paketa nakon detekcije kolizije se naziva *back-off* algoritam. Slično kao kod *Slotted ALHOA* principa, ovim se ne garantuje potpuno izbegavanje kolizije, ali je verovatnoća njenog nastanka znatno manja.

2.5 Eternet – realizacija CSMA/CD

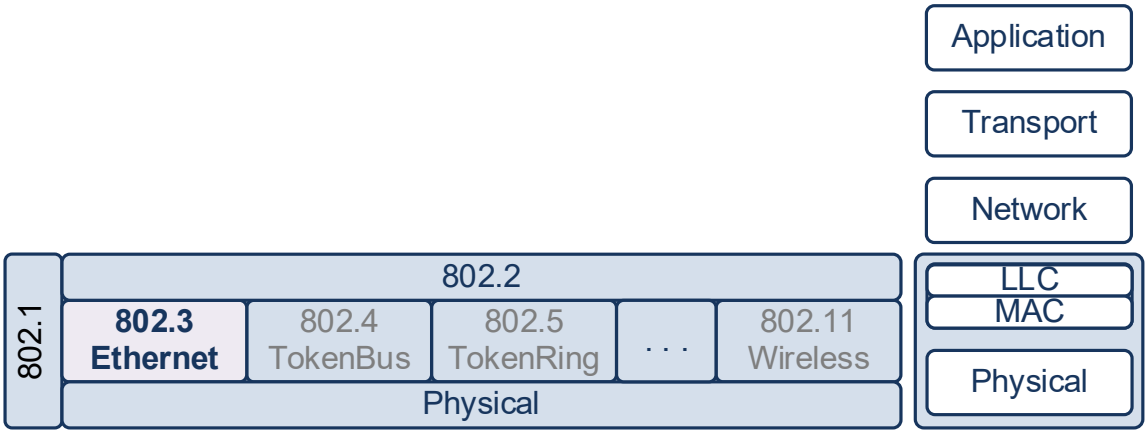
Inspirisano ALOHAnet mrežom, realizacija CSMA/CD principa korišćenjem koaksijalnog kabla kao prenosnog medijuma je omogućila daleko veće brzine prenosa podataka. Prva mreža ove vrste je omogućavala povezivanje do 256 računara brzinom od čak 2.94 Mbps, korišćenjem koaksijalnog kabla maksimalnog rastojanje od jedne milje (1.6 km). Tačnije, CSMA/CD princip je nastao pri razvoju i praktičnoj realizaciji ove mrežne tehnologije, koja je po analogiji sa etrom (eng. *ether*), hipotetičkim medijumom za prenos elektromagnetnih talasa u prostoru, nazvana Eternet (eng. *Ethernet*).

Eternet mreža je nastala u istraživačkom centru kompanije Xerox u Palo Altu, što je brzo privuklo pažnju ostalih tehnoloških kompanija. U konzorcijumu sa kompanijama *Digital Equipment* i *Intel*, Eternet se unapređuje do brzine prenosa od 10 Mpbs, što se 1980. godine objavljuje kao otvoreni standard koji su počeli da primenjuju i mnogi drugi proizvođači komunikacione opreme. Ova varijanta je nazvana *Ethernet II*, takođe poznato i kao *DIX Ethernet*, što je izvedeno kao akronim iz naziva prethodno navedenih kompanija (*Digital Equipment*, *Intel* i *Xerox*).

Dalja standardizacija Eterneta se sprovodi od strane komiteta za standarde pri organizaciji IEEE. Inače, serija IEEE sandarda pod brojem 802 se odnosi sloj veze podataka (*Data-Link Layer*) u okviru koga se definišu LAN tehnologije. IEEE 802.2 standard definiše viši podsloj sloja veze, tzv. *Logical Link Control*, zajednički za sve izvedene mrežne tehnologije MAC podsloja. Primera radi, *Token Ring* tehnologija, razvijena od strane kompanije IBM, definisana je u oznaci 802.5, dok su kasnije razvijene bežične mreže (eng. *wireless*) definisane u seriji standarda u oznaci 802.11. Tako je i Eternet tehnologija 1985. godine zauzela svoje mesto u 802 seriji standarda i to u oznaci IEEE 802.3 (Slika 2.11). Nažalost, ova varijanta je bila delimično izmenjena i nekompatibilna u odnosu na DIX varijantu, pa su proizvođači opreme morali da implementiraju oba standarda.

Eternet je dalje nastavio da se razvija u pogledu brzina prenosa. Godine 1995. standardizovan je Eternet na brzinama od 100 Mbps, a već 1998. godine i na brzinama od 1 Gbps.

Navedeni standardi u najvećoj meri definišu fizičke, elektromagnetne i optičke karakteristike prenosnih medijuma (bakarnih i optičkih kablova) i prateće opreme koja obezbeđuje ovako velike brzine prenosa, dok se logički koncept Eterneta malo menjao. To je za posledicu imalo kompatibilnost svih varijanti Eterneta i interoperabilnost uređaja koji su radili na različitim brzinama. Uz jednostavan dizajn, koji nije limitirao brzinu prenosa podataka, sve ovo je omogućilo ne samo da Eternet tehnološki evoluira i opstane do današnjih dana, već je vrlo brzo postala dominantna LAN tehnologija, dok su mnoge druge tehnologije praktično izumrle.



Slika 2.11. IEEE 802 serija standarda sloja veze podataka

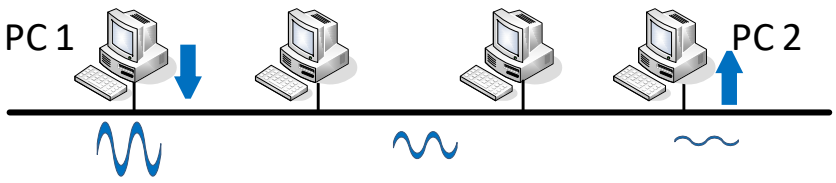
3. Eternet

U prethodnom poglavlju je opisan konceptualni princip rada CSMA/CD metode pristupa medijumu, na kojoj se bazira Eternet tehnologija. Tehnička realizacija Eternet tehnologije obuhvata i mnoge druge elemente i implementacione detalje, koji ćemo u ovom poglavlju ispratiti takođe kroz razvoj i unapređenje tehnologije tokom vremena.

3.1 Kolizija i njeni hirovi

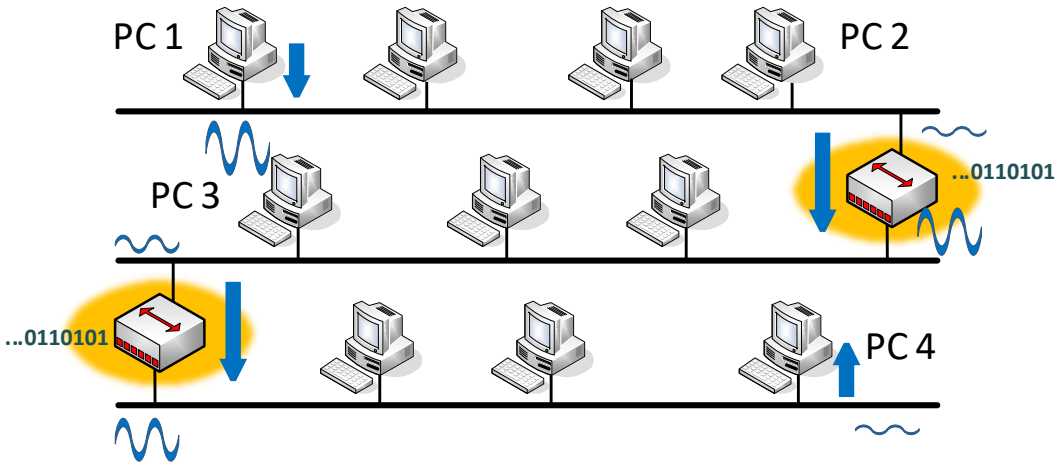
Eternet je nastao iz potrebe povezivanja uređaja u poslovnom okruženju, što tipično čine prostorije (kancelarije) u poslovnoj zgradi ili čak više susednih zgrada u kampusu. Stoga Eternet spada u kategoriju lokalnih računarskih mreža (eng. *Local Area Network*, skr. LAN).

Neminovno prisutna kolizija i uslovi za njenu detekciju nameću ograničenje u pogledu veličine mreže. Drugo ograničenje je uzrokovano fizičkim karakteristikama prenosnog medijuma, u ovom slučaju koaksijalnih kablova, budući da elektromagnetni signali slabi i deformiše se tokom prostiranja kroz kabl. Uslov za ispravan rad je da i najudaljeni uređaj može da rekonstruiše primljeni signal u ispravnu sekvencu nula i jedinca (Slika 3.1).



Slika 3.1. Slabljenje signala na segmentu koaksijalnog kabla

Prvobitni koaksijalni kablovi koji su se koristili u Eternet mrežama su bili dosta robusni i omogućavali su prenos signala do 500 metara. Iako se to, na prvi pogled, može učiniti značajnim rastojanjem, to ipak nije dovoljno da se u iole većim u poslovnim zgradama povežu sve kancelarije na više spratova. Da bi se prevazišlo ovo ograničenje, potrebno je povezati više ovakvih segmenata koaksijalnog kabla u niz i tom prilikom obezbediti prijem signala na kraju jednog segmenta, njihovo „osvežavanje“ i prosleđivanje na drugi segment. Za ovo su zaduženi posebni uređaji koji se nazivaju ripiteri (eng. *repeater*). Drugim rečima, ripiteri spajaju dva koaksijalna segmenta, primaju signale sa njih, koje rekonstruišu u originalni niz nula i jedinica (kao i svaki drugi uređaji), a zatim generišu nove fizičke signale koje prosleđuju na drugi segment. Na ovaj način ripiteri imaju ulogu pojačavača signala, čime se omogućava veće fizičko rastojanje u Eternet mreži (Slika 3.2)



Slika 3.2. Povezivanje više segmenta koaksijalnog kabla preko ripitera

Za razliku od drugih uređaja, ripiteri ne učestvuju u detekciji kolizije, već samo prenose sve signale, uključujući i JAM signale. Efektivnim povećavanjem ukupnog prenosnog medijuma, odnosno rastojanja u mreži, povećava se i sam kolizioni domen, što utiče na uslove koji se moraju zadovoljiti radi detekcije kolizije.

Uslov za detekciju kolizije je da vreme izlaska paketa minimalne veličine na mrežu bude veće od dvostrukog najvećeg vremena propagacije signala u mreži. Brzina izlaska paketa na mrežu zavisi od brzine protoka podataka (eng. *bandwidth*), označeno sa B_w, što je za prvoditni Eternet po koaksijalnim kablovima iznosilo 10 Mbps. Sa ovim brzinama prenosa 10 miliona bita će da izađe na mrežu za 1 sekundu, što znači da će 1 bit da izađe na mrežu za desetomilioniti deo sekunde, odnosno za 0.1μs (mikrosekunda). Ovo vreme izlaska jednog bita na mrežu naziva se ***bit-time*** (T_b) i u opštem slučaju iznosi:

- $T_b = 1 / B_w$

Budući da se ograničava minimalna veličina paketa, koja izraženo u bitima iznosi L_{min}, uvodi se najmanje vreme izlaska paketa, tzv. ***slot-time*** (T_s), koje će da iznosi:

- $T_s = L_{min}\ T_b = L_{min} / B_w$

Brzinu protoka podataka ne treba mešati sa brzinom prenosa signala (koji prenosi podatke), a koja po bakarnom medijumi iznosi oko 200.000 km/s. Vreme propagacije signala po jednom segmentu (T_p) maksimalne veličine od 500m će stoga iznositi 2.5μs. Za više spojenih segmenata ovo vreme se množi sa brojem segmenata, ali se mora dodati i vreme osvežavanja signala u samim ripiterima. U vreme kada se standardizovao Eternet na brzinama od 10 Mbps (početak 80-ih godina prošlog veka) za maksimalno vreme osvežavanja, odnosno propagacije signala kroz ripiter (T_r) ustanovljeno je 3μs, dok je u praksi, u zavisnosti od elektronike samog ripitera, ovo vreme često bilo i dosta manje.

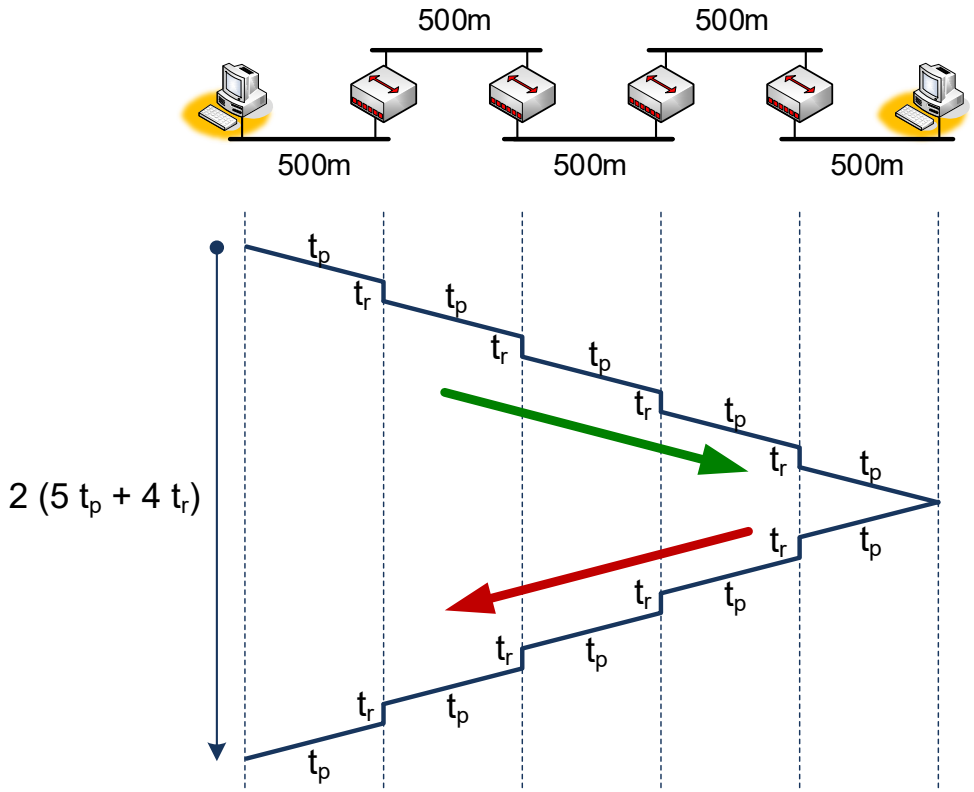
I konačno, sa navedenim konkretnim vremenskim ograničenjima, standardom je utvrđeno ograničenje od maksimalno 5 segmenata koaksijalnih kablova dužine do najviše 500m, a koji su međusobno spojeni sa 4 ripitera. Maksimalno dvostruko vreme propagacije signala u ovom najekstremnijem slučaju iznosi:

- $T_{d\ (max)} = 10\ T_p + 8\ T_r = 49\mu s$

Iz uslova za detekciju kolizije, ovo ujedno predstavlja i minimalno vreme *slot-time*, iz čega se dobija da paket ne sme biti manji od 490 bita, odnosno:

- $L'_{min} = T'_s \; B_w = 49ms \; 10Mbps = 490 \; b$

Budući da se veličina paketa izražava u bajtovima, vrednost koja se lepo uklapa u gore navedeno ograničenja iznosi 512 bita, odnosno 64 bajta, što je standardnom postavljeno za minimalnu veličinu Eternet paketa. Ovo ograničenje se implementira u elektronici mrežnih kartica, odnosno interfejsa uređaja, pa važi i u mrežama manjih dijametara, npr. mreža od samo jednog segmenta.



Slika 3.3. Maksimalno vreme propagacije paketa u najnepovoljnijem slučaju

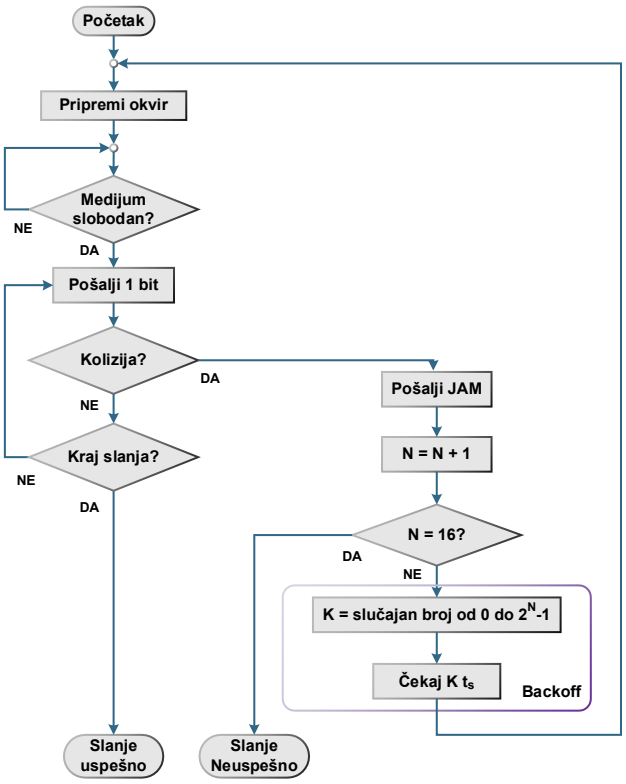
Nakon uspešnog slanja jednog paketa, a pre slanja drugog (od strane istog ili različitog uređaja), mora da se obezbedi da medijum bude slobodan određeni kratak vremenski interval. Ovo je bitno iz dva razloga. Prvo, paketi moraju da budu razdvojeni, kako bi mogli uspešno da se razlikuju i uspešno prime. Drugi razlog je što nakon slanja svog paketa elektronika mrežne kartice treba da pređe u režim prijema, što zahteva određeno vreme. Ovaj vremenski interval se naziva *Inte Frame Gap* i postavljen je na period od 96 *bit-time*, što za Eternet na 10 Mbps iznosi 9.6 μs.

U slučaju da je slanje paketa neuspešno usled nastanka kolizije, potrebno je sačekati slučajan vremenski interval koji se određuje tzv. eksponencijalnim *back-off* algoritmom. Ovaj algoritam uzima u obzir i trenutni broj neuspelih pokušaja slanja paketa (*N*), tako što se bira pseudo-slučajan ceo broj u intervalu od 0 do 2^N-1, a vreme čekanja se dobija kada se ovaj broj pomnoži sa fiksnim *slot-time* intervalom. Na taj način se eksponencijalno povećava opseg vremena koji se čeka, a time i eksponencijalno smanjuje verovatnoća nastanka naredne kolizije. Primetimo

da nakon nastanka prve kolizije (N=1) najmanje dva uređaja sprovode ovaj algoritam biranjem vrednosti 0 ili 1, pa je verovatnoća nastanka nove kolizije čak 50%, ali se svakim narednim pokušajem verovatnoća nastanka kolizije se smanjuje za polovinu.

Uzimajući gore navedeno u obzir, proces slanja Eternet paketa na mrežnu može se predstaviti algoritmom koji prikazuje Slika 3.4, gde je bitno istaći sledeće:

- Pre slanja paketa prenosni medijum se osluškuje i u slučaju zauzeća, čeka se da se oslobodi.
- Kada medijum postane slobodan, sačeka će se još i dodatni kratak vremenski period (*Inter Frame Gap)* da bi se počelo sa slanjem paketa.
- Paket se šalje sekvencijalno, bit-po-bit, uz stalnu proveru da li je došlo do kolizije.
- Ako se detektuje kolizija, prekida se dalje slanje paketa i šalje se JAM signal.
- Inkrementira se brojač pokušaja slanja paketa, koji je ograničen na 16.
- Čeka se dodatno vreme prema *back-off* algoritmu, nakon čega počinje novi pokušaj slanja paketa.



Slika 3.4. Proces slanja Eternet paket na mrežu

3.2 Do poslednjeg bajta

Eternet funkcioniše na drugom nivou i osnovni zadatak mu je da podatke trećeg nivoa prenese učesnicima u lokalnoj računarskoj mreži. Tom prilikom učesnici u komunikaciju moraju da se prepoznaju, odnosno da budu jedinstveno identifikovani korišćenjem tzv. adresa. Dalje, paketi se prenose nezavisno jedni od drugih, tako da oni osim podataka koje prenose, moraju da sadrže i dodatne kontrolne informacije u zaglavlja paketa. Sve ove informacije se prenose kao binarne nule i jedinice, koje su struktuirane u striktan poredak bajtova, tzv. format paketa.

Slika 3.5. Format Eternet paketa

3.3 *MAC* adrese

Učesnici u komunikaciji se identifikuju preko adresa, koje se u ovom slučaju nazivaju MAC adrese, budući da Eternet radi na sloju pristupa medijumu - *MAC* nivo (eng. *Media Access Control*).

MAC adrese se sastoje od 6 bajtova, koji zapravo identifikuju mrežne priključke i to tako što su fabrički upisane u njihovu elektroniku, tzv. *Burned-in Address*. Ove adrese moraju da budu jedinstvene i nazivaju se unikast (eng. *Unicast*). Jedinstvenost adresa se fabrički garantuje, tako što proizvođač opreme zakupi blok adrese, koje imaju prva tri bajta fiksna koja identifikuju proizvođača (eng. *Organizational Unique Identifier*, skr. *OUI*), dok u okviru tog bloka poslednja tri bajta (24 bita) mogu proizvoljno da variraju, dajući ukupno 2^{24} mogućih adresa. Kada se jedan blok istroši, zakupi se novi, a ima ih sasvim dovoljno da se praktično garantuje da će svi mrežni adapteri ikada proizvedeni biti jedinstveni.

MAC adrese se korisnicima prikazuju i interpretiraju u heksa-dekadnom obliku. Tom prilikom ne postoji jedinstvena konvencija kako se cifre grupišu i razdvajaju, kao ni to da li se koriste velika ili mala slova abecede. Primeri zapisa iste *MAC* adrese su sledeći:

- 82-A3-E1-CA-38-1B
- 82.A3.E1.CA.38.1B
- 82a3.e1cA.381b
- 82A3.E1CA.381B.

3.4 Format Eternet paketa

Najava paketa

Budući da se paketi prenose asinhrono, odnosno mogu da se pojave u bilo kom trenutku, elektronika mrežnih kartica na prijemu mora najpre da ih prepozna. Zato Eternet paketima najpre prethodi 8 bajtova fiksne strukture, naizmeničkog niza jedinica i nula, koje služe za sinhronizaciju bitskog poretka u vremenu. Prvih 7 bajtova se naziva preambula (eng. *preamble*), dok se osmi bajt završava sa dve jedinice („10101011“), a naziva se *Start Frame Delimiter*, budući da se njime najavljuje početak pravog Eternet okvira. Stoga se ovi podaci ne smatraju sastavnim delom Eternet okvira.

Zaglavlje Eternet paketa

Eternet paket počinje zaglavljem koje sadrži informacije „ko“, „kome“ i „šta“ prenosi, podeljeno u sledeća polja:

- Odredišna MAC adresa
- Izvorišna MAC adresa
- Tip/Dužina

Tipična razmena paketa u mreži se sprovodi između dva učesnika, tzv. izvorišta i odredišta, a koji su identifikovani izvorišnom i odredišnom MAC adresom. Ovaj vid komunikacije se naziva unikast.

Izvorišni uređaj šalje paket na mrežu, a ako nema kolizije paket, će po CSMA/CD principu da pristigne do svakog učesnika u mreži. Svaki uređaj će da primi paket, izdvoji iz zaglavlja prvih 6 bajta koji čine odredišnu MAC adresu i tu vrednost da uporedi sa svojom fabrički upisanom MAC adresom. Budući da je MAC adresa jedinstvena, samo jedan uređaj može da prepozna svoju MAC adresu kao odredišnu, u kom slučaju se paket prihvata, dok svi ostali uređaju odbacuju paket, jer se ove vrednosti razlikuju.

U pojedinim slučajevima je korisno da se paket isporuči svim učesnicima i taj vid komunikacije se naziva brodkast (eng. *broadcast*). Kasnije ćemo imati primere sistemskih protokola koji na ovaj način oglašavaju pojedine informacije koje su korisne svim učesnicima u mreži. Za ovu namenu se koristi posebna tzv. brodkast MAC adresa, koja se sastoji od svih jedinica, što u heksadekadnom zapisu iznosi FFFF.FFFF.FFFF. Implicitno se smatra da brodkast MAC adresa identifikuje sve uređaje, pa će svi mrežni adapteri da prihvate paket sa ovom adresom u polju odredišne MAC adrese.

Izvorišna MAC adresa naravno identifikuje pošiljaoca paketa. Iako se čini da je ova informacija neophodna kako bi se poslao odgovor, budući da su unikast komunikacije obično dvosmerna, to ipak nije slučaj. Protokoli viših nivoa zapravo održavaju komunikaciju u oba smera, dok Eternet služi da prenese podatke viših nivoa (zaključno sa aplikativnim nivoom), ali nezavisno u jednom i drugom smeru. Drugim rečima, Eternet ne pamti izvorišnu MAC adresu iz zaglavlja inicijalnog paketa, već pri vraćanju odgovora u suprotnom smeru, kao uostalom i za svaku komunikaciju, koristi poseban mehanizam za otkrivanje odgovarajuće MAC adrese. Na ovom mestu napomenimo da se tom prilikom koristi tzv. ARP protokol, koji je kasnije detaljno opisan u posebnom poglavlju. Takođe ćemo u nastavku videti i neke od primena izvorišne MAC adrese iz zaglavlja paketa.

Osnovna uloga Eterneta je da prenese podatke višeg nivoa, na kome mogu istovremeno da funkcionišu različiti protokoli. Svaki protokol ima standardizovanu brojčanu identifikaciju, a prvobitna varijanta Eterneta (*Ethernet DIX*) je za označavanje protokola višeg nivoa kome treba predati podatke uvela polja dužine dva bajta, pod nazivom „Tip“ (eng. *Type*). Naredna varijanta Eterneta standardizovana od strane IEEE organizacije je koristila dodatna polja, koja su zapravo bila enkapsulirana kao zaglavlje međusloja do je navedeno polje od dva bajta koristila za označavanje dužine podataka koji se prenose u paketu (eng. *Length*). Ove dve varijante nisu bile kompatibilne, pa su proizvođači mrežnih uređaja bili prinuđeni da implementiraju oba standarda. To je ipak prevaziđeno u narednoj reviziji Eternet standarda, od kada se ovo polje od dva bajta koristi dvojako: za vrednosti do 1500 se tretira kao dužina, budući da je to i najveća moguća dužina Eternet paketa, dok se veće vrednosti tretiraju kao identifikacije protokola.

Potpis

Osim podataka u zaglavlju paketa, Eternet koristi i posebno kontrolno polje dužine 4 bajta koje se nalazi na kraju paketa. Ovo polje se naziva FCS (eng. *Frame Check Sequence*) i služi za kontrolu eventualno nastale greške. Prilikom formiranja paketa na izvorištu, svi bajtovi zaglavlja i podataka sekvencijalno učestvuju u računanju funkcije CRC (eng. *Cyclic Redundancy Check*), koja se sastoji od bajtovskih operacija koje se jednostavno i brzo

hardverski sprovode (sabiranje, računanje ostatka pri binarnom deljenju, pomeraj itd). Dobijena vrednost od 4 bajta se postavlja na kraj poruke u polje FCS i tretira kao „potpis“ čija vrednost zavisi od svakog bita prethodnih polja u poruci. Prilikom prijema paketa sprovodi se isti postupak računanja CRC funkcije, a ako izračunata vrednost odgovara sadržaju ovog polja, smatra se da je poruka preneta bez grešaka. U slučaju greške samo na jednom bitu ove vrednosti će da se razlikuju, u kom slučaju se smatra da je paket oštećen i on se odbacuje. Štaviše, CRC funkcija je takva da promena jednog ili više bita daje potpuno različite vrednosti, tako da je praktično nemoguće da se dve greške međusobno anuliraju, što je slučaj kod provere bitske parnosti.

Napomenimo i to da za odbačene pakete Eternet nema nikakav mehanizam da obavesti pošiljaoca o tome i zahteva ponovno slanje paketa, kao što imaju neki drugi protokoli ove vrste. Na prvi pogled ove se može učiniti kao nedostatak, ali je krajnji efekat sasvim suprotan – to ga čini jednostavnim za implementaciju, a time i jeftinijim i široko prihvaćenim na tržištu. Dodatno, jednostavnost je, kao što ćemo ubrzo videti, omogućila fleksibilnost za prilagođavanja i migraciju na veće brzine i druge principe prenosa.

Podaci

I zaglavlje i prateće FCS polje služe da bi se preneli stvarni podaci, koji su sadržani u središnjem polju podataka (eng. *Data*) koje je po svojoj prirodi varijabilne dužine. Maksimalna dužina ovog polja je ograničena na 1500 bajtova, ali zbog uvedene minimalne veličine celog Eternet paketa radi detekcije kolizije, ovo polje ima i minimalnu veličinu. Kada se od minimalne veličine paketa od 64 bajta (ne računajući preambulu i SFD polje od ukupno 8 bajta) oduzme veličina zaglavlja od ukupno 14 bajta, kao 4 bajta pratećeg FCS polja, dobija se minimalna veličina polja podataka od 48 bajtova. Primetimo i to da je maksimalna veličina Eternet paketa 1518 bajtova.

Postavlja se pitanje kako ispoštovati minimalnu veličinu ovog polja, ako nemamo dovoljno podataka za prenos? U tom slučaju se na kraj stvarnih podataka „veštački“ umeću dodatni bajtovi do minimalne veličine, a koji se zapravo neće ni koristiti (eng. *padding*). Istina, i podaci viših nivoa sadrže kontrolne podatke u svojim zaglavljima, ali je njihova ukupna dužina ipak manja od 48 bajtova, pa je potrebno voditi računa o ovom ograničenju.

Pažljivi čitalac može da postavi i dodatno pitanje: kako se zna gde je granica između stvarnih podataka i ovih dodatno umetnutih bajtova koje na prijemu treba ignorisati? Ovi dodatni podaci u umetnuti na Eternet nivou, dok se stvarna dužina „korisnih“ podataka prenosi u posebnom polju zaglavlja višeg nivoa. Na taj način će viši nivo na prijemu znati koliko tačno podataka treba da uzme, dok će ostale bajtove da ignoriše.

3.5 Zbogom ripiteri

Za korisnike je nastanak kolizije i ponovno slanje paketa gotovo neprimetna i prihvatljiva pojava, ali samo do određenog nivoa intenziteta saobraćaja. Sa druge strane intenzitet saobraćaja u mreži zavisi od broja učesnika, količine podataka i učestanosti kojom se oni šalju, što zavisi od načina korišćenja od strane korisnika. Takođe, verovatnoća nastanka kolizije zavisi i od same tehnologije, odnosno propusnog kapaciteta mreže – pri većim brzinama prenosa deljeni medijum će kraće vreme da bude zauzet, što smanjuje verovatnoću nastanka kolizije.

Povezivanjem segmenata koaksijalnih kablova uz pomoć ripitera omogućeno je da se dosegnu uređaji na većim rastojanjima, ali i da se pokrije više prostorija, što potencijalno donosi veći broj uređaja. Veći broj učesnika u mreži doprinosi intenzivnijem saobraćaju i učestalijoj koliziji, koja propagira kroz ripitere do svih uređaja. Takođe, i komunikacione potrebe korisnika su vremenom porasle, što je dodatno opteretilo mrežu koja ima svoj ograničen limit. Iako je Eternet nakon svoje pojave odlično prihvaćen na tržištu, kolizija je počela da predstavlja sve veći problem i dalja sudbina ove tehnologije ozbiljno je dovedena u pitanje.

Da bi se smanjila kolizija, bilo je potrebno rešiti dva suprotstavljena zahteva: zadržati veliku mrežu sa dosta učesnika, a ipak lokalizovati koliziju. Rešenje je bilo da se deljeni medijum lokalizuje na nivo pojedinačnih segmenta, tako što se paketi u komunikaciji između uređaja na jednom segmentu ne propuštaju na drugi segment. Ipak, da bi se zadržala mogućnost komunikacije na nivou cele mreže, pri komunikaciji uređaja sa različitih segmenata neophodno je propustiti pakete. Da bi se to sprovelo, ripiteri moraju da rade dodatni posao - da odluče da li da propuste ili blokiraju pakete. Ali oni tada više nisu ripiteri, već nova vrsta uređaja, koja je nazvana bridž (eng. *bridge*, u pojedinoj literaturi na srpskom jeziku i pod terminom: mrežni most).

Da bi bridž mogao da odluči da li da propusti ili blokira paket, neophodno je da utvrdi sledeće:

- Kome je paket namenjen, odnosno ko je odredišni uređaj?
- Gde se nalazi odredišni uređaj u mreži?

Prvi problem se jednostavno rešava, budući da svaki paket u svom zaglavlju nosi informaciju o odredišnoj MAC adresi, koje identifikuje odredišni uređaj. Za razliku od ripitera, koji prepoznaje paket samo na nivou bita, bridž mora da prepozna i strukturu Eternet paketa i iz njegovog zaglavlja da izdvoji odredišnu MAC adresu. Upravo ova osobina je razlog da se smatra da ripiteri rade na prvom (fizičkom) nivou, dok bridževi rade na drugom (*data-link*) nivou.

Drugi problem je daleko izazovniji. Ipak, nije potrebno da se zna tačna pozicija uređaja u mreži, već je dovoljno da se zna na kojoj strani bridža se nalazi koji uređaj. „Strane bridža“ u ovom slučaju su samo dve i odnosne se na dva mrežna porta koji omogućavaju da bridž spaja („premošćuje“) dva koaksijalna segmenta. Drugim rečima, za svaki uređaj u mreži bridž treba da zna koji od njegova dva porta vodi do tog uređaja. Uređaji su identifikovani preko svojih MAC adresa, dok portovi bridža imaju svoje interne identifikacije. Bridž stoga ima internu tabelu koja uparuje MAC adrese uređaja sa internim oznakama portova koji vode do tog uređaja, a koja se naziva bridžing tabela.

Postupak rada bridža sada se može predstaviti na sledeći način:

- Za svaki pristigli paket gleda se zaglavlje, iz koga se izdvaja odredišna MAC adresa.
- Odredišna MAC adresa se traži u bridžing tabeli, za koju se uzima njoj upareni port.
- Ako je nađeni port različit od porta na koji je pristigao paket, paket se propušta na taj port (eng. *forwarding*), jer on vodi do odredišnog uređaja.
- U suprotnom slučaju, ako port iz tabele odgovara portu na koji je paket pristigao, paket se ne propušta, odnosno uništava, budući da se odredište nalazi na istoj strani bridža i da je paket svakako došao do odredišta putem deljenog medijuma.

Jasno je da bridž ima dodatni i ne tako mali posao da mora da procesira svaki paket na gore opisani način, ali se otvara novo pitanje – kako se uspostavlja bridžing tabela?

Možda bi bilo i moguće da se kroz posebno podešavanje bridža ručno unosi sadržaj bridžing tabele, ali bi to bilo daleko od praktično izvodljivog i prihvatljivog rešenja. Korisnici ne moraju da poznaju tehnologiju rada bridža, kao i svoje MAC adrese, tako da bridževi moraju svoj rad da sprovode autonomno. To se postiže kroz proces učenja (eng. *learning*), kojim se prepoznaju MAC adrese i njihove pozicije u mreži i na taj način uspostavlja bridžing tabela. Nije teško pogoditi da se ovo „učenje“ sprovodi tako što se za svaki pristigli paket, osim odredišne MAC adrese, prepoznaje i izvorišna MAC adrese, a informacija na koji port je pristigao paket ukazuje na kojoj strani se nalazi ta MAC adresa, što se upisuje u bridžing tabelu. Ova informacija nije potrebna za odluku šta raditi sa trenutnim paketom, ali će se svakako iskoristiti za buduće pakete koji su namenjeni za tu MAC adresu.

Poslednji detalj se odnosi na to šta uraditi sa pristiglim paketom ako njegova odredišna MAC adresa još nije naučena, odnosno ne postoji u bridžing tabeli – da li paket odbaciti ili ga propustiti? Rešenje je da se paket ipak propušta, jer je mnogo manja šteta nego da se on uništi – ako se odredište nalazi na „drugoj strani“ bridža to je neophodno, dok se u suprotnom slučaju ispostavlja kao nepotrebno. Ipak, budući da su uređaji u mreži dosta aktivni, vrlo brzo će bridž da nauči gde se nalazi koja MAC adresa u mreži, pa će već sledeće pakete da se uspešno prosleđuje ili blokira. Budući da dolazi do propagacije paketa čak i kada je ona nepotrebna, ovaj proces se naziva *Flooding* (u prevodu „poplava”, ali izraz „*flading*“ ustalio u domaćoj stručnoj terminologiji).

Bridžing tabela je dinamička struktura u memoriji bridža. Ona je inicijalno prazna po uključivanju bridža i vremenom, kako pristižu paketi, se popunjava kroz *Learning* proces. Takođe, veličina bridžing tabele je ograničena na određenu vrednost, koja je dovoljno velika za normalnu upotrebu, ali ipak fiksna (npr. nekoliko hiljada redova). Bridž stoga mora da vodi računa i o neaktivnim MAC adresama, što je slučaj kada je uređaj ugašen ili uklonjen sa mreže. Stoga se ulazima u bridžing tabeli pridružuju tajmeri koji ukazuju na neaktivnost MAC adresa u mreži. Prilikom *Learning* procesa odgovarajući tajmer se resetuje, a u slučaju da se MAC adrese ne javlja određeni fiksni vremenski interval (eng. *time-out*), smatra se da je taj podatak zastareo i on se briše iz bridžing tabele. Ovaj proces se naziva zastarivanje, odnosno *Ageing*.

Da sumiramo, bridževi u svom osnovnom radu sprovode 5 procesa:

- *Forwarding* – prosleđivanje paketa sa jednog na drugi port bridža prema odredišnom uređaju, a na osnovu odredišne MAC adrese iz pristiglog paketa i sadržaja bridžing tabele.
- *Filtering* (*Blocking*) – nepropuštanje, odnosno blokiranje paketa, uz njegovo uništavanje u bridžu, a na osnovu odredišne MAC adresa pristiglog paketa i sadržaja bridžing tabele.
- *Learning* – popunjavanje bridžing tabele sa infomacijama o MAC adresi uređaja na mreži i portu koji vode do nje, uz resetovanje internog tajmera, a na osnovu izvorišne MAC adresa pristiglog paketa.
- *Flooding* – prosleđivanje paketa sa jednog na drugi port bridža kada MAC adresa odredišta ne postoji u bridžing tabeli.
- *Aging* – Brisanja redova iz bridžing tabele za MAC adrese koje su neaktivne određeni vremenski period.

Navedeni način rada bridževa prouzrokuje sledeće korisne osobine:

- Bridževi za osnovni rad ne zahtevaju nikakvo podešavanje, odnosno konfigurisanje – dovoljno ih je samo pravilno povezati na mrežu i uključiti.
- Po uključenju u mrežu, briževi će svaki paket proslediti do svog odredišta, odnosno neće prouzrokovati gubitak ni jednog paketa.
- Postavka i rad bridževa u mreži ne zahteva nikakvo dodatno podešavanje ostalih uređaja u mreži. Štaviše, za ostale uređaje u mreži bridževi su nevidljivi, pa je princip njihovog rada originalno nazvan *Transparent Bridging* [8].

Ipak, ovakav način rada bridža može i da se zloupotrebi, što je pojava koja konstantno prati razvoj tehnologije. Moguće je softverski generisati Eternet pakete koji imaju lažne, odnosno nepostojeće MAC adrese u polju izvorišta (tzv. *spoofing*). Ovakvi pakete će da dođu do bridža, koje će kroz *Learning* proces da upiše ove nepostojeće MAC adrese u svoju bridžing tabelu. Naravno da niko neće slati pakete na te MAC adrese, ali će one da zauzmu koristan prostor u bridžing tabeli. Zapravo i cilj ove maliciozne aktivnost je da se sa velikim brojem lažnih MAC adresa bridžing tabela prepuni nepotrebnim podacima i time onemogući normalan rad bridža. Tom prilikom bridž će raditi dodatni posao odražavanja bridžing tabele, uz konstantno sprovođenje *Flooding* proces.

3.6 Zbogom koaksijalci

U prvobitnoj Eternet mreži svi uređaji su se povezivali na koaksijalni kabal, koji je činio jedinstveni deljeni medijum po kome se informacije prenose putem elektromagnetnih signala. Da bi se omogućile brzine prenosa podataka od 10 Mbps, što je za to vreme bilo izuzetno veliko, bilo je neophodno da se zadovolje dosta stroge elektromagnetne karakteristike prenosnog medijuma. Sam kabl je fabrički izrađen tako da zadovoljava ove karakteristike, ali se to mora zadržati i prilikom povezivanja svakog uređaja.

Prvobitni Eternet je standardizovan po oznakom 10Base5, gde broj 10 označava brzinu prenosa u jedinici Mbps, broj 5 označava maksimalnu dužinu koaksijalnog segmenta izraženu u stotinama metara, a oznaka „*Base*“ modulaciju signala. Zbog fizičkih gabarita korišćenog koaksijalnog kabla ova varijanta je poznata i kao debeli Eternet (eng. *Thick Ethernet*). Kabl je zaista bio dosta debeo, što ga je činilo krutim i nepraktičnim za postavljanje u objektima. Uređaji se ipak nisu direktno povezivali na njega, već preko dodatnih tanjih i savitljivijih kablova, koji su se na koaksijalni kabl povezivali posebnim konektorima, tzv. AUI konektor. Tipična realizacija mreže je bila takva da se debeli koaksijalni kabl postavljao po hodnicima objekata, a odakle su vodili pojedinačni kablovi do povezanih uređaja, a čija je maksimalna dužina bila ograničena na 50m. Iako je time omogućeno da se lakše dođe do prostorija gde se nalaze računari, dodatni kablovi i konektori su povećavali cenu realizacije mreže, kao i mogućnost narušavanja celokupnih elektromagnetnih karakteristika mreže.

Vrlo brzo, pojavila se nova verzija Eterneta u oznaci 10Base2, koja je bila bazirana na dosta tanjem i jeftinijem koaksijalnom kablu, a koja je po analogiji nazvana tanki Eternet (eng. *Thin Ethernet*). Ovi kablovi su bili dosta savitljiviji, lakši za instalaciju i mogli su lako da uđu i postave u radni prostor, gde su se uređaji direktno povezivali putem jednog konektora sa tri priključka, tzv. „T-konektora“. Na mestu povezivanja uređaja, tanki koaksijalni kabl se fizički prekidao i nastavljao preko dva priključka T-konektora, dok se treći priključak povezivao direktno na uređaj. Na taj način koaksijalni kabl se u nastavcima, preko T-konektora protezao od jednog do drugog uređaja u mreži, ali tako da moraju da se zadrže sve propisane elektromagnetne karakteristike celokupne mreže. To je bilo jednostavno i jeftino za realizaciju, ali se ubrzo kao

veliki praktični problem ispoljilo očuvanje potrebnih elektromagnetnih karakteristika. Kablovi u radnom prostoru nisu bili fiksni i dolazili su do radnih stolova, gde su bili izloženi fizičkim pomeranjima, što je izazivalo njihova oštećenja. Ovo je izazivalo narušavanje elektromagnetnih karakteristika, što je neretko dovodilo do prestanka rada celokupne mreže. Oštećenja su obično bila na spoju unutar konektora, neprimetna sa spoljne strane. Detektovanje ovih prekida je podrazumevalo proveru svih konektora u mreži, često uz ponovno prespajanje na mestima gde se sumnjalo na problem. Kako je mreža za korisnike bila sve značajnija, ovakvi česti i dugotrajni prekidi su se sve manje tolerisali.

Tehnologija realizacije mrežnih uređaja i kablova ponovo je našla rešenje, i ovog puta bez izmene samog Eterneta protokola. Pošto je svaki pojedinačni priključak bio potencijalni izvor problema, bilo je neophodno da se on izoluje i učini nezavisnim od ostatka mreže. Ovo se postiglo prelaskom sa tzv. bas topologije na zvezdastu topologiju, u kojoj se iz jednog centra svaki uređaj povezuje posebnim kablom. Pošto je bilo neophodno zadržati funkcionalnost deljenog medijuma i CSMA/CD tehnologije, kablovi su se u centralnoj tački povezivali na poseban uređaj, tačnije svaki kabl na posebni port ovog uređaja, koji je nazvan hab (eng. *hub*). U funkcionalnom smislu, hab radi isto što i ripiter – signale sa jednog porta dekoduje u binarne podatke i „osvežene“ prenosi na sve ostale portove. Budući da hab ima više portova (obično 8, 16 ili 24), on se može smatrati za „višeportni ripiter“.

Kablovi koji realizuju direktno povezivanje uređaja na hab nisu više koaksijalni, već kablovi sa više provodnika („višežilni“), gde se prenos u jednom smeru (eng. *transmit*) obavlja preko dva provodnika, tzv. „parica“, a prenos u dugom smeru (eng. *receive*) preko druga dva provodnika. Elektromagnetne karakteristike ovih provodnika su optimalne, ako su ove parice upredene, pa se ovi kablovi nazivaju „upredene parice“. Ovi kablovi mogu da imaju i dodatni omotač od metalne folije ili mrežice, čime se elektromagnetne karakteristike dodatno poboljšavaju. Ipak, daleko češće se koriste kablovi koji imaju samo PVC izolaciju, pa se na engleskom nazivaju *Unshielded Twisted Pair*, a mnogo poznatiji po skraćenici UTP. Čak i bez metalnog „oklopa“, UTP kablovi mogu uspešno da prenose signale između haba i uređaja na rastojanju i do 100m.

U odnosu na koaksijalne kablove i bas topologiju, ako postoji problem u nekom kablu ili konektoru, signal će da bude izobličen i hab neće moći da prepozna binarne podatke koji mu pristižu, pa ih neće propagirati na ostatak mreže. Time je problem izolovan samo na jednoj vezi između haba i uređaja, što se jednostavno detektuje i otklanja. Ova prednost je bila toliko značajna, da je trošak dodatnog kabliranja u zvezdastoj topologiji mala cena u odnosu na dobit. Štaviše, ovaj princip kabliranja nije bio nov – u poslovnim zgradama su već su postojale telefonske instalacije koje su iz jednog čvora završavale u radnom prostoru. Iako su se u pojedinim slučajevima mogli koristiti i ovi telefonski kablovi, potreba je bila takva da su novi UTP kablovi ubrzo postali obavezan deo svih poslovnih objekata, ne samo novih, već i postojećih. Tipična instalacija je podrazumevala jednu ili više koncentracija kablova po spratu, odakle su se UTP kablovi prostirali do obližnjih prostorija, poštujući ograničenje od 100m, što je činilo tzv. „horizontalno kabliranje“. U ovim lokalnim „spratnim“ koncentracijama su se postavljali habovi, koji su se međusobno povezivali „po vertikali“ do zajedničke tačke u objektu, što je činilo tzv. „vertikalno kabliranje“, odnosno kičmu mreže (eng. *backbone*). Ovi principi realizacije zvezdaste mreže korišćenjem UTP kablova se nazivaju „strukturno kabliranje“, a objedinjeni su međunarodnim standardom ISO/IEC 11801 [9], odnosno njegovim „američkim“ ekvivalentnom ANSI/TIA-568 [10].

Potrebno je istaći da se tom prilikom Eternet nije menjao – zadržan je isti format i veličina paketa, principi detekcije kolizije, brzine prenosa i druge karakteristike. Ipak, zbog izmenjenih

fizičkih karakteristika (konektora, utičnica i kablova), Eternet u ovom obliku je označen sa 10BaseT. Ova kompatibilnost je omogućila čak i da se kombinuju različite tehnologije kabliranja – pojedini habovi su imali i T-konektor, koji im je omogućavao povezivanje na tanki koaksijalni kabl. U neki slučajevima se tanki Eternet koristio za kičmu mreže, na koju su se povezivali habovi, od kojih su se prostirali UTP kablovi po obližnjim prostorijama.

3.7 Zbogom kolizijo

Strukturno kabliranje i habovi su omogućili da problemi na fizičkom nivou budu lokalizovani, bez uticaja na ostatak mreže. Mreža je time postala još veća, a glavni problem koji limitira njeno korišćenje je i dalje ostao - kolizija. Srećom, strukturno kabliranje u zvezdastoj topologiji je omogućilo da se i problem kolizije lokalizuje. Podsetimo se da bridž upravo lokalizuje koliziju na nivou jednog segmenta, koji je u tom slučaju bio koaksijalni kabl. Kod strukturnog kabliranja, svaki UTP kabl se može tretirati kao poseban segment, a kolizija se može lokalizovati ako hab dodatno sprovodi i funkciju bridža. Hab tada više nije hab, već nova vrsta uređaja koji je nazvan svič (eng. *switch*).

Svič ima više portova kao i hab, ali u funkcionalnom smislu on predstavlja bridž – na isti način sprovodi sve funkcije bridža: *Forwarding*, *Filtering*, *Flooding*, *Learning* i *Aging*. Kao što hab predstavlja „višeportni ripiter“, tako se i svič može smatrati za „višeportni bridž“.

Fizička sličnost sa habovima je omogućila da se habovi jednostavno zamene sa svičevima. Kao i svaki novi uređaj, svičevi su isprva bili dosta skupi, pa je racionalno bilo da se prvo zameni hab u centralnom čvorištu, čime se kolizija u mreži svodi na pojedine celine. Potpunom zamenom habova sa svičevima kolizija je svedena na nivo pojedinačnih veza – mogla je da se javi samo, ako se na vezi pojave paketi istovremeno u oba smera. Ovaj režim prenosa podataka na direktnim vezama između dva učesnika, kada se paketi u jednom trenutku mogu slati samo u jednom smeru, naziva se poludupleksni režim (eng. *half-duplex mode)*.

Podsetimo se da za razliku od koaksijalnih kablova, UTP kablovi signale u odlaznom i dolaznom smeru prenose po fizički razdvojenim provodnicima (paricama). Time se omogućava da se oni mogu prenositi istovremeno u oba smera, u režimu koji se naziva potpuno dupleksni (eng. *full-duplex*). Mrežne kartice takođe treba da podržavaju ovaj režim rada, u kom slučaju je mogućnost nastanka kolizije potpuno eliminisana.

Podsetimo se da je Eternet nastao na CSMA/CD principima i deljenom medijumu, gde pod opterećenjem kolizija limitira propusni opseg na svega 18.4% ukupnog kapaciteta. Eliminacijom kolizije propusni opseg ne samo što je mogao da se iskoristi do 100% kapaciteta, već u potpunom dupleksnom režimu efektivno i do 200%, uzimajući u obzir prenos u oba smera. Migriranjem tehnologije na bridževe, strukturno kabliranje i habove, do potpune zamene sa svičevima, kolizija i svi bazični principi na kojima je Eternet nastao su nestali, dok je Eternet ostao čak potpuno neizmenjen. Jedino što se menjalo je brzina prenosa.

3.8 Need for speed

Osim stalne potrebe korisnika za povećanjem brzine prenosa podataka, veća brzina Eternet mreže bi takođe smanjila i verovatnoću nastanka kolizije. Brzine prenosa od 10 Mbps, odnosno vreme izlaska jednog bita na mrežu od 0.1 μs, zbog uslova za nastanak kolizije dovele su do ograničenja maksimalne veličine mreže i minimalne veličine okvira na 64 bajta. Povećanjem brzine prenosa uz zadržavanje istih principa detekcije kolizije, ova ograničenja su se morala redefinisati.

A povećanje je bilo značajno – čak 10 puta, čime se ostvarila brzina prenosa od 100 Mbps, što je označeno sa „*Fast Ethernet*“, a standardizovano po brojem IEEE 802.3u, a u varijanti sa UTP kablovima pod oznakom 100BaseTX. Sa druge strane, *bit-time* vreme je 10 puta manje i iznosi 0.01 μs, pa se i rastojanje u mreži mora smanjiti. Sa druge strane, ripiteri i habovi su u međuvremenu postali brži, pa je i vreme propagacije signala kroz njih moglo da se smanji odnosu na ranije uspostavljeno maksimalno vremena od 3μs.

Tom prilikom se razlikuju dve vrste ripitera (u ovom kontekstu habovi se takođe smatraju za ripitere):

- Ripiteri prve klase (eng. *Class I repeaters*), koji na povezvanim portovima mogu da kombinuju različite varijante Eterneta sa različitim tehnikama kodovanja binarnih podataka,
- Ripiteri druge klase (eng. *Class II repeaters*), koji omogućavaju povezivanje samo istorodnih varijanti Eterneta.

Ova razlika u dodatnom poslu koju sprovede ripiteri prve klase utiče i na maksimalno vreme propagacije signala, koje iznosi 0.7 μs, dok u ripiter druge klase nešto brži, sa vremenom 0.46 μs.

Uzimajući navedene parametre u obzir, ispostavlja se da se minimalna veličina okvira od 64 bajta može zadržati, ako se smanji dijametar u mreži na 200m i ograniči broj ripitera, i to na 1 ripiter prve klase ili 2 ripitera druge klase. U praktičnoj realizaciji to je tipično podrazumevalo samo jedan hab sa UTP vezama do 100m. Ali veće brzine prenosa zahtevaju još strožije kriterijume za različite parametre elektromagnetnih karakteristika, pa su i UTP kablovi morali da se unaprede da mi mogli da zadrže maksimalna rastojanja do 100m. Zahtevane karakteristike su standardizovane u tzv. kategorije kablova: Cat-3, Cat-5 itd.

Ali zašto stati na ovim brzinama? Uz ponovno povećanje brzine od 10 puta, dostiže se brzina od 1000 Mbps, odnosno 1Gbps, ali i bit-time od 0.001 μs. Ovog puta, da bi se zadržala maksimalna dužina povezanih uređaja od 100 m, povećana je minimalna veličina okvira sa 64 bajta na 512 bajta. Dalje se ispostavilo da UTP kablovi u kategoriji Cat-5 nisu mogli da garantuju ispravan prenos do 100m zbog karakteristika jednog parametra (smetnji koje signal pri slanju stvara na prijemnom signalu, koji je znatno slabijeg intenziteta). Ova karakteristika je unapređenja u kategoriji pod oznakom Cat-5e (slove „e“ potiče od engleske reči „*enhanced*“ - unapređenje).

Prelaskom na svičeve, ograničenje minimalne veličine okvira su izgubile smisao, ali su one ipak zadržane zbog kompatibilnosti za habovima. Sa druge strane, UTP kablovi su nastavili da se razvijaju u kategorijama Cat-6, Cat-7, Cat-8 itd.

Eternet je i dalje nastavio da povećava brzinu u gigabitskoj skali: 10Gbps, 40Gbps, 100Gbps itd. Ovog puta je definitivno napušten koncept kolizije i habova, pa čak i UTP kablova – koriste se samo svičevi povezani na optičke kablove. Druga specifičnost Eterneta na ovim izuzetno velikim brzinama je što se ne koristi za masovno povezivanje krajnjih uređaja (zbog cene, ali i stvarnih potreba), sa izuzetkom servera posebne namene, već se koristi za direktno međusobno povezivanje glavnih svičeva na kičmi mreže. A zbog korišćenih optičkih kablova, ove veze mogu da budu i na dužinama od više desetina ili stotina kilometara.

4. Današnje Eternet mreže

Čitalac koji je pročitao prethodno poglavlje može da se upita zašto je bilo potrebno detaljno objašnjavati principe, uređaje i način realizacije Eternet mreže koji se odavno ne koristi – kolizija, ripiteri, habovi, bridževe, koaksijalni kablovi itd. I pored opravdanog razloga koji se krije u narednom poglavlju, Eternet to ipak zaslužuje, jer odlično demonstrira principe i proces uspešnog inženjerskog razvoja. Dok su ostale tehnologije bile opterećene ambicioznim zahtevima, složenom realizacijom i skupim uređajima, ključna prednost Eterneta je ležala u njegovoj jednostavnosti, koja je omogućila prilagođavanje novim potrebama. Eternet se suštinski nije ni menjao, već su se menjala prateća tehnička rešenja koja su ga implementirala (topologija, uređaji, kablovi). Ni format paketa se nije menjao, što je omogućilo kompatibilnost i interoperabilnost sa prethodnim varijantama i postepenu primenu kod korisnika. Eternet je tako evoluirao, rešavajući problem skalabilnosti uvođenjem svičeva, stabilnog rada uvođenjem zvezdaste topologije, povezivanje dve tačke (eng. *point-to-point*) na velikim rastojanjima putem optičkih kablova prevazilazeći okvire lokalnih računarskih mreža.

Ovaj proces razvoja pratećih tehničkih rešenja se nastavio i dalje, a najvažniji tehnologije se prikazuju u nastavku poglavlja.

4.1 Spanning tree protocol

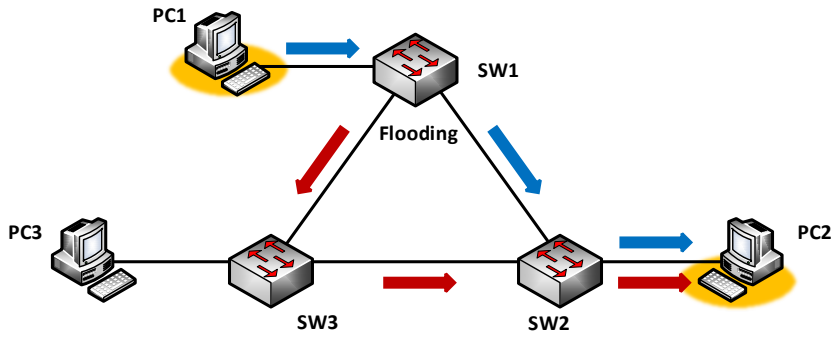
Prethodno je istaknuta prednost zvezdaste topologije i UTP kablova u odnosu na realizaciju mreže u bas topologiji putem koaksijalnih kablova. Ipak, zadržan je i jedan veliki nedostatak – u slučaju prekida pojedine veze, ili još dramatičnije, čvora u topologiji (npr. prestanak rada sviča), mreža se deli na više nepovezanih celina. Razlog tome je što ne postoje redundantne grane u topologiji, koje bi omogućile automatsko aktiviranje alternativnih veza. U slučaju korišćenja habova, jasno je da bi ove redundantne grane izazvale nekontrolisano širenje, odnosno kruženje paketa, ali ovaj problem postoji i kod svičeva. Čak i kod bridževa i koaksijalnih segmenata je postojala potreba da se obezbedi redundantnost veza u mreži. Iz toga doba datira i problem, ali i rešenje koje uspostavljeno, a koje se koristi i danas, tzv. *Spanning Tree Protocol*, poznat po skraćenici STP.

Problem redundantnih veza

Imajući u vidi način na koji rade svičevi, pogledajmo najpre koji problemi mogu da se jave u slučaju da postoje redundantne veze i to na najjednostavnijem slučaju sa tri sviča povezana u trougao i po jedan računar na svakom od njih.

Dupliranje paketa

Slika 4.1 prikazuje slučaj kada uređaj PC1 šalje paket namenjen za PC2, gde plave strelice označavaju regularno slanje paketa od sviča SW1 do sviča SW2. Ipak, u slučaju kada svič SW1 u svojoj bridžing tabeli nema MAC adresu uređaja PC2, on će da sprovodi flading proces, prosleđujući kopiju paketa na drugi, redundantni link. Ovaj pakete će da dođe do sviča SW3, a zatim se preko sviča SW2 prosleđuje do odredišnog uređaja PC2. Odredišni uređaj je tako jedan isti paket primio dva puta, bez mogućnosti da se prepozna ova greška.

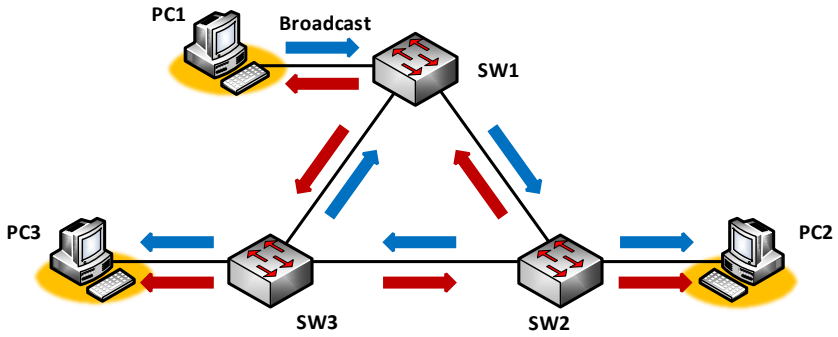


Slika 4.1. Slučaj dupliranja paketa

„Oluja“ brodkast paketa

Još drastičniji slučaj se javlja kada neki uređaj pošalje brodkast paket, odnosno paket sa odredišnom brodkast adresom, koji je namenjen svim uređajima na mreži, pa ga svičevi prosleđuju na sve izlazne portove. Paket posla od uređaja PC1 će pa dođe do sviča SW1, a kopija paketa će da se prosledi na obe strane, odnosno i do sviča SW2 i do sviča SW3, koji ga dalje prosleđuju na preostale portove. Paket će doći do svih uređaja u mreži, što je i bila namera, ali će pristići po dva puta, na slici označeno plavim i crvenim strelicama. Ovi paketi će se vratiti do sviča SW1 i zatim stići do izvorišnog uređaja, što takođe nije željeno ponašanje.

Ali pravi problemi tek nastaju – svič SW1 će ove pakete ponovo proslediti do svičeva SW2, odnosno SW3, čime se formira petlja beskonačnog kruženja paketa u oba smera. Prolaskom kroz svaki svič u ovoj petlji, paketi će da se multipliciraju i na ostale portove, i time neprekidno pristižu do svih uređaja u mreži. Pojavom drugih brodkast paketa, ne samo što će svi uređaji da trpe opterećenje, već će i cela mreža u ovom delu postati praktično neupotrebljiva za prenos drugih paketa. Ovaj proces je toliko intenzivan da se naziva oluja brodkast paketa (eng. *broadcast storm*).

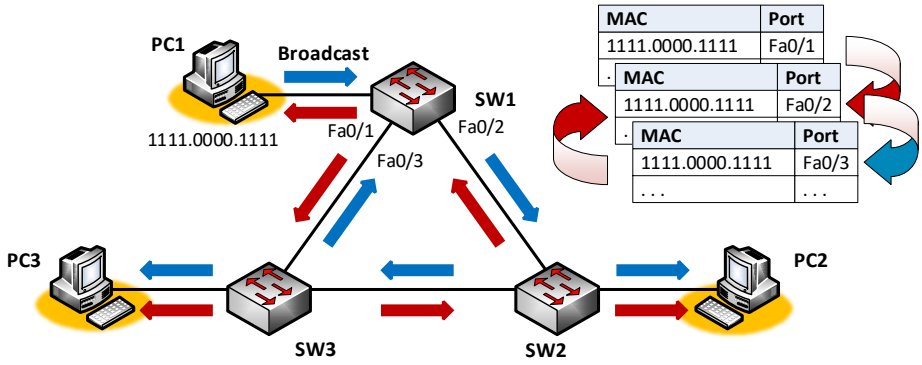


Slika 4.2. „Oluja“ brodkast paketa

Nestabilnost bridžing tabele

Brodkast paketi koji u prethodnom slučaju kruže u petlji u oba smera, ne samo što opterećuju svičeve, već ih i dodatno bune pri *learning* procesu koji se stalno sprovodi. Ovi paketi su potekli od jednog uređaja i imaju njegovu MAC adresu kao izvorišnu. Prilikom dolaska u svič iz jednog smera (npr. plava strelica), svič će da zaključi da se izvorište nalazi u delu mreže koji dolazi sa ulaznog porta. Neposredno nakon toga će pristići kopija ovog paketa, ali iz drugog smera (crvena strelica), pa svič menja prethodno naučenu informaciju i upisuje novi port kao put koji vodi do izvorišta. Proces se nastavlja naizmeničnim pristizanjem ovih paketa, i stalnim

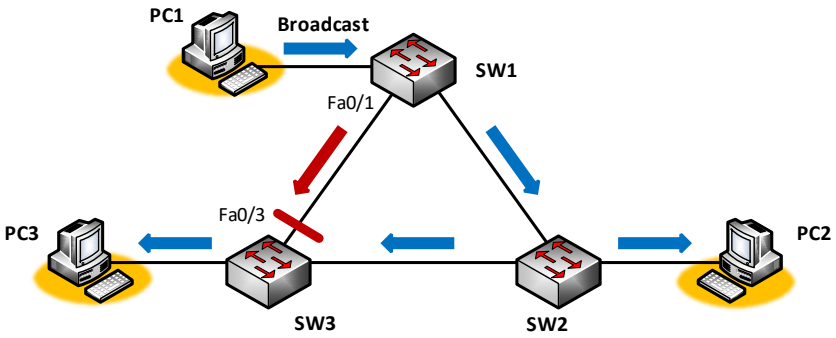
promenama porta u bridžing tabeli. U slučaju sviča SW1 ove informacije su čak i netačne, a neki drugi paket namenjen za izvorišni uređaj PC1 će biti poslat na pogrešnu stranu, što će dovesti do njegovog lutanja.



Slika 4.3. Nestabilnost bridžing tabele

Princip rada STP

Iako su nam redundantne veze poželjne i potrebne, prethodni primeri ilustruju pogubnost fizičkih petlji u topologiji. Rešenje je da se one zabrane, ali na logičkom nivou, a da se aktiviraju samo u slučaju potrebe, odnosno u slučaju prekida primarnih veza. Eternet je suviše jednostavan da reši ovaj problem, što nije nedostatak, dokle god to može da se reši na drugi način. I ovog puta ovu dodatnu logiku unose svičevi, tako što prepoznaju petlje i blokiraju svoje portove na vezama koje čine petlje (Slika 4.4). Da bi to bili u mogućnosti, svičevi sprovode STP protokol.



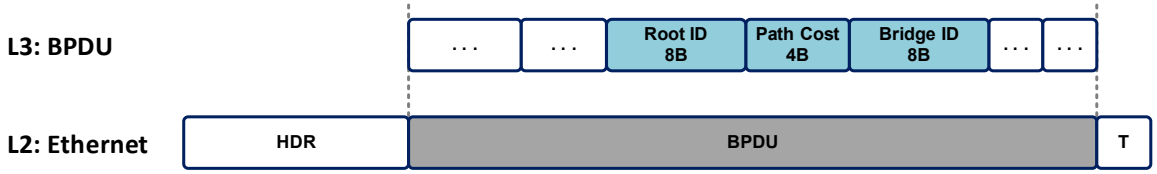
Slika 4.4. Ukidanje petlji na logičkom nivou

Svič lako može da blokira svoj port, tako što ne prima ili odbaci pristigle pakete. Daleko veći problem je da se petlje prepoznaju, da se one prekinu na pravom mestu tako da se može zadržati povezanost svih delova mreže, odnosno da mreža ostane jedinstvena, i to u bilo kom slučaju, bez obzira koliko je mreža složena i koliko petlji poseduje. Ovaj posao treba da radi svaki svič i to iz svoje pozicije u mreži, ali na način da se uspostavi usaglašeno i ispravno stanje u celoj mreži. Da bi to radili, svičevi ne moraju da poznaju strukturu cele mreže, ali moraju da imaju odgovarajuće informacije koje će im omogućiti da uspostave ispravno stanje. Ove informacije se razmenjuju između svičeva putem posebnih poruka. Uopšte, razmena informacija između komunikacionih uređaja u računarskim mrežama u cilju obavljanja određenog zajedničkog posla uspostavlja se putem komunikacionog protokola. STP je samo

jedan od mnogih komunikacionih protokola koji se danas koriste i prvi koji se detaljno prikazuje u ovom udžbeniku.

Poruke koje se razmenjuju putem STP protokola se nazivaju BPDU poruke (eng. *Bridge Protocol Data Unit*)[4]. One se enkapsuliraju u Eternet okvire i prenose se samo do susednih svičeva. Izvorišna adresa u Eternet okviru je MAC adresa porta sviča koji šalje poruku, ali se problem javlja kod odredišne adrese – to bi trebalo da bude MAC adresa susednog sviča, ali se postavlja pitanje kako svič koji šalje zna tu MAC adresu.

Srećom, postoji i drugi način da se pošalje poruka, a da se ne zna tačna adresa odredišnog uređaja – korišćenjem brodkast ili multikast adresa. Pakete sa brodkast odredišnom adresom prihvataju svi uređaji, dok će one sa multikast adresom da prihvate samo uređaji koji znaju da treba da prate komunikaciju na toj specifičnoj multikast adresi. U slučaju STP protokola koristi se fiksna multikast adresa 0180.C200.0000. Svičevi koji sprovode STP prate saobraćaj na ovoj adresi i prihvataju pristigle pakete, dok svi ostali uređaji koji ne sprovode STP (npr. računari) ne prepoznaju ovu adresu i odbacuju pakete.



Slika 4.5. BPDU poruke STP protokola se prenose preko Eternet okvira

U zavisnosti od svoje namene, postoje tri vrste BPDU poruka, od kojih svaka ima strogo definisani format:

- Konfiguraciona poruka (eng. *Configuration* BPDU, skr. CBPDU)
- Notifikacija promene topologije (eng. *Topology Change Notification*, skr. TNC)
- Potvrda promene topologije (eng. *Topology Change Acknowledgment*, skr. TCA)

Informacije o stanju u mreži, iz kojih se zaključuje da li postoje petlje, se prenose konfiguracionim porukama. Tom prilikom se uvode određeni parametri i metrike. Najpre, svičevi treba da uspostave svoje identifikatore. Oni moraju da budu jedinstveni, što se postiže izborom jedne MAC adrese sviča[5]. Kao što ćemo videti nešto kasnije, korisno je da svič poseduje i određeni prioritet u poslu koji obavlja, što se takođe pridružuje identifikaciji, i to sa većom težinom u odnosu na MAC adresu.

Identifikacija sviča, tzv. *Bridge ID*, se sastoji od osam bajtova:

- 2 bajta prioriteta, koji imaju veću težinu (početni bajtovi) i mogu se menjati;
- 6 bajtova MAC adrese, koji imaju manju težinu i ne mogu se menjati.

Pravilo je da manja vrednost identifikacije određuje veći prioritet.

STP protokol ne samo što treba da omogući svičevima da prepoznaju i uklone petlje, već i da to učine na optimalan način, tako što blokiraju veze sa manjim brzinama, a zadržavaju veze

[4] U domaćem inženjerskom žargonu ustaljen je naziv koji se izgovara spelovanjem „BPDU“, odnosno: „bi-pi-di-ju“
[5] Osim što svaki port sviča ima svoju MAC adresu, svič ima određeni broj dodeljenih „unutrašnjih“ MAC adresa koje koristi za različite logičke identifikacije.

sa većim brzinama. Zato se uvodi metrika koja se izvodi iz brzine veza, tzv. cena (eng. *cost*). Cena je bolja kada je niža, pa se računa obrnuto proporcionalno u odnosu na brzinu – vrednost 1000 Mbps se deli sa aktuelnom brzinom veze. Brzina veze je određena brzinom na kojoj port sviča radi, pa se uvodi **cena porta**, tzv. *port cost*. Na ovaj način portovi od 10 Mbps će imati cenu 100, za Fast Eternet portove cena će biti 10, a za gigabitne portove dobija se cena 1. Problem je nastao pojavom Eternet na većim brzinama, 10 Gpbs i više. Tada su cene revidirane i uspostavljene su fiksne vrednosti, koje su prikazane u Tabeli 1. Napomenimo da brzina na oba porta veze koja povezuje dva sviča mora biti usaglašena, pa oni imaju istu cenu.

Put kojim se prenose paketi između dve proizvoljne tačke u mreži naziva se putanja, a STP uvodi i **cenu putanje** (eng. *path cost*), koja se sastoji od zbira cena svih veza koje čine putanju.

Brzina	Inicijalne cene	Revidirane cene
10 Mbps	100	100
100 Mbps	10	19
1 Gbps	1	4
10 Gbps	1	2

Tabela 1.

Objasnimo najpre kako se sprovodi STP globalno, kao „misaoni proces“, posmatrajući mrežu iz „ptičije perspektive“, a zatim i kako to sprovode svičevi iz svoje tačke u mreži. Formalno posmatrano, proizvoljnu topologiju mreže koja može da sadrži petlje potrebno je redukovati na topologiju stabla, koja je po definiciji potpuno povezana i bez petlji. Topologija stabla se često predstavlja tako da poseduje tzv. koren stabla, iz koga se granaju ostali delovi. Pri tome koren stabla može biti bilo koji čvor.

STP funkcioniše tako što bira jedan čvor u mreži i od njega širi strukturu stabla bez petlji - *spanning tree*. U strukturi stabla od svakog čvora može postojati samo jedna putanja do korena, pa se stoga zadržavaju grane na putanji do korena stabla koja ima najmanju cenu, dok se grane koje pripadaju drugim putanjama odbacuju, jer one unose petlje.

Gore opisani proces studenti mogu jednostavno da sprovedu čak i na ispitu, biranjem bilo kog čvora u mreži i „razvlačenjem stabla“ od njega uzimajući u obzir brzine veza i cene koje su time određene. Za svičeve je to dosta teže, jer sve što oni imaju od informacija je ono što dobiju kroz BPDU poruke.

4.2 STP iz pozicije svičeva

Svičevi sprovode STP protokol u četiri koraka:

1. Svi svičevi treba da se usaglase koji svič predstavlja koren stabla – tzv. *Root* svič[6].
2. Svaki svič određuje samo jedan port koji ima najmanju cenu putanje do *Root* sviča, tzv. *Root* port.

[6] Originalni naziv iz teksta standarda, a koji se i danas često koristi, je *Root Bridge*, budući da je STP nastavo u doba bridževa, pre pojave svičeva.

3. Na svakoj od preostalih veza određuje se port koji ima najmanju cenu putanje do *Root* sviča – tzv. *Designated* port.
4. Portovi sviča koji nisu ni *Root* ni *Designated* se blokiraju – tzv. *Blocked* port.

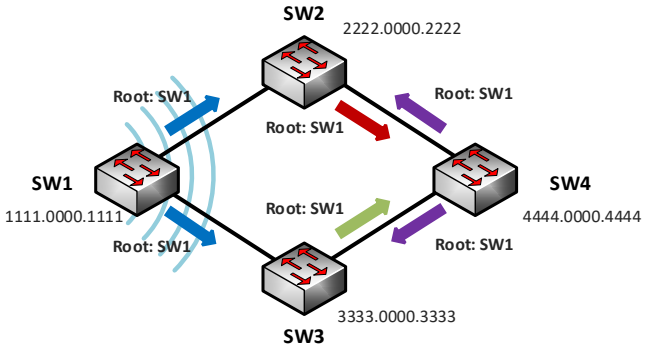
Izbor *Root* sviča

Pravilo je da *Root* svič postaje svič sa najmanjom identifikacijom (*Bridge ID*), a razmenom BPDU poruka svi svičevi treba da utvrde koji je to svič.

Konfiguracione BPDU poruke sadrže polje „*Bridge ID*“, koje sadrži identifikaciju sviča koji je poslao poruku, kao i polje „*Root ID*“ koje sadrži identifikaciju sviča za koji svič koji šalje poruku smatra da je *Root* svič. Inicijalno, pri početku rada (uključenje ili restart), svič nije dobio ni jednu BPDU poruku, nema saznanja o ostalim svičevima i delegiraće sebe za Root sviča, postavljanjem svoje identifikacije u polje *Root* ID i slanjem ovakve BPDU poruke na sve izlazne portove, a samo direktno povezani svičevi primaju ove poruke.

Ako posmatramo isti svič, on će ubrzo početi da dobija BPDU poruke od dugih direktno povezanih svičeva, a koje sadrže identifikacije ostalih svičeva, uključujući i određene vrednosti u *Root* ID polju. Ako je ova vrednost veća od identifikacije posmatranog sviča, on će nastaviti da oglašava sebe kao *Root* svič u BPDU porukama koje nastavlja periodično da šalje svake 2 sekunde. Ako je vrednost u *Root* ID polju primljene poruke manja od identifikacije sviča, svič tada zaključuje da postoji bolji kandidat za *Root* svič, zadržava njegovu identifikaciju u svojim BPDU porukama koje nastavlja da šalje susednim svičevima, i to neposredno svaki put kada dobije takvu BPDU poruku.

Posmatrano na nivou cele mreže, vrlo brzo će u svim BPDU porukama u *Root* ID polju da se ustali najmanja identifikacija sviča koji postaje *Root* svič. *Root* svič nastavlja da šalje svoje BPDU poruke na 2 sekunde, čime se „diktira ritam“ propagacije poruka – ostali svičevi po dobijanju ovih poruka, kreiraju i šalju svoje BPDU poruke, zadržavajući identifikaciju *Root* sviča u *Root* ID polju.



Slika 4.6. Izbor Root sviča i oglašavanje u stacionarnom stanju

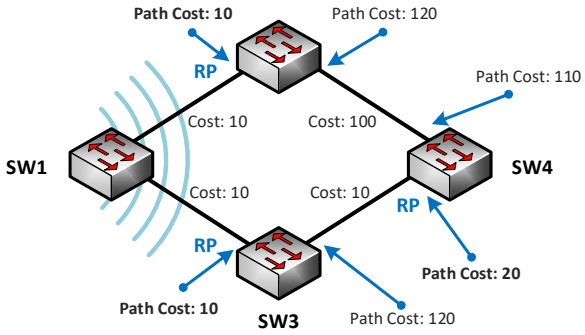
Izbor *Root* porta

Efekat prethodno opisane propagacije konfiguracionih BPDU poruka, koji inicijalno potiču od *Root* sviča, je da prijem ovakve poruke na port sviča označava da od tog porta postoji putanja koja potiče do *Root* sviča. Ako svič dobije konfiguracionu BPDU poruku i na drugi port, to znači da postoji i druga putanja do *Root* sviča, a time i petlja u mreži koju treba razrešiti. Drugim rečima, svič treba da izabere samo jedan port koji vodi do *Root* sviča, i to port na putanji sa najmanjom cenom.

Stoga konfiguracione BPDU poruke poseduju i polje „*Path cost*“ koje nosi informaciju o ceni putanje do *Root* sviča. Vrednost ovog polja se akumulira prilikom propagacije poruke kroz mrežu na sledeći način:

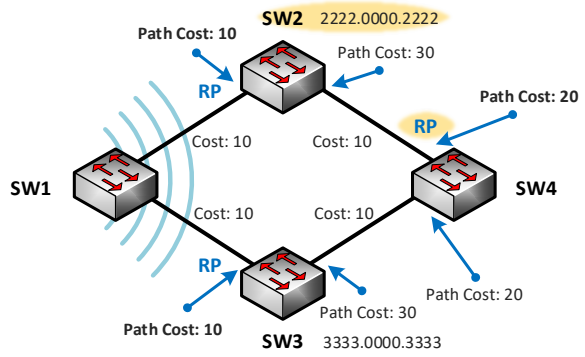
- Root svič šalje inicijalne BPDU poruke sa vrednosti nula u ovom polju (budući da je on Root svič).
- Svič po prijemu BPDU poruke, na dobijenu vrednost u „Path cost“ polju, dodaje cenu porta na kome je primio tu poruku. Ovako dobijena vrednost predstavlja cenu putanje koja od posmatranog sviča vodi do *Root* sviča i ta vrednost se postavlja u „*Path cost*“ polje novih BPUD poruka koje svič nastavlja da oglašava.
- Ostali svičevi „dalje od *Root* sviča“ će da dobiju ove poruke i na isti način uvećavaju cenu putanje do sviča i nastavljaju da oglašavaju kumuliranu vrednost.

Ako svič dobije BPDU poruke na dva ili više porta, izabraće port koji je daje **najmanju vrednost** parametra cena putanje. Ovaj port se naziva ***Root* port** jer na najbolji način vodi do *Root* sviča, pa će se on zadržati u STP stablu koje se uspostavlja. Budući da stablo ne sme da ima petlji, **svič može da poseduje samo jedan *Root* port**. Primetimo i to da *Root* svič nema *Root* port.



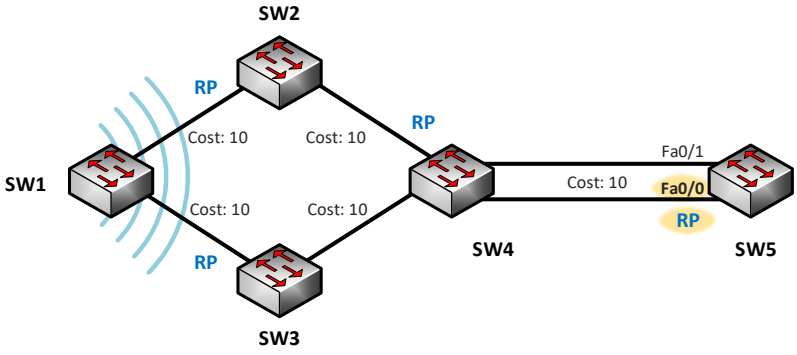
Slika 4.7. Izbor Root porta na sviču SW4 osnovu manje vrednosti parametra cena putanje

U slučaju da na dva porta pristižu poruke sa istom cenom putanje do *Root* sviča, i dalje se mora izabrati samo jedan port. Tada se za *Root* port bira onaj koji dobija BPDU poruku od sviča sa manjom identifikacijom (Slika 4.8)



Slika 4.8. Izbor Root porta na sviču SW4 za istu vrednost parametra cena putanje, kada se bira port prema sviču sa manjom identifikacijom (SW2)

Da li neki svič može da primi različite BPDU poruke od drugog sviča? Ovaj slučaj je moguć ako su svičevi vezani preko dve ili više paralelnih veza, što je dozvoljena situacija, ali se tada u STP stablu može zadržati samo jedna veza, a za *Root* port se bira port koji na sviču ima manju internu identifikaciju (Slika 4.9).

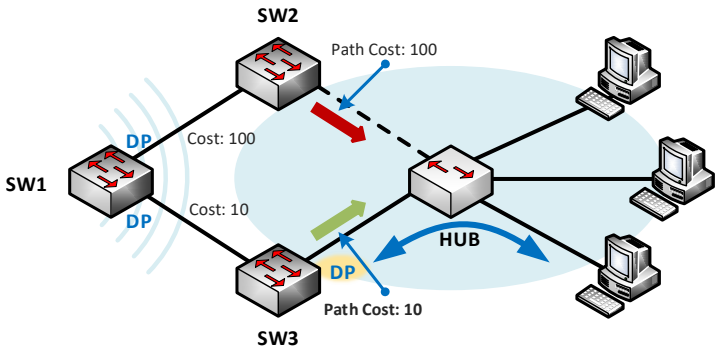


Slika 4.9. Izbor Root porta na sviču SW5 za istu vrednost parametra cena putanje i identifikacije sviča (SW4), kada se bira port koji ima manju internu identifikaciju

Izbor *Designated* portova

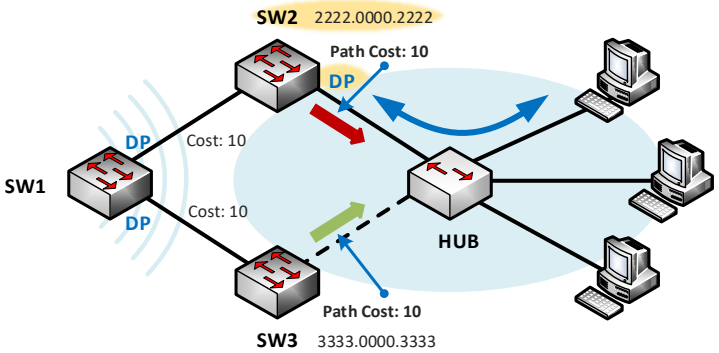
Smisao izbora *Root* porta je da se na sviču odredi samo jedna veza koja na najbolji način vodi do *Root* sviča. Nakon toga, na sviču se dalje posmatraju linkovi na preostalim portovima i cilj je da se i na svakom linku prepozna port koji na najbolji način vodi do *Root* sviča. I ovom prilikom se gleda cena putanje, a bira se port koji oglašava manju vrednost. Ovaj port se naziva *Designated* port. U ovom trenutku se uloga *Designated* porta najbolje može demonstrirati ako na vezi između dva sviča postoje i drugi uređaji, što se postiže postavljanjem haba (Slika 4.10). STP protokol treba ispravno da funkcioniše i u ovakvim situacijama, a uloga *Designated* porta je da omogući samo jednu vezu sa ovog segmenta do *Root* sviča, a time i sa ostatkom mreže.

Nešto kasnije ćemo videti značaj *Designated* porta i u slučaju direktne veze između dva sviča, i to u situacijama kada dolazi do promene topologije.



Slika 4.10. Izbor Designated porta na segmentu između svičeva SW2 i SW3 osnovu manje vrednosti parametra cena putanje

Designated port se posmatra na nivou segmenta, a i ovom prilikom može da bude samo jedan. U slučaju da portovi na segmentu oglašavaju istu vrednost *Path cost* atributa, tada se bira port na sviču koji ima manju identifikaciju, što je SW2 na primeru koji prikazuje Slika 4.11.



Slika 4.11. Izbor Designated porta na segmentu između svičeva SW2 i SW3 za istu vrednost parametra cena putanje, kada se bira port na sviču sa manjom identifikacijom (SW2)

Svi portovi *Root* sviča imaju najmanju moguću vrednost atributa cena putanje (0), pa su svi oni *Designated* portovi.

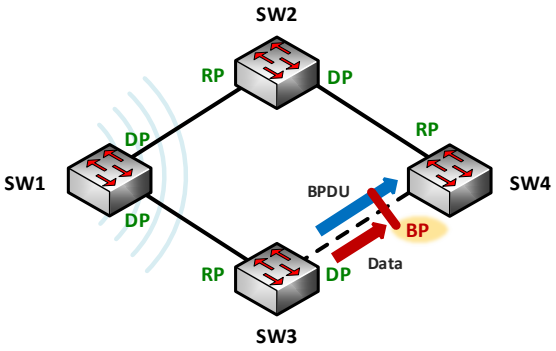
Blokiranje preostalih portova

Poslednji korak pri sprovođenju STP protokola je i najjednostavniji – svi preostali portovi koji nisu ni *Root* ni *Designated* se proglašavaju za blokirane portove (eng. *Blocked*). *Root* i *Designated* portovi se postavljaju u aktivno stanje u kome primaju i šalju sve pakete (*Forwarding*), dok se blokirani portovi postavljaju u neaktivno stanje u kome blokiraju pakete.

Prilikom blokiranja paketa postoji jedan izuzetak – jedini paketi koji prolaze kroz blokirane portove su dolazne BPDU poruke koje pristižu od *Designated* porta na povezanom linku, odnosno mrežnom segmentu. Ovo ponašanje je neophodno kako bi svič i dalje mogao da prati cenu putanje do *Root* sviča i preko blokiranog porta – u slučaju promene topologije (npr. usled prekida nekih drugih veza) ova veza može da postane najbolja koja vodi do *Root* sviča.

Blokirani portovi će zapravo da izazovu krajnji cilj STP protokola, a to je prekidanje petlji. Linkovi koji sadrže samo *Root* i *Designated* portove će da pripadaju STP stablu, što nije slučaj sa linkovima sa blokiranim portovima.

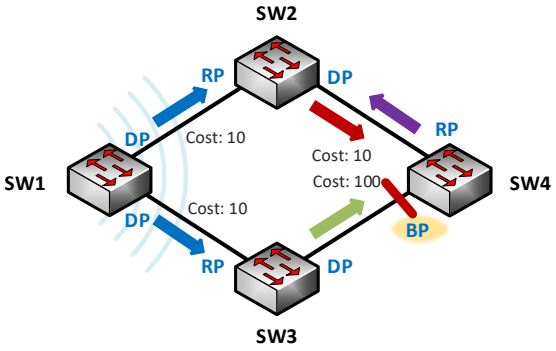
Primetimo da je ispravnije smatrati da STP ne ukida linkove u topologiji, već blokira pojedine portove. I na linkovima koji nisu u STP stablu postoji jedan (i samo jedna) *Designated* port, koji na link prosleđuje sve pakete. Ovi paketi pristižu do blokiranog porta, koji ih čak prihvata kao Eternet okvire, raspakuje njihov sadržaj, ali odbacuje sve sem BPDU poruka (Slika 4.12).



Slika 4.12. Odbacivanje svih paketa na blokiranom portu sem BPDU poruka

4.3 STP u stacionarnom stanju

Stacionarno stanje podrazumeva da u mreži nema promena, a da je STP uspostavio odgovarajuće stanje svih portova i time ukinuo petlje. Stacionarno stanje je potrebno i održavati, pa STP ipak ne prestaje sa radom. *Root* svič nastavlja da emituje konfiguracione BPDU poruke svake 2 sekunde. Ostali direktno povezani svičevi na svojim *Root* portovima primaju ove poruke, dodaju cenu portova na atribut cena putanje i reemituju ih kao nove poruke na svoje *Designated* portove, što je slučaj sa svičem SW2 na primeru koji ilustruje Slika 4.13. BPDU poruke se ne prenose na blokirane portove, tako da ih svič SW4 ne prenosi dalje do sviča SW3. Ipak, BPDU poruke se prihvataju i na blokiranim portovima, pa će svič SW4 da prihvati BPDU poruke (i to samo njih, ne i ostale) koje pristižu od sviča SW3 i da ih reemituje preko *Root* porta do sviča SW2. Ovo je korisna informacija za SW2, jer se time utvrđuje da postoji i druga fizička putanja do *Root* sviča (SW2-SW4-SW3-SW1), koja se može aktivirati u slučaju promene topologije, što se prikazuje u nastavku.



Slika 4.13. STP u stacionarnom stanju

Promena topologije

Promena topologije mreže nastaje ako pojedini link ili ceo svič prestane sa radom ili se uključi i aktivira. Ovo se detektuje izmenjenim vrednostima atributa cena putanje, novim BPDU porukama, ili pak izostankom BPDU poruka. Zadatak STP protokola je da i tada uspostavi stacionarno stanje bez petlji, a ovaj proces se naziva STP konvergencija.

Tokom konvergencije mreža je nestabilna, jer portovi menjaju svoja stanja. Tom prilikom je posebno bitno da se ni u jednom trenutku ne pojavi petlja. Promena stanja porta iz aktivnog, tzv. *forwarding* stanja u kome su *Root* i *Designated* portovi, u blokirano stanje deaktivira pripadajući link, što ne može da izazove petlju. Ipak, to nije slučaj pri aktiviranju prethodno blokiranih portova, pa se to mora sprovoditi na posebno pažljiv način koji zahteva određeno vreme.

STP stoga definiše određena vremena, tzv. tajmere (eng. *timer*) i to:

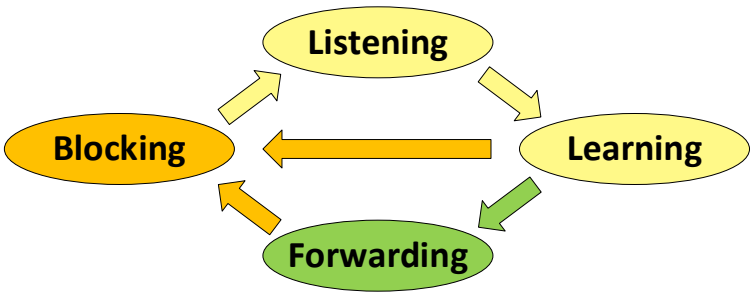
- *Hello* tajmer u trajanju od 2 sekunde, koji se odnosi na period oglašavanja konfiguracionih BPDU poruka od strane Root sviča.
- Tajmer maksimalna starost (eng. *Max Age*) u trajanju od 10 Hello tajmera, odnosno 20 sekundi, koji se odnosi na period koji će svič da čeka ako na nekom portu prestane da prima BPDU poruke, nakon čega će pokrenuti proces konvergencije.
- Tajmer dodatnog kašnjenja (eng. *Forward Delay*) od 15 sekundi, koji služi za postepeno aktiviranje prethodno blokiranih portova.

Ovi tajmeri moraju da budu usaglašeni i isti na svim svičevima u mreži, pa se njihove vrednosti prenose putem konfiguracionih BPDU poruka.

Već je istaknuto da portovi u aktivnom (*forwarding*) stanju propuštaju sve pakete u oba smera, dok se u blokiranom stanju odbacuju svi paketi u oba smera, osim dolaznih BPDU poruka. Osim toga, postoje i dva tranziciona stanja koja se mogu javiti tokom konvergencije. Ona se uvode radi postepene aktivacije portova iz blokiranih u aktivna stanje, a sve iz razloga izbegavanja petlji tokom konvergencije:

- *Listening* stanje, u koje se ulazi iz blokiranog stanja i provodi 15 sekundi (*Forward Delay* tajmer). U odnosu na blokirano stanje, port počinje da šalje BPDU poruke, čime obaveštava svičeve na ovim, još uvek neaktivnim vezama o cenama novih putanja do Root sviča.
- *Learning* stanje, u koje se ulazi iz *listening* stanja i provodi 15 sekundi (*Forward Delay* tajmer). U odnosu na prethodno stanju, na portu se prihvataju svi paketi, ali se oni ne prosleđuju dalje. Time svič samo uči MAC adrese na ovim portovima i popunjava bridžing tabelu, pripremajući se za aktivno stanje, čime se smanjuje potreba za kasnijim *flooding*.

Proces mogućih prelazaka između ovih stanja prikazuje Slika 4.14.



Slika 4.14. Proces promene stanja portova tokom konvergencije

Da bi se prilagodili promenama u mreži, svičevi najpre moraju da detektuju promene, a one mogu da nastanu prekidom ili dodavanjem pojedinih veza ili samih svičeva. Takođe je potrebno istaći da svičevi na svojim portovima mogu da detektuju fizičke prekide i dodavanja aktivacije direktnih veza.

O promeni topologije mreže usled prekida pojedinih veza svičevi se obaveštavaju kroz dva koraka:

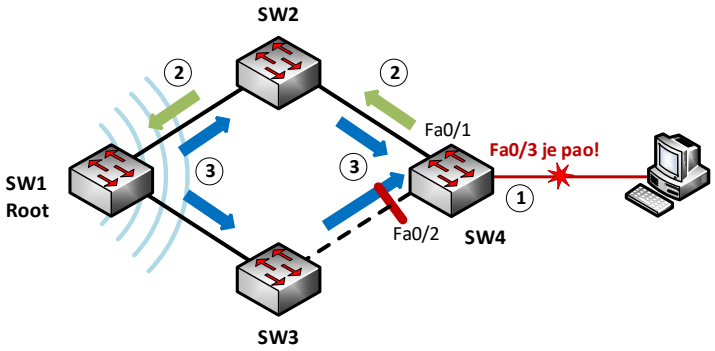
- Korak 1: Svič koji je detektovao promenu šalje *Topology Change Notification* (TCN) BPDU poruku na svoj *Root* port, koju i naredni svičevi prosleđuju na svoje *Root* portove sve do *Root* sviča.
- Korak 2: Po prijemu TCN BPDU poruke, *Root* svič na sve portove šalje konfiguracionu BPDU poruku koja ima postavljenu oznaku da je došlo do promene topologije, tzv. *TCN flag*. Ova poruka propagira do svih svičeva, koji će da detektuju TCN oznaku, preračunavaju eventualno promenjene vrednost *Path cost* atributa, i shodno tome određuju stanja svojih portova.

Proces detekcije promena u mreži i sprovođenja konvergencije ćemo prikazati kroz sledeće karakteristične slučajeve.

Prekid na vezama sa krajnjim uređajima

Iako smo do sada razmatrali samo veze između svičeva, STP se sprovodi na svim portovima, pa tako i na linkovima koji povezuju krajnje uređaje, tzv. pristupnim linkovima (eng. *access link*). Štaviše, tipičan je slučaj da većinu portova sviča čine upravo pristupni linkovi.

Na primeru koji prikazuje Slika 4.15, svič SW4 detektuje prekid pristupnog linka (korak 1) i preko sviča SW2 šalje TCN BPDU poruku do *Root* sviča SW1 (korak 2), nakon čega *Root* svič šalje konfiguracione BPDU poruke sa postavljenom TCN oznakom (korak 3). Svi svičevi su informisani o promeni topologije, ali vrednosti atributa cena putanje koji oni primaju ostaju nepromenjene, pa neće doći do promeni stanja ni jednog porta. Drugim rečima, prekid pristupnog linka ne izaziva STP konvergenciju.

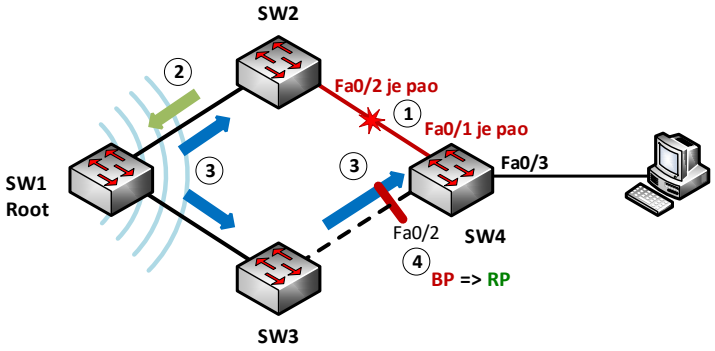


Slika 4.15. STP pri prekidu veze sa krajnjim uređajem

Prekid na direktnim vezama između svičeva

Budući da STP primarno tretira veze između svičeva, prekid ovih veza će izazvati konvergenciju i promenu STP stanja, a posebno u slučaju prekida veza koja čini trenutno STP stablo. Na primeru koji ilustruje slika Slika 4.16, prekid veze između svičeva SW2 i SW4 će detektovani oba sviča. Svič SW4 neće moći da pošalje TCN BPDU poruku, jer je prekinuta veza na njegovom *Root* portu, ali će to učiniti svič SW2 (korak 2). *Root* svič prima TCN BPDU i šalje konfiguracionu BPDU poruku sa postavljenom TCN oznakom. Ova poruka dolazi do svih svičeva, uključujući i svič SW4 koji je na ostatak mreže sada povezan samo preko blokiranog porta.

Ovo je ujedno primer zašto svičevi moraju da prihvataju BPDU poruke na blokiranim portovima, jer će im to omogućiti da zaključe da li treba da aktiviraju blokirani port. To je upravo situacija sa svičem SW4, jer je vrednost atributa cena putanje koju na ovaj način dobija sada jedina, pa stoga i najbolja. Svič ulazi u konvergenciju i postavlja port najpre u *Listening* stanje od 15 sekundi, a zatim i u *Learning* stanje gde se provodi dodatnih 15 sekundi. Tek nakon toga port prelazi u *Forwarding* stanje i počinje da prima i prosleđuje sve pakete. STP je omogućio da svič aktivira preostalu vezu koja je do tada činila petlju, ali je za to bilo potrebno čak 30 sekundi. Tokom tog perioda su svič i svi na njega povezani uređaji bili funkcionalno nepovezani sa ostatkom mreže.

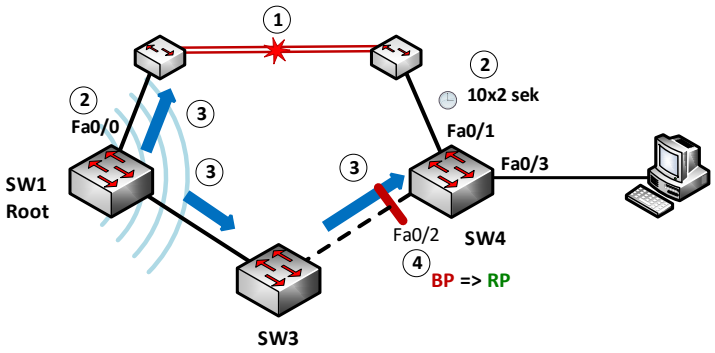


Slika 4.16. Konvergencija kod prekida direktne veze između svičeva

Prekid na indirektnim vezama između svičeva

Svičevi mogu da budu povezani i preko drugih uređaja koji rade na fizičkom nivou. Malo je verovatno da su to habovi, ali čest je primer da to budu tzv. medija konvertori, koji električne signale konvertuju u optičke i obrnuto.

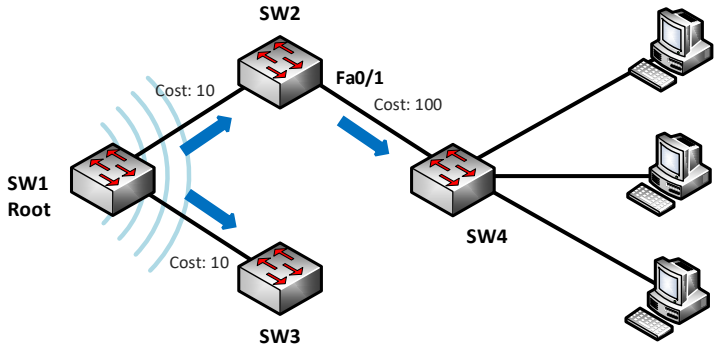
Iako svičevi ne mogu da detektuju prekide na ovakvim vezama, jer su njihove direktne veze sa medija konvertorima i dalje aktivne, svič dalje od *Root* sviča će prestati da dobija BPDU poruke (Slika 4.17, svič SW4). Izostanak 10 ovakvih poruka, u ukupnom trajanju od 20 sekundi, biće signal sviču SW4 da je veza postala neaktivna i da treba da pokrene konvergenciju. I u ovom slučaju konfiguracione BPDU poruke koje se od *Root* sviča emituju na 2 sekunde, će da se učitavaju na blokirani port sviča SW4, koji se aktivira, prelaskom u *Listening* a zatim i u *Learning* stanje, gde se u svakom provodi po 15 sekundi. Ukupno vreme za koje je svič SW4 bio odsečen od ostatka mreže u ovom slučaju iznosi čak 50 sekundi.



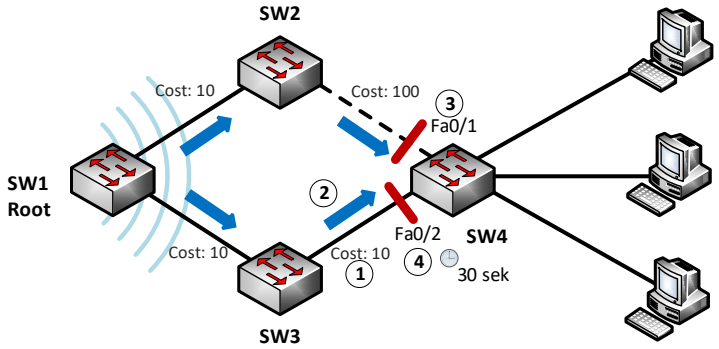
Slika 4.17. Konvergencija kod prekida direktne veze između svičeva

Dodavanje novih veza

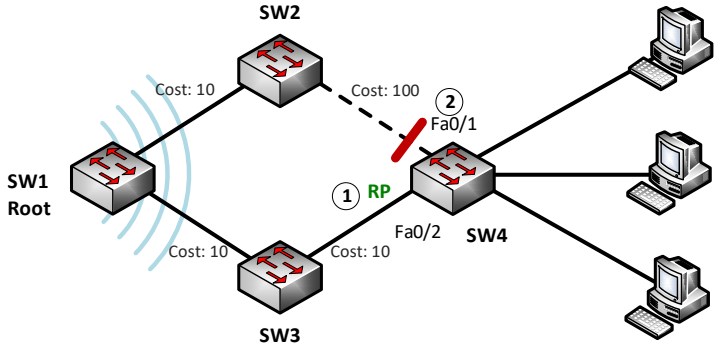
Do promene topologije dolazi i kada se doda nova veza. Naredni primer ilustruje dodavanje veze između svičeva SW3 i SW4. Neposredno pre toga (Slika 4.18) svič SW4 prima vrednost cene putanje 110 od sviča SW2. Nakon dodavanja nove veze, svič SW4 od sviča SW3 počinje da prima nižu vrednost atributa cena putanje (vrednost 20). Istog trenutka port prema sviču SW2 se postavlja u blokirano stanje, ali se port prema sviču SW3 još uvek ne može aktivirati, jer mora da provede u *Listening* i *Learning* stanjima po 15 sekundi. Iako se u ovom slučaju ne radi o prekidu, već dodavanju resursa u mrežu (nova veza), svič SW4 će ipak da bude 30 sekundi odsečen od ostatka mreže (Slika 4.19). Konvergencija traje sve dok se port prema sviču SW3 ne postavi u *Forwarding* stanje, u ovom slučaju kao *Root* port (Slika 4.20).



Slika 4.18. Dodavanje novih veze – stacionarno stanje pre dodavanja



Slika 4.19. Dodavanje novih veze – konvergencija nakon dodavanja



Slika 4.20. Dodavanje novih veze – stacionarno stanje nakon dodavanja

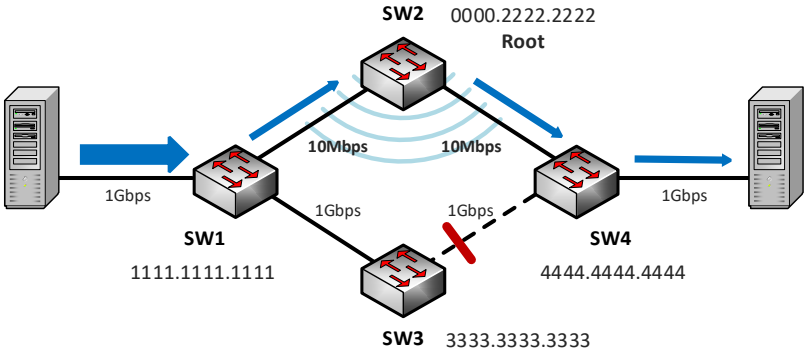
Dodatna podešavanja na svičevima

STP će ispravno da radi u bilo kojoj mreži bez potrebe da se na svičevima bilo šta podešava. Ispravan rad ne znači i idealan, čak ne uvek ni optimalan rad. Najveći nedostatak STP protokola svakako predstavlja preveliko vreme konvergencije, od 30 ili 50 sekundi, tokom kog perioda pojedini delovi mreže mogu da budu potpuno izolovani. Ovaj problem, kao i još neki drugi potencijalni problemi se mogu otkloniti ili ublažiti dodatni podešavanjima na svičevima. Podešavanje na svičevima se radi iz konfiguracionog režima, bilo iz grafičkog režima ili zadavanjem posebnih komandi iz komandnog režima (eng. *Command-line Interface*, skr. CLI).

Kontrola izbora Root sviča

Korišćenjem mehanizma oglašavanja atributa cena putanje do svih svičeva, STP protokol nastoji da na optimalan način uspostavi topologiju mreže bez petlji. Ipak, ova optimalnost je relativna, jer se uspostavlja u odnosu na *Root* svič. Ako su prioriteti svih svičeva isti, tada će se za *Root* svič izabrati svič sa najmanjom MAC adresom, što se može smatrati da je slučajan izbor, koji ne zavisi od mrežnih administratora.

Na primeru koji prikazuje Slika 4.21 najmanju MAC adresu ima svič SW2, koji postaje *Root* svič i u odnosu na njega se uspostavlja STP stablo povezanosti. Sve veze *Root* sviča pripadaju stablu, a problem je što ovaj svič ima veze na najmanjim brzinama od svega 10 Mpbs. Iako između servera sa slike postoji fizička putanja od 1 Gbps, uspostaviće se putanje koja vodi preko *Root* sviča korišćenjem veza od 10 Mbps, pa će protok da bude drastično smanjen.



Slika 4.21. Neoptimalna putanja usled izbora Root sviča

Navedeni primer ilustruje koliko je važno da izbor *Root* sviča bude pod kontrolom mrežnih administratora. Iz ovog razloga je ostavljena mogućnost postavljanja atributa prioriteta sviča (eng. *Priority*), koga čine prva dva bajta identifikacije sviča. Manja vrednost prioriteta je bolja, pa se postavljanjem ovog atributa na najnižu vrednost na određenom sviču forsira da on postane *Root* svič. U prethodnom primeru dovoljno bi bilo i da se na sviču SW2 postavi najveća vrednost prioriteta, čime on postaji najlošiji kandidat za izbor *Root* sviča.

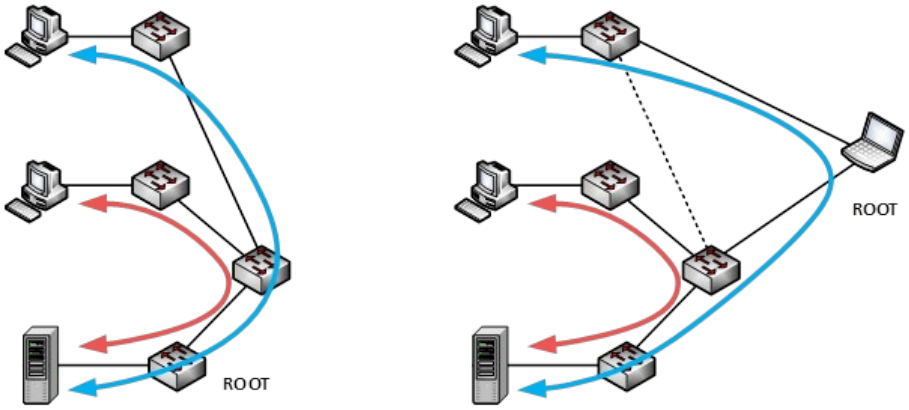
Zabrana drugim uređajima da postanu Root svič

Postavka prioriteta na najnižu (najbolju) vrednost na nekom sviču neće garantovati da će on postati *Root* svič, jer i neki drugi svič takođe može da ima tu istu vrednost. Još drastičniji slučaj je kada neko drugi postavi svič sa ovom vrednosti, koji ako ima manju MAC adresu postaje *Root* svič. Ovo čak ne mora ni da bude svič, već običan računar sa dva porta koji oponaša svič – prima i prosleđuje pakete i sprovodi STP protokol. Slika 4.22 prikazuje situaciju kada napadač svoj laptop poveže na dva sviča u mreži, simulira STP protokol i tom prilikom šalje BPDU poruke koje oglašavaju najmanju vrednost identifikatora sviča. Na taj način napadačev laptop postaje *Root* svič, na osnovu koga se uspostavlja novo stablo povezanosti. Korisnici u mreži ne primećuju ovu promenu, a napadač je u mogućnosti da preuzme saobraćaj koji prolazi između grana mreže povezanih na laptop.

Za zaštitu od ovakvih situacija pojedini svičevi imaju mogućnost da se na određenim portovima zabrani prijem BPDU poruka. Ova opcija se naziva *BPDU Guard*. To se tipično postavlja na pristupnim linkovima na kojima ne očekujemo da se povezuju svičevi. Ako bi se na ovim

portovima pojavila dolazna BPDU poruka, to bi bio signal za neregularno i potencijalno opasno stanje, pa bi svič ceo port automatski postavio u neaktivno stanje (eng. *disabling*).

Opcija *Root Guard* ima mogućnost da detektuje BPDU poruke koje oglašavaju *Root* svič sa manjom identifikacijom i samo u tom slučaju isključi port. Ova opcija je manje restriktivna, jer će dozvoliti povezivanje drugih svičeva dokle god oni ne pretenduju da postanu *Root* svič.



Slika 4.22. Opasnost od krađe saobraćaja:
a) regularna situacija, b) računar napadača postaje Root svič

PortFast opcija

Opcije *BPDU Guard* i *Root Guard* se postavljaju na pristupnim linkovima iz sigurnosnih razloga. Ranije smo videli da se i na pristupnim linkovima takođe sprovodi STP. To će za posledicu imati da po uključivanju računara, port sviča na koji je on povezan najpre uđe u *Listening* stanje, zatim i u *Learning* stanja, i tek nakon 30 sekundi postane aktivan. Drugim rečima, mreža će kasniti za podizanjem računara.

Ovaj proces se može ubrzati ako se port sviča konfiguriše *PortFast* opcija, koja nalaže da se preskoči *Listening* i *Learning* stanje. Ovo ima smisla podesiti samo na pristupnim linkovima, odnosno na portovima na kojima smo sigurni da se neće povezivati svičevi. Time se na portu ipak ne ukida STP protokol, a čak i da se na njih poveže koji drugi svič, eventualne petlje se mogu detektovati. Tom prilikom treba biti posebno oprezan, jer izostanak *Listening* i *Learning* stanja može dovesti do kratkotrajnih petlji, koje se mogu brzo zagušiti i onemogućiti prolazak BPDU poruka koje treba da dovedu do prekida petlji.

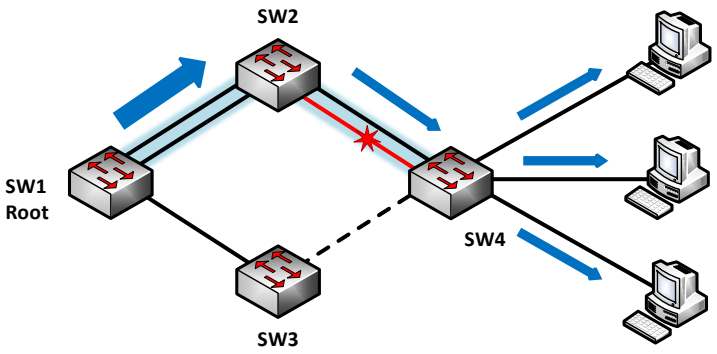
EtherChannel opcija

Pojedini svičevi omogućavaju da se više paralelnih linkova tretiraju kao jedna logička veza srazmerno većeg kapaciteta. Ova opcija je masovno uvedena od strane proizvođača *Cisco Systems* nazivom *EtherChannel*[7], da bi se kasnije standardizovala pod nazivom *Link Aggregation Control Protocol* (LACP) [11].

Primarna namena *EtherChannel* opcije je da se poveća propusni opseg između svičeva na kičmi mreže, ali se time indirektno povećava i stabilnost STP protokola. U slučaju prekida jednog ili više fizičkih linkova u *EtherChannel* vezi, dokle god postoji bar jedan operativni link BPDU poruke će da prolaze i neće doći do promene topologije i konvergencije. *EtherChannel*

[7] *EtherChannel* protokol je inicijalno osmišljen od strane kompanije *Kalpana*, koju je 1994. godine kupila kompanija *Cisco Systems* i nastavila da koristi *EtherChannel* kao tehnologiju u svojim komunikacionim uređajima.

podržavaju i računari, što je posebno korisno za servere koji treba da podnesu veliko opterećenje.



Slika 4.23. Stabilnost STP postavljanjem EtherChannel opcije

4.4 Rapid Spanning Tree Protocol – RSTP

Iako je STP protokol nastavo davne 1985. godine još u doba bridževa, bez izmena je mogao da se primeni i na svičeve, a zbog svoje efikasnosti održao se do današnjih dana. Ipak, najveći nedostatak STP protokola predstavlja spora konvergencija. Za mnoge današnje aplikacije, a još više za korisnike, 30 sekundi prekida u mreži i čekanje da se uspostavi stacionarno stanje je daleko od prihvatljivog. STP je zahtevao unapređenje, što je postignuto kroz noviju verziju pod nazivom *Rapid Spanning Tree Protocol* (skr. RSTP) [12]. RSTP je zadržao osnovne principe vezane za uspostavljanje stabla bez petlji, kao što je izbor *Root* sviča i stanja portova, ali je uveo nekoliko novina.

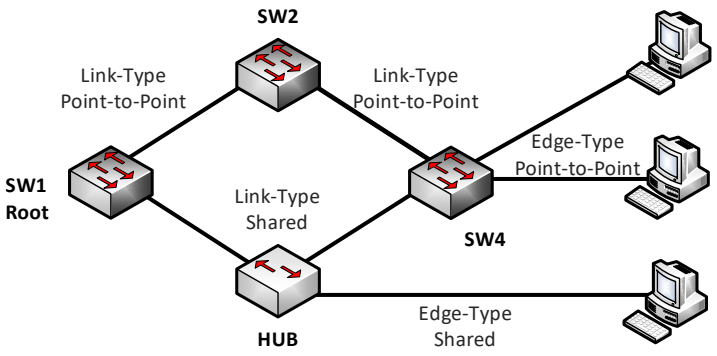
Spomenimo najpre terminološke novine, gde RSTP pravi razliku između pojedinih vrsta linkova, i to:

- *Link type* – veza između svičeva
- *Edge type* – veza između sviča i krajnjeg uređaja (pristupni link)

RSTP je i dalje kompatibilan sa habovima, ali pravi razliku između veza ostvarenih preko haba u odnosu na veze preko sviča, i to:

- *Point-to-Point* – direktne veze između svičeva
- *Shared* – veza preko deljenog segmenta posredstvom haba

Kombinacije ovih tipova veza ilustruje Slika 4.24.

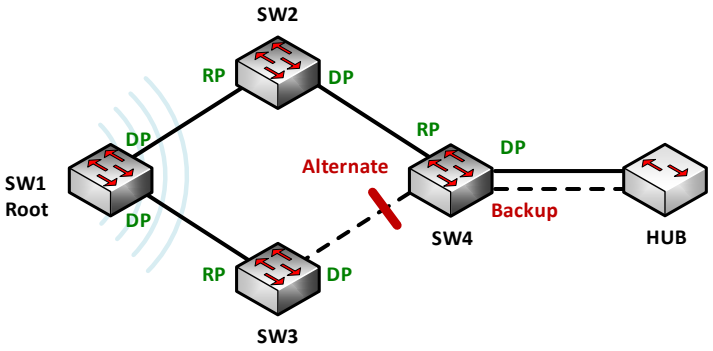


Slika 4.24. Vrste linkova kod RSTP protokola

RSTP dalje uvodi dva nova dodatna stanja portova

- *Alternate port* – port na sviču koji je prvi sledeći kandidat za izbor *Root* porta.
- *Backup port* – port na sviču koji je prvi sledeći kandidat za izbor *Designated* porta.

Budući da se *Designated* port bira na nivou segmenta, *Backup* port se javlja u veoma retkom, ali dozvoljenom slučaju, kada se hab poveže na svič sa dva linka (Slika 4.25).



Slika 4.25. Alternate i Backup portovi

Iako kompatibilan sa habovima, RSTP nije kompatibilan sa STP protokolom. Primera radi, cene portova su promenjene i sada se računaju deljenjem brzine veze od vrednosti 20 Tbps, pa se dobijaju vrednosti kao u narednoj tabeli.

Brzina	STP inicijalne cene	STP revidirane cene	RSTP cena
10 Mbps	100	100	2.000.000
100 Mbps	10	19	200.000
1 Gbps	1	4	20.000
10 Gbps	1	2	2.000

Tabela 2. Cene portova u zavisnosti od brzine protoka

Najveća novina koju donosi RSTP odnosi se na značajno ubrzanje konvergencije koja nastaje pri promeni topologije u mreži. Umesto da se pasivno čeka 2 puta po 15 sekundi kako bi se dalo dovoljno vremena da se stabilizuje situacija čak i u veoma kompleksnim mrežama, RSTP pristupa aktivnoj komunikaciji u cilju eliminisanja petlji u tranzicionim stanjima pri konverziji. Uvode se dve nove vrste BPDU poruka pomoću kojih dva povezana sviča međusobno komuniciraju i dogovaraju se oko stanja portova na zajedničkom linku:

- *Proposal* – iniciranje dogovora o stanjima portova.
- *Agreement* – odgovor sa definisanim stanjem portova.

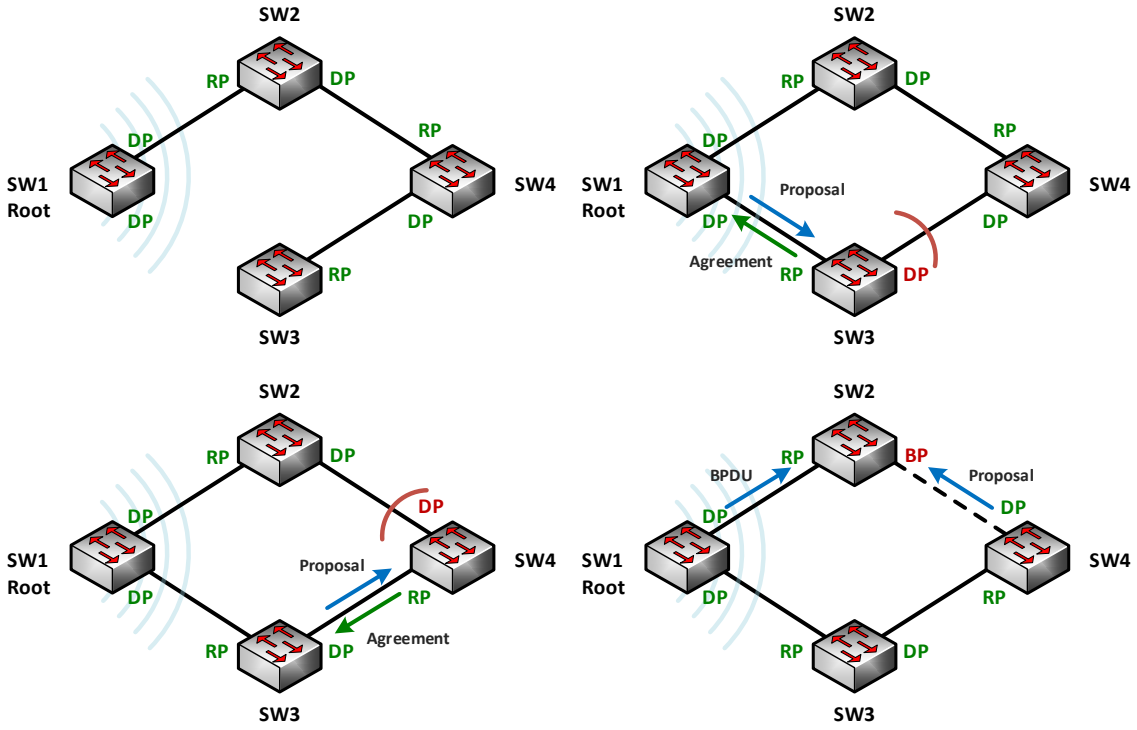
Proces konvergencije se sprovodi na sledeći način:

- Svičevi koji detektuju promenu u mreži šalju *Proposal* poruke susednim svičevima

- Svičevi koji dobiju *Proposal* poruke rade sledeće:
  - Svoje preostale portove stavljaju u privremeno blokirano stanje kako bi sprečile eventualne petlje.
  - Šalju *Agreement* poruku kao odgovor na primljenu *Proposal* poruku, kojom se definitivno utvrđuje stanje porta na tom linku.
  - Na ostale portove, koji su trenutno u blokiranom stanju, šalju novu *Proposal* poruku, čime pokreću novi proces dogovora sa narednim svičevima.

Razmena *Proposal* i *Agreement* poruka, odnosno uspostavljanje stanja na jednom linku, je gotov trenutno. Nakon toga ovaj proces propagira i na ostale linkove i veoma brzo se prenosi po celoj mreži.

Na primeru koji prikazuje Slika 4.26, nakon dodavanja linka između sviča SW1, koji je *Root* svič, i sviča SW3, svič SW3 detektuje bolju putanju do *Root* sviča i ulazi u sinhronizaciju. Najpre se privremeno blokiraju svi ostali portovi da bi se sprečile petlje (veza između SW3 i SW4), pa se šalje *Agreement* poruka ka sviču SW1, nakon čega se aktiviraju portovi na ovoj vezi (*Forwarding* stanje). Proces se prenosi dalje između sviceva SW3 i SW4, gde se takođe detektuje bolja putanja do *Root* sviča pa portovi menjaju stanje. I konačno, u *Proposal* poruci koju svič SW2 dobije od sviča SW4 nudi se lošija cena putanja u odnosu na putanju koju svič SW2 dobija neposredno od sviča SW1, pa se ova grana izuzima iz stabla povezanosti. Time se proces konvergencija završava.

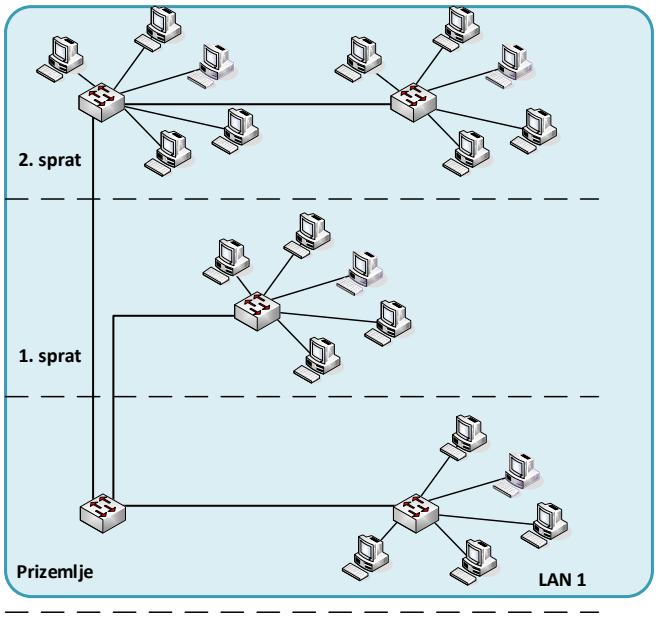


Slika 4.26. RSTP konvergencije. a) inicijalno stanje, b) komunikacija između sviceva SW1 i SW3 nakon njihovog povezanja, c) komunikacija između sviceva SW3 i SW4, d) komunikacija između sviceva SW4 i SW2, kraj konvergencije

Kao i kod STP protokola, prekid na indirektnim vezama između dva sviča se detektuje odsustvom BPDU poruka. Ovaj broj poruka je takođe smanjen sa 10 na 3 poruke, što uzrokuje vreme detekcije od svega 6 sekundi.

4.5 Virtualne lokalne računarske mreže - VLAN

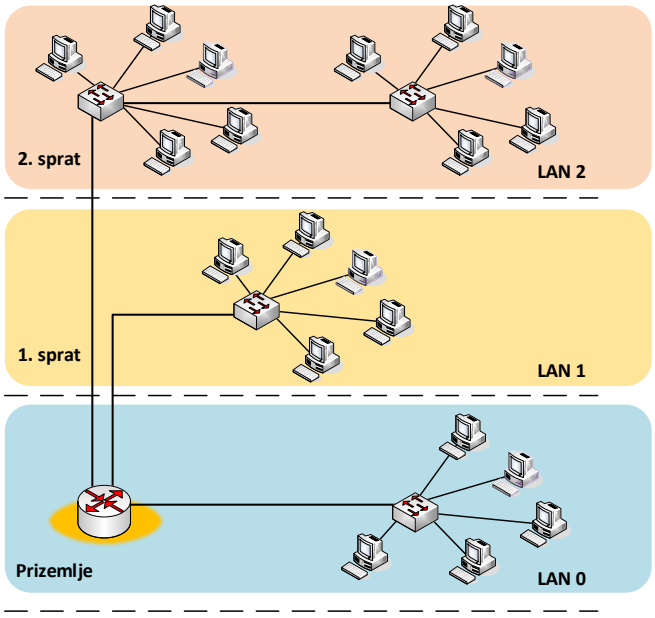
Svičevi su prevazišli problem kolizije i povećali propusni opseg mreže, ali su takođe i omogućili i da fizička mreža nema tehnička ograničena po pitanju veličine. Strukturno kabliranje u velikim poslovnim zgradama omogućava povezivanje svih uređaja u jedinstvenu Eternet mrežu (Slika 4.27).



Slika 4.27. Jedinstvena Eternet mreža na nivou poslovnog objekta

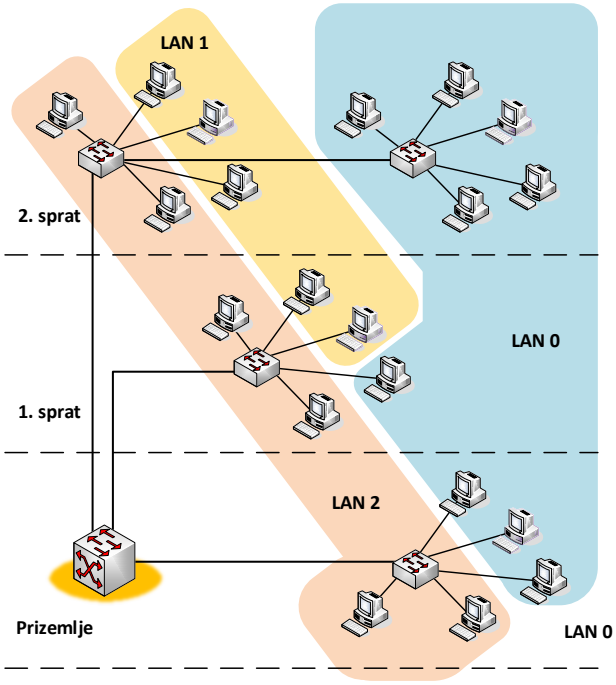
Ipak, u funkcionalnom, a još više u bezbednosnom smislu, postoje ograničenja koja limitiraju veličinu mreže. Svičevi imaju ograničene bridžing tabele tipično do nekoliko hiljada redova, a veći broj uređaja u mreži bi doveo do njihovog neefikasnog rada. Takođe, veliki broj uređaja u mreži bi izazivao česte flading procese i brodkast saobraćaj, što bi dodatno opterećivalo mrežu. Još veći problem je što paketi tom prilikom završavaju do svih učesnika u mreži, a podaci koji se prenose postaju potencijalno dostupni svima.

Iz navedenih razloga teži se segmentaciji mreže na manje celine. U fizičkom smislu to se najjednostavnije postiže odvajanjem blisko povezanih sviceva, npr. po spratovima objekta (Slika 4.28). Svaka ovakva celina predstavlja posebnu LAN mrežu na L2 nivou. Na ovom mestu istaknimo da ove LAN mreže nisu izolovane, već da se njihovo povezivanje sprovodi na L3 nivou preko uređaja koji se nazivaju ruteri. Koncepti komunikacije na L3 nivou i rad rutera je dovoljno velika tema koja će biti pokrivena kroz naredna poglavlja, a na ovom mestu prihvatimo samo da ruteri omogućavaju komunikaciju više LAN mreža.



Slika 4.28. Fizički segmentirana Eternet mreže na nivou poslovnog objekta

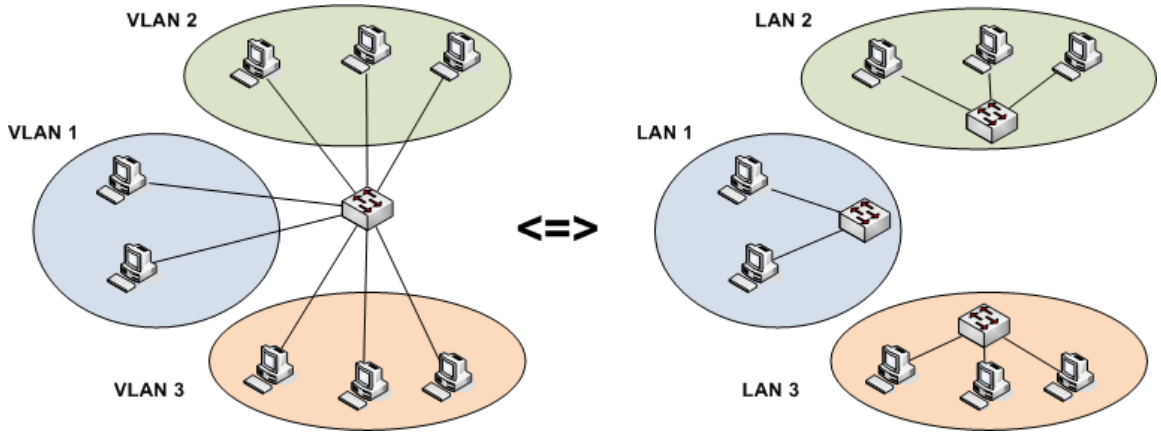
Mreža segmentirana po spratovima ili fizičkim delovima objekta je dosta nefleksibilna za poslovne potrebe, koje nameću podelu prema vrsti korisnika i nameni korišćenja mreže. U poslovnim mrežama to su obično različite službe, poput prodaje, razvoja, marketinga i sl. a koje mogu biti razdvojene i raspoređene u bilo kom delu objekta. Potreba za fleksibilnom podelom mreže je još uverljivija na primeru fakulteta, gde se segmentacija može sprovesti na laboratorije, učionice, profesorske kabinete, studentsku službu itd. U idealnom slučaju želimo da postignemo logičko grupisanje uređaja nezavisno od njihove fizičke lokacije, odnosno nezavisno od toga na koji svič su povezani, kao što to ilustruje Slika 4.29.



Slika 4.29. Logička segmentirana Eternet mreže na nivou poslovnog objekta

Imajući u vidu logiku rad sviča i kontrolu nad paketima koja se može postići, na nivou jednog sviča moguće je grupisati portove u nezavisne celine koje se pojedinačno funkcionalno ponašaju kao odvojeni svičevi, odnosno odvojene mreže (Slika 4.30). Svaka od ovih celina poseduje odvojene bridžing tabele i sprovodi nezavisno procesiranje paketa. Na ovaj način se dobijaju potpuno funkcionalne ali nezavisne LAN mreže, koje se nazivaju virtualne lokalne računarske mreže (eng. *Virtual Local Area Network*, skr. VLAN[8]).

Važno je istaći da se VLAN u funkcionalnom smislu ponaša potpuno ekvivalentno kao i fizička LAN mreže – *flooding* i brodkast saobraćaj se prenosi samo u okviru jednog VLAN-a, bez mogućnosti da Eternet paket iz jednog VLAN-a završi u drugom.



Slika 4.30. Logička segmentirana Eternet mreže na nivou jednog sviča

VLAN-ove je moguće „razvući“ i na više svičeva, tako što bi se za svaki VLAN izdvojila jedna fizička veza između svičeva. Naravno, ovo rešenje ne bi bilo nimalo skalabilno, jer bi odvojene fizičke veze brzo potrošile potove na svičevima. Rešenje je da se između svičeva i dalje koristi samo jedna fizička veza, a da se VLAN-ovi logički razdvoje. Ova veza koja prenosi više VLAN-ova se naziva trank link (eng. *trunk link*), dok se ostale veze nazivaju pristupni linkovi.

I pristupne i trank linkove svičevi prepoznaju preko povezanog porta, a pakete na isti način prosleđuju korišćenjem bridžing tabela sa naučenim informacijama koja se MAC adresa nalazi na kom portu. Na ovaj način paketi će da nađu put do odredišta nezavisno od toga da li se prenose po pristupnom ili trank linku.

Problem nastaje kada se sprovodi *flooding* ili prenosi brodkast paket koji mora da se prenese do svih uređaja samo unutar originalnog VLAN-a. Kada ovakav paket sa pristupnog linka iz jednog VLAN-a pređe na trank link i dođe do drugog sviča treba da se prosledi samo tom originalnom VLAN-u. Jedini način da svič zna koji je to VLAN je da ta informacija bude sadržana u paketu. Za to nam je najpre potrebna identifikacija VLAN-ova, a zatim i da se identifikator VLAN-a prenosi u Eternet paket.

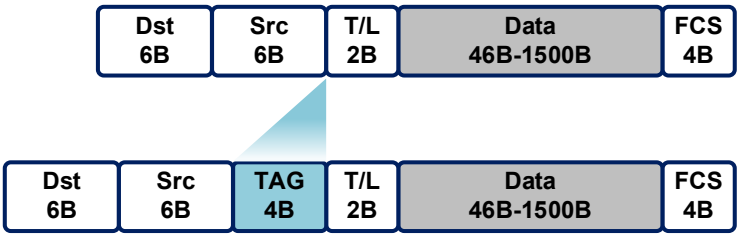
Za identifikaciju VLAN-a uvodi se celobrojna vrednost od 12 bita, što ukupno daje do 4096 identifikacija. Ova vrednost se zajedno sa još nekim parametrima postavlja u dodatno polje veličine 4 bajta, neposredno nakon polja izvorišne MAC adrese Eternet paketa.

[8] U inženjerskoj praksi na engleskom, ali i našem jeziku, VLAN se izgovara „vi-lan“

Ova tehnika označavanja, odnosno „tagovanja“ VLAN-ova naziva se *VLAN Frame Tagging*, a definisana je standardom 802.1Q. Ona se sprovodi isključivo na trank linkovima, i to na sledeći način:

- Kada paket dođe sa akces linka i treba da se prosleđuje na trank link, svič prepoznaje kom VLAN-u pripada port na koji ne pristigao paket, manja paket dodavanjem 4 bajta sa identifikacijom tog VLAN-na i paket prosleđuje na trank link.
- Kada paket dođe sa trank linka i koji treba da se prosledi na drugi trank link, on je već tagovan odgovarajućim VLAN-om, pa se prosleđuje neizmenjen.
- Kada paket koji dođe sa trank linka i treba da se prosledi na pristupni link, svič očitava dodato polje sa identifikacijom VLAN-a i paket prosleđuje na odgovarajući port koji pripada originalnom VLAN-u. Tom prilikom svič mora da ukloni ranije dodata 4 bajta i time rekonstruiše originalne Eternet pakete.

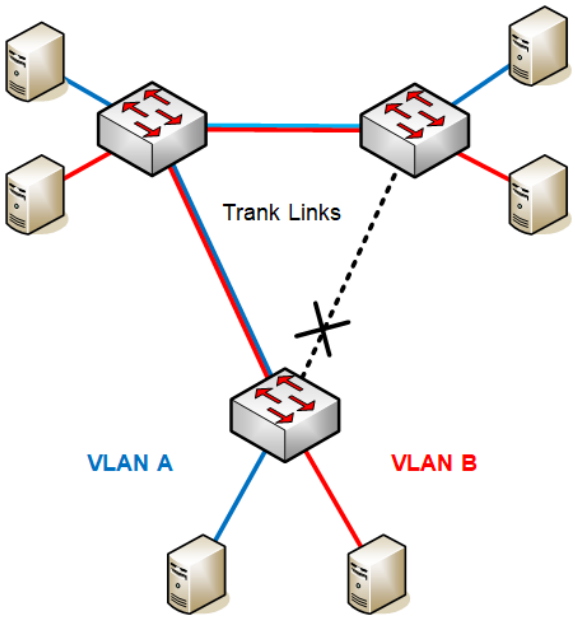
Navedeni postupak označavanja paketa na trank linkovima se sprovodi za sve pakete – i one koji se prenose samo do naznačenog odredišta, kao i pakete koji se prenose kroz *flooding* proces na više portova ili brodkast komunikacijom do svih odredišta.



Slika 4.31. VLAN Frame Tagging – 4 bajta dodata u originalni Eternet paket

VLAN-ovi omogućavaju da se fizička LAN mreža logički organizuje na proizvoljan način. Tipična implementacija je da se veze između svih svičeva, koje čine tzv. kičmu mreže (eng. *backbone*), postave u trank režim rada, a da se uređaji povezuju na bilo koji svič dodavanjem u odgovarajući VLAN. Korisno je da kičma mreže sadrži redundantne veze koje je čine otpornom na otkaze uz primenu STP ili RSTP protokola.

VLAN tehnika kao noviji koncept morao je da se prilagodi principima STP protokola, tako što se STP stablo uspostavlja na nivou trank linkova i primenjuje na sve VLAN-ova (Slika 4.32). Ovakav režim „deljenja“ STP i RSTP protokola od strane svih VLAN-ova, naziva se *Common Spanning Tree* (skr. CST).



Slika 4.32. Common Spanning Tree - *Zajedničko STP stablo za sve VLAN-ove*

Ovakav klasičan pristup gde se veze između svičeva ili koriste ili uopšte ne koriste uobičajen je za STP i RSTP protokole, ali donekle ograničava fleksibilnost i nezavisnost VLAN-ova. Potpuna međusobna nezavisnost VLAN-ova podrazumeva da se STP i RSTP protokoli sprovode unutar svakog pojedinačnog VLAN-a. Time se uvodi dodatni posao za svičeve, ali se omogućava i optimalnije korišćenje fizičkih linkova. U prethodnom jednostavnom primeru moguće je podesiti prioritete svičeva za različite VLAN-ove tako da se u jednom VLAN-u blokira jedan link, dok se u drugom VLAN-u blokira drugi link. Rezultat je da se u stabilnom stanju koriste oba linka i to punog kapaciteta ekskluzivno za jedan ili drugi VLAN. Ovaj pristup je usvojen kao zvaničan standard pod oznakom IEEE 802.1s i nazivom *Multiple Spanning Tree Protocol* skr. MSTP) [13].

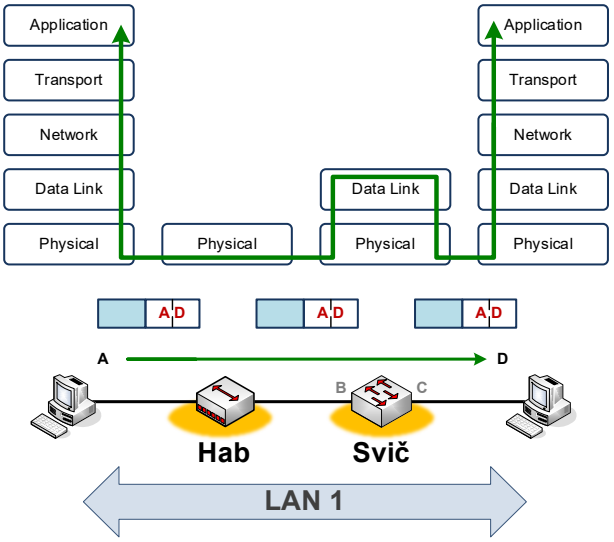
Drugi deo:

Mrežni sloj

5. Internet protokol

Prethodna poglavlja su razmatrala komunikaciju na drugom sloju, odnosno sloju veze podataka (*Data-Link Layer*), koji se neposredno oslanja na prvi, fizički sloj. Eternet protokol, kao dominantna tehnologija drugog sloja, predstavljen je u kontekstu lokalnih računarskih mreža, što je i bila prvobitna namena Eterneta – povezivanje uređaja na bliskim rastojanjima i velikim brzinama. Pojavom svičeva i optičkih vlakana, Eternet je postao podjednako efikasan i na mnogo većim rastojanjima, od više desetina i stotina kilometara. Šta više, Eternet se danas koristi kao dominantni protokol za prenos podataka na pojedinačnim vezama koje čine Internet mrežu. U prethodnoj rečenici akcenat je stavljen na „pojedinačne veze“, koje na drugom nivou koriste Eternet protokol i bilo bi potpuno pogrešno reći da je Internet mreža Eternet tipa. Za razliku od nezavisnih Eternet linkova koji povezuju dve tačke na proizvoljnom rastojanju, Eternet mreže sa više učesnika su i dalje LAN mreže, ne samo po pitanju fizičke rasprostranjenosti, već još više zbog ograničene skalabilnosti po pitanju broja uređaja, veza i učesnika.

Podsetimo se najpre da se Eternet paket formira na strani izvorišnog uređaja, navodeći i izvorišnu i odredišnu MAC adresu. Za razliku od habova koji rade na prvom sloju posmatrajući paket kao niz nula i jedinica, svičevi na drugom sloju gledaju sadržaj Eternet paketa, na osnovu čega odlučuju o njegovom daljem prosleđivanju. Tom prilikom se Eternet paket nepromenjen prosleđuje i kroz habove i kroz svičeve (Slika 5.1) sve do krajnjeg odredišta.



Slika 5.1. Komunikacija između dva uređaja u istoj LAN mreži

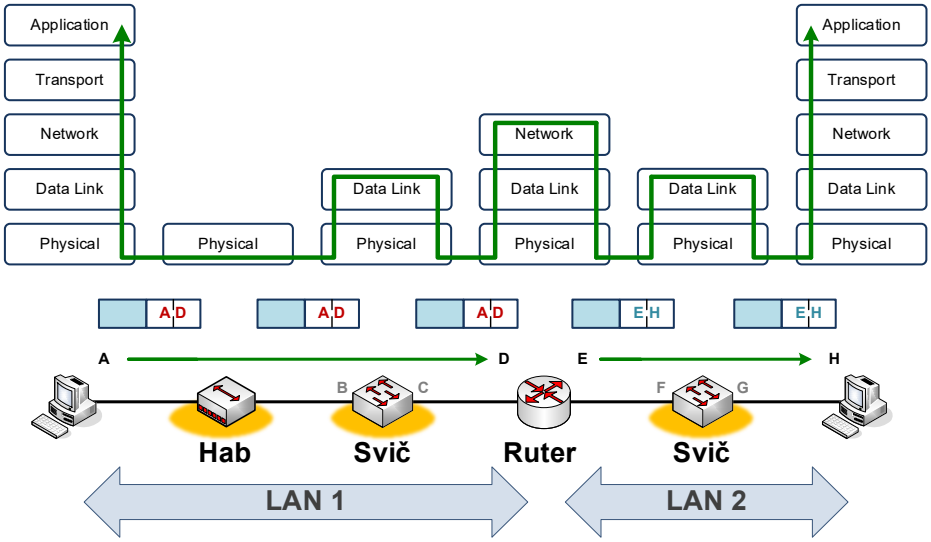
Prethodno je istaknuta potreba za segmentacijom LAN mreže na nezavisne VLAN-ove, a koji se međusobno povezuju na trećem tzv. mrežnom nivou, korišćenjem rutera. Upravo mrežni nivou i ruteri omogućavaju skalabilnost globalnih razmera, što predstavlja osnovnu tehnologiju današnjih računarskih komunikacija, kao i celokupne Internet mreže. Protokol razmene

podataka na ovom nivou se naziva Internet protokol, opšte poznat po skraćenici *IP* (*Internet Protocol)*[9] [14].

5.1 Princip komunikacije preko IP protokola

Posredstvom IP protokola komunikacija na mrežnom nivou se odvija između krajnjih uređaja, koji mogu da pripadaju različitim i veoma udaljenim LAN mrežama. I na mrežnom nivou uređaji se jedinstveno identifikuju posredstvom adresa, koje se u ovom slučaju zovu IP adrese. Slično kao kod Eternet paketa, IP adrese izvorišta i odredišta se navode u zaglavlju IP paketa, a koji se u celini enkapsulira i prenosi unutar protokola drugog nivoa, u ovom slučaju Eternet paketa. IP protokol ima registrovani identifikacioni broj koji se navodi u polju *Type* Eternet zaglavlja, a koji iznosi 800_{hex} [15].

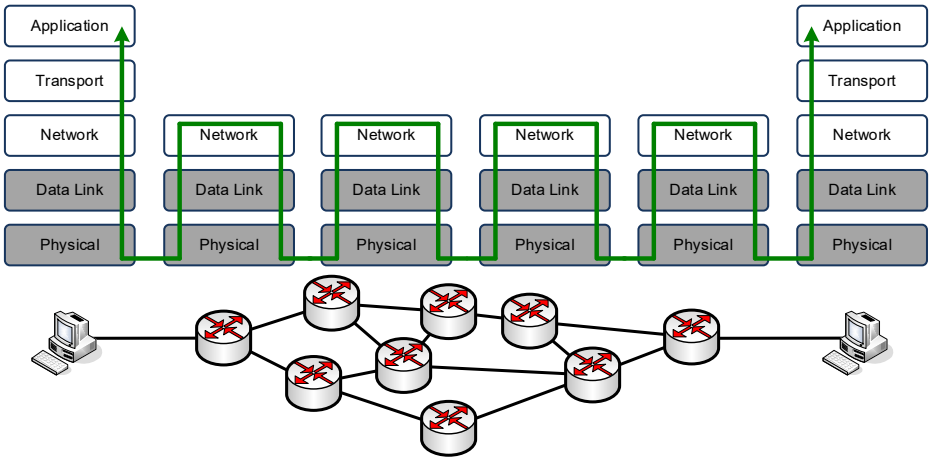
Na primeru koji prikazuje Slika 5.2, komunikacija se odvija između računara A i H u različitim LAN mrežama, koje su povezane preko rutera. U mreži LAN 1, IP paket se najpre prenosi unutar jednog Eternet paket, koji se šalje od izvorišnog uređaja A samo do rutera, navodeći za odredište MAC adresu porta rutera (na slici označenog sa D). Ruter, kao odredište na L2 nivou, prima ovaj Eternet paket, iz njega uzima IP paket, gleda odredišnu adresu u njegovom zaglavlju i zaključuje da treba da ga pošalje u LAN 2 prema odredišnom uređaju H. Nakon toga ruter formira novi Eternet paket, koji za izvorišnu MAC adresu ima port rutera u mreži LAN 2 (na slici označen sa E), a za odredište MAC adresu krajnjeg uređaja (na slici označeno sa H).



Slika 5.2. Komunikacija između dve uređaja u različitim LAN mrežama

Slično prethodnom primeru, komunikacija između dva proizvoljno udaljena uređaja na mrežnom sloju se odvija preko IP paketa, koji se gotovo nepromenjeni prosleđuju od rutera do rutera kroz niz segmenata na drugom nivou. U tipičnom slučaju ovi segmenti su direktni L2 linkovi između rutera, a samo na strani krajnjih uređaja oni predstavljaju LAN mreže. Takođe, mreža rutera može da bude proizvoljno velika i raznovrsna, uključujući redundantne veze koje omogućavaju višestruke putanje (Slika 5.3).

[9] U ovoj knjizi oznaka IP označava originalnu i još uvek masovno korišćenu verziju IP protokola – verziju 4 (*IPv4*). Danas je takođe u upotrebi i novija verzija 6, u oznaci *IPv6*, za koju je ispravnije reći da predstavlja novi protokol, koji nije kompatibilan sa verzijom 4.



Slika 5.3. Komunikacija između dva proizvoljno udaljena uređaja

5.2 Karakteristike IP protokola

Slično Eternet tehnologiji, i IP protokol je postao dominantan ne zbog svoje superiornosti u funkcionalnom smislu, već pre svega zbog svoje otvorenosti i jednostavnosti. Otvorenost je omogućila da bude podržan od strane mnogih proizvođača komunikacione opreme, a jednostavnost je omogućila da bude dostupan i široko prihvaćen u praksi. Brojni „nedostaci“, odnosno nepostojeće napredne funkcionalnosti, su rešavane na drugi način – funkcijama na višim slojevima, pratećim protokolima i tehnologijama koje su primenjivane na uređajima.

Osnovne karakteristike IP protokola su sledeće:

- Prenos bez prethodno uspostavljene veze (*Connectionless*)

  Svaki IP paket prenosi se nezavisno, bez potrebe da se prethodno uspostavi posebna veza i dogovor strana koje komuniciraju. To za posledicu ima sledeće:

  - Pošiljalac može da šalje pakete u bilo kom trenutku.
  - Pošiljalac ne zna da li je primalac trenutno dostupan (povezan na mrežu), kao ni to da li uopšte postoji.
  - Primalac može paket da dobije od bilo kog pošiljaoca u bilo kom trenutku.

- Nazavisnost od vrste medijuma i tehnologije prenosa (*Media Independent*)

  IP prenos ne zavisi od vrste medijuma na fizičkom nivou, kao ni od primenjene tehnologije na drugom nivou na bilo kom segmentu između rutera na putu do odredišta.

- Nepouzdan prenos (*Unreliable*)

  Ne garantuje se da će svi paketi biti isporučeni, ali će IP pokušati da to učini „u najboljoj nameri“ (*best-effort*). Neki paketi u prenosu mogu da budu oštećeni ili odbačeni na samim ruterima (npr. zbog opterećenja ili zagušenja). IP protokol ne obezbeđuje da pošiljalac zna da li je paket isporučen, kao što ni primalac ne zna da li je paket za njega uopšte poslat.

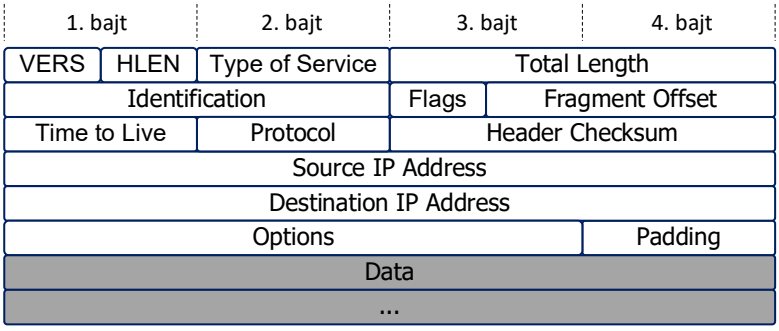
- Proizvoljna topologija povezivanja

  Ruteri mogu da budu povezani na proizvoljan način, uključujući i postojanje redundantnih veza. To za posledicu ima sledeće:

- IP protokol ne razrešava eventualne petlje u putanji (kasnije ćemo videti da to rade ruteri).
- Paketi mogu da se prenose po različitim putanjama u jednom smeru između dva učesnika. To može da bude korisno, jer se time balansira saobraćaj i bolje koriste mrežni resursi (*load-balancing*).
- Različite putanje u istom smeru mogu da izazovu promenu redosleda paketa na prijemnoj strani (*reordering*).
- Paketi mogu da se prenose po različitim putanjama u odlaznom i dolaznom smeru između dva učesnika, tzv. asimetrično rutiranje.

5.3 Format IP paketa

IP je protokol trećeg (mrežnog) nivoa, čiji je osnovni zadatak da prenosi enkapsulirane poruke četvrtog (transportnog) nivoa. Već je istaknuto da zaglavlje IP poruke sadrži IP adrese izvorišnog i odredišnog uređaja, svaka veličine od po 4 bajta. Osim toga zaglavlje sadrži i polja koja ilustruje Slika 5.4.



Slika 5.4. Format IP paketa (IP verzija 4)

- ***Version*** (**VERS**) – Verzija IP protokola.

  Aktuelne verzije su verzija 4 i verzija 6, koje imaju različite formate. Kada pristigne IP paket kao niz bajtova, prva 4 bita ukazuju na verziju, na osnovu čega se prepoznaje format paketa i shodno tome preuzimaju ostala polja.

- ***Header Length*** (***HLEN***) – Dužina zaglavlja

  Polje veličine 4 bita, koji nosi informaciju u dužini celog zaglavlja, izraženo u rečima od po 4 bajta. Ovo je potrebno iz razloga što zaglavlje može biti proizvoljne dužine (zbog opcionog polja), ali zbog veličine ovog polja, najviše 64 bajta.

- ***Type of Service*** (***ToS***) – „Tip servisa“

  Označava „tip servisa“, odnosno „klase saobraćaja“ čiji se podaci prenose u paketu, kako bi se po potrebi omogućilo da pojedini paketi imaju veći prioritet u odnosu na druge. Ruteri mogu da gledaju ovo polje i paketima većeg prioriteta daju prednost pri prosleđivanju. Paketi manjeg prioriteta mogu da budu duže zadržani u ruteru ili čak odbačeni u slučaju velikog opterećenja.

- ***Total Length*** – Ukupna veličina paketa

Polje veličine 2 bajta, koje označava ukupnu veličinu IP paketa u bajtovima, uključujući i zaglavlje (maksimalno do 65.536 bajtova).

- ***Identification***, ***Flags***, ***Fragment Offset*** – Polja za fragmentaciju paketa

  IP paketi se prenose „s-kraja-na-kraj“, nezavisno od tehnologije na nižem nivou. Sa druge strane, paketi mogu da budu veći od maksimalno dozvoljene veličine paketa koju tehnologije na drugom nivou ograničavaju, što je slučaju Eternet protokola iznosi 1500 bajtova. Iz potrebe da se i veći paketi prenose i u ovom slučaju, IP protokol podržava deljenje paketa na manje delove, tzv. fragmentacija (*fragmentation*), što se objašnjava nešto kasnije u nastavku poglavlja.

- ***Time-to-Live*** (**TTL**) – „Životni vek“ paketa

  Namena IP protokola je da omogući komunikaciju u proizvoljno velikim mrežama, što se i obistinilo na globalnom nivou razvojem Internet mreže. Da bi se sprečilo eventualno nekontrolisano beskonačno kruženje paketa koje može nastati usled različitih grešaka, IP protokol uvodi krajnji mehanizam zaštite od beskonačnog kruženja, tako što ograničava „životni vek“ paketa, izražen kroz maksimalni broj koraka, odnosno prolazaka kroz rutere. Ovo polje, veličine 1 bajta, se na izvorištu postavlja na određenu i dovoljno veliku vrednost, a pri prosleđivanju na svaki naredni link, ruter će ovu vrednost da umanji za 1. U slučaju da se dosegne vrednost nula, ruter će da uništi paket.

  I pored veličine današnje Internet mreže, zbog hijerarhijske organizacije povezivanja, broj koraka do odredišta u regularnim situacijama ipak nije mnogo velik, a često je i jednocifren.

- ***Protocol*** – Identifikacija protokola višeg nivoa

  Ovo polje nosi identifikaciju protokola četvrtog nivoa, čiji se podaci prenose u telu poruke, a radi procesa multipleksiranja i demultipleksirnja. Na sličan način se i u zaglavlju Eternet paketa prenosi identifikacija IP protokola, a koja ima različite vrednosti za verziju 4 i verziju 6.

- ***Header Checksum*** – Kontrola grešaka u zaglavlju

  Za proveru eventualne greške na nivou bio kog bita u zaglavlja, koristi se tzv. „prvi komplement“ (invertovane binarne cifre) sume reči od 16 bita na nivou celog zaglavlja.

- ***Source IP address***, ***Destination IP address*** – IP adrese izvorišta i odredišta

  IP adrese identifikuju učesnike u IP komunikaciji, odnosno pošiljaoca i primaoca IP poruke. U verziji 4 IP protokola one su veličine od 4 bajta. IP adrese predstavljaju posebno važan koncept IP protokola i korišćenja savremenih računarskih mreža, pa se za njih izdvaja posebno potpoglavlje, koje sledi u nastavku.

- ***Options***, ***Padding*** – Opciona polja

  Tokom inicijalnog razvoja IP protokola, za potrebe testiranja, ali i za eventualno buduće potrebe, uvedeno je opciono polje (*Options*) varijabilne dužine. Budući da polje HLEN izražava dužinu zaglavlja u rečima od po 4 bajta, u slučaju korišćenja opcionog polja, zaglavlje mora da se proširi bajtovima koji se neće koristiti, za šta služi polje *Padding*. Ova polja se danas ipak retko koriste.

Fragmentacija IP paketa

Protokoli na drugom nivou imaju ograničenu maksimalnu veličinu podataka koju mogu da prihvate od trećeg nivoa, koja se označava sa MTU (*Maximum Transmission Unit*). U slučaju da je IP paket veći od ove vrednosti na nekom delu puta do odredišta, IP protokol omogućava da se paket podeli na više manjih celina i u delovima prenese preko paketa drugog sloja. Ovaj proces se naziva fragmentacija. Fragmentirani paket je potrebno i obnoviti, što se naziva reasembliranje, i sprovodi se na krajnjem odredištu. Svaki fragment IP paketa takođe predstavlja IP paket, koji mora da sadrži i sve potrebne informacije kako bi svi fragmenti mogli da se reasembliraju u originalni paket.

Iz navedenih razloga, svaki IP paket u komunikaciji između dva uređaja ima svoju jedinstvenu internu identifikaciju tokom određenog perioda. Ovaj podatak se prenosi u polju *Identification* koje je veličine 2 bajta.

Polje *Flags* sadrži kontrolne bite, odnosno flegove, i to:

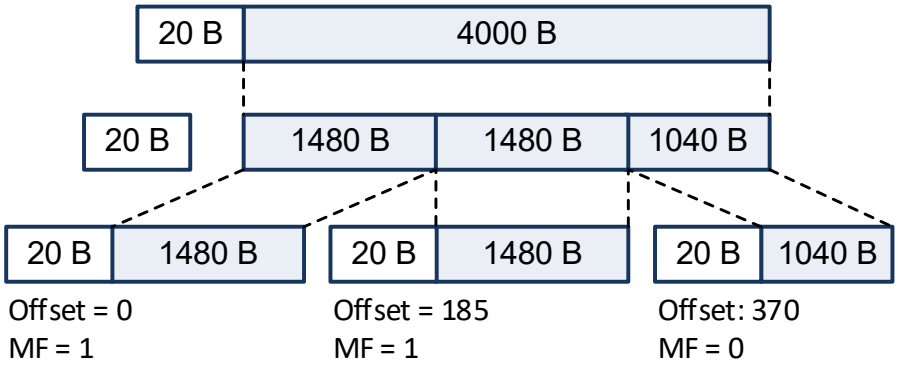
- ***DF Flag*** (*Don't Fragment*) – Setovana vrednost ovog flega eksplicitno zabranjuje fragmentaciju paketa, čak iako je on prevelik da se prenese u poruci nižeg sloja, u kom slučaju će biti uništen. Uobičajeno je da je ovaj fleg resetovan, odnosno da se dozvoljava fragmentacija.
- ***MF Flag*** (*More Fragment Flag*) – Setovana vrednost ovog flega ukazuje da postoje dodatni podaci iz originalnog paketa koji se prenose u jednom ili više narednih fragmenata. Samo fragment koji nosi poslednju sekvencu bajtova iz originalnog paketa nema setovan ovaj fleg.

Polje *Fragment Offset* ukazuje na rednu poziciju sekvenci bajtova iz fragmenta u odnosu na podatke u originalnom paketu, ali izraženo u jedinicama od po 8 bajtova. Drugim rečima, originalni podaci se dele u sekvence bajtova čije je dužina umnožak broja 8, a *Fragment Offset* sadrži bajtovsku poziciju fragmenta podeljenu sa brojem 8.

U slučaju potreba za fragmentacijom, podaci koji se prenose u paketu se dele na manje celine, od kojih se formiraju odvojeni IP paketi, svaki sa zaglavljem koje sadrži sledeće vrednosti:

- Polja *VERS*, *ToS*, *Identification*, *TTL* i IP adrese izvorišta i odredišta se nasleđuju iz zaglavlja originalnog paketa.
- Polje *Fragment Offset* se postavlja na odgovarajuću vrednost u zavisnosti od pozicije fragmenta u odnosu na podatke u originalnom paketu.
- *MF Flag* se setuje u svim fragmentima, osim u poslednjem.
- Polja *Header Length*, *Packet Length* i *Checksum* se računaju u odnosu na novoformirano zaglavlje.

Na primeru koji prikazuje Slika 5.5, podaci se dele u manje celine od po 1480 bajtova, kako bi zajedno sa IP zaglavljem činili niz od 1500 bajtova. Svaki naredni pomeraj (*Fragment Offset*) će biti uvećan za po 185 (vrednost 1480 podeljena sa brojem 8).



Slika 5.5. Primer fragmentacije IP paketa

Navedeni postupak fragmentacije se može ponovo primeniti i na prethodno već fragmentirane IP pakete u slučaju da se za to javi potreba, npr. kada se naiđe na link koji ima još manju MTU vrednost. I tada se podaci po istom principu dele na manje celine, uz preračunavanje relativne pozicije novog fragmenta u odnosu na podatke iz originalnog paketa.

Fragmentacija predstavlja dodatni posao za rutere, ali se zato reasembliranje sprovodi na odredišnom uređaju. A da bi se sprovelo reasembliranje, najpre je potrebno prepoznati fragmentirani IP paket. Ovo se sprovodi na osnovu vrednosti polja *Fragment Offset* i MF flega – samo nefragmentirani IP paketi imaju vrednosti nula i u polju *Fragment Offset* i u polju MF fleg, dok u IP paketima koji predstavljaju fragmente bar jedno od ova dva polja sadrži vrednost različitu od nule.

Na prijemnoj strani je pravilo da svi fragmentirani paketi najpre moraju da se objedine u originalni IP paket, kako bi se sadržani podaci celovito predali protokolu višeg nivoa. Proces reasembliranje se stoga sprovodi na sledeći način:

- Svi fragmenti istog originalnog paketa se prepoznaju na osnovu polja *Identification*.
- Za svaku jedinstvenu identifikaciju iz polja *Identification* alocira se memorija odgovarajuće veličine.
- Podaci iz pristiglih fragmenata se popunjavaju od pozicije na koju ukazuje polje *Fragment Offset*. Pri tome fragmenti ne moraju da pristižu u istom poretku, a nepopunjene pozicije se prepoznaju.
- Poslednji fragment se prepoznaje po resetovanom MF flegu, a ako su popunjene sve prethodne pozicije, smatra se da su podaci iz originalnog paketa u potpunosti rekonstruisati i oni se predaju protokolu višeg nivoa.
- U slučaju da se ne popune sve celine ni posle određenog vremena čekanja (*timeout* period), svi podaci ovog paketa se odbacuju i smatra se da je originalni paket izgubljen.

5.4 IP adrese

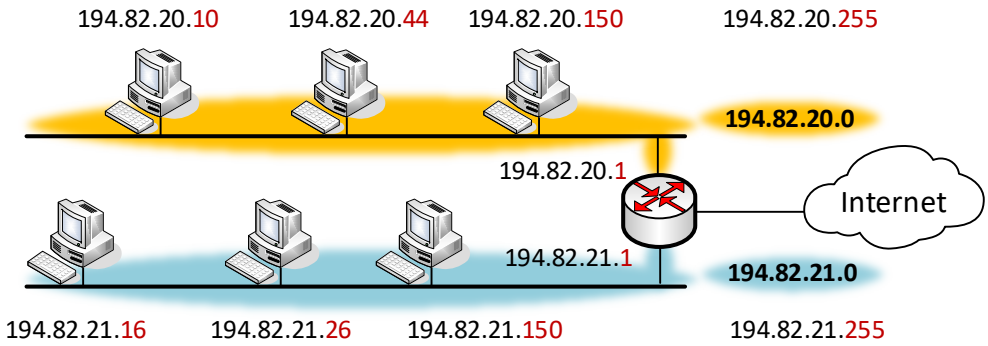
IP adrese u verziji 4 su veličine 4 bajta, što je pogodno za mašinsku obradu od strane rutera i drugih uređaja. Ipak, za korišćenje od strane ljudi, IP adrese se označavaju u tzv. „*dotted decimal*“ notaciji, gde se svaki bajt predstavlja kao dekadni broj u vrednosti od 0 do 255, a koji su razdvojeni tačkama (npr. 147.91.11.241).

Kao i MAC adrese, IP adrese se odnose na portove uređaja, koji se u terminologiji mrežnog sloja obično nazivaju interfejsi (*interface*). Uređaj sa više interfejsa može da ima više IP adresa, koje identifikuju uređaj. Primetimo da tom prilikom interfejs istovremeno ima i IP adresu i MAC adresu. Ali za razliku od MAC adresa, koje su fiksirane pri fabričkoj izradi i korisnik ne mora ni da ih poznaje, IP adrese se logički dodeljuju, odnosno konfigurišu od strane korisnika. Sloboda dodele IP adresa povlači za sobom i odgovornost, što zahteva određeno teorijsko znanje, ali i praktične veština upravljanja adresnim prostorom.

Osnovna pravila podrazumevaju sledeće:

- Uobičajeno je da se svaki segment na drugom nivou (LAN mreža, VLAN, ili *point-to-point* link) jednoznačno mapira u mrežni segment na trećem nivou, tzv. IP mrežu.
- IP adresa se tretira kao niz od 32 bita, koji se deli na dva dela:
  - Prvi deo bita veće težine naziva se „mrežni deo“ ili „prefiks“.
  - Ostatak adrese, odnosno biti manje težine, naziva „host deo“.
- IP adresa u kojoj host deo sadrži sve bitske nule, naziva se mrežna adresa i ona identifikuje pripadajuću IP mrežu.
- IP adresa u kojoj host deo sadrži sve bitske jedinice, predstavlja brodkast adresu pripadajuće mreže (*broadcast*), koja implicitno identifikuje sve uređaje na toj mreži. U praktičnoj primeni, IP paket koji sadrži brodkast adresu za odredište, završiće na svim uređajima u pripadajućoj IP mreži.
- IP adrese svih uređaja u jednoj IP mreži moraju da dele isti mrežni deo, dok host deo adrese može da uzima proizvoljne vrednosti, osim svih bitskih nula ili jedinica.

Posledica navedenih pravila je da se adrese uređaja u fizičkoj mreži grupišu u okviru iste mrežne adrese. Takođe, maksimalni broj uređaja u mreži ograničen je veličinom host dela, gde su dve adrese unapred rezervisane – mrežna adresa i brodkast adresa. Ako je broj bita u host delu n, ukupan broj adresa koje mreža može da sadrži je 2^n-2. Treba naglasiti da se host adrese dodeljuju svim interfejsima koji su fizički povezani na datu mrežu, uključujući i interfejse rutera.



Slika 5.6. Primer dve IP mreže povezane preko rutera

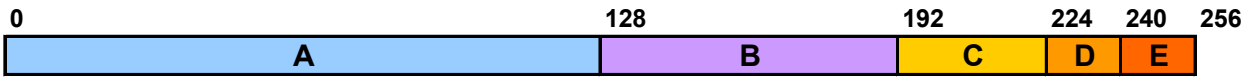
Slika 5.6 prikazuje dve fizičke LAN mreže koje su povezane preko rutera. Adrese su podeljene tako da tri bajta predstavljaju mrežni deo, a ceo četvrti bajt predstavlja host deo.

Sledeće pitanje koje se postavlja je kako se određuje granica između mrežnog dela i host dela?

Posmatrajmo najpre celokupni adresni prostor od adrese 0.0.0.0 do adrese 255.255.255.255, ali kao niz celih brojeva dužine 32 bita, počev od svih bitski nula, pa do svih jedinica. Pri inicijalnom kreiranju IP protokola, adresni prostor je podeljen na 5 delova, tzv. klasa (Slika 5.7).

- Klasa A – koja za početni bit ima vrednost „0“, pa time obuhvata polovinu adresnog prostora, gde je prvi bajt u opsegu dekadnih brojeva od 0 do 127.
- Klasa B – koja za početna dva bita ima „10“, pa time obuhvata četvrtinu adresnog prostora, gde je prvi bajt u opsegu dekadnih brojeva od 128 do 191.
- Klasa C – koja za početna tri bita ima „110“, pa time obuhvata osminu adresnog prostora, gde je prvi bajt u opsegu dekadnih brojeva od 192 do 223.
- Klasa D – koja za početna četiri bita ima „1110“, pa time obuhvata šesnaestinu adresnog prostora, gde je prvi bajt u opsegu dekadnih brojeva od 224 do 239.
- Klasa E – koja za početna četiri bita ima „1111“, pa time obuhvata šesnaestinu adresnog prostora, gde je prvi bajt u opsegu brojeva od 240 do 255.

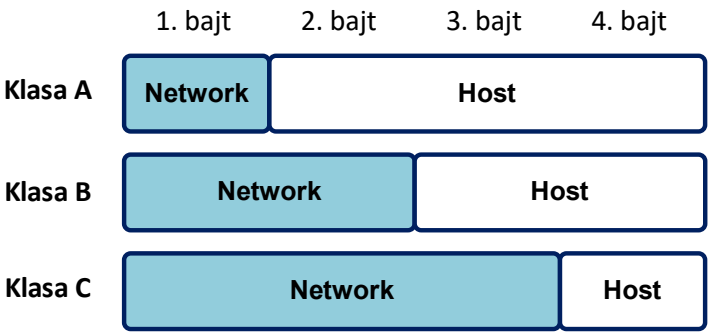
A	start	0	0	0	0	00000000	00000000	00000000	00000000
	end	127	255	255	255	01111111	11111111	11111111	11111111
B	start	128	0	0	0	10000000	00000000	00000000	00000000
	end	191	255	255	255	10111111	11111111	11111111	11111111
C	start	192	0	0	0	11000000	00000000	00000000	00000000
	end	223	255	255	255	11011111	11111111	11111111	11111111
D	start	224	0	0	0	11100000	00000000	00000000	00000000
	end	239	255	255	255	11101111	11111111	11111111	11111111
E	start	240	0	0	0	11110000	00000000	00000000	00000000
	end	255	255	255	255	11111111	11111111	11111111	11111111



Slika 5.7. Podela adresnog prostora na klase

Inicijalno je određeno da se za adresiranje mreža i uređaja mogu koristiti samo klase A, B i C, gde je mrežni deo za adrese u klasi A postavljen na jedan bajt, u klasi B na dva bajta, a u klasi C na tri bajta. Ovakav način fiksno uspostavljene granice između mrežnog i host dela prema pripadajućoj klasi naziva se ***classful* adresiranje** (Slika 5.8).

Klasa D je rezervisana za multikast adrese, dok su adrese iz klase E rezervisane samo za eksperimentalne potrebe.



Slika 5.8. Classful adresiranje – fiksna podela na mrežni deo i host deo u klasama A, B i C

Budući da IP adrese moraju da budu jedinstvene na nivou celog Interneta, njihovo globalno korišćenje mora biti usaglašeno, što koordinira organizacija IANA - *Internet Assigned Numbers Authority*. Adresni opsezi iz klasa A, B i C su dodeljeni Regionalnim Internet registrima (*Regional Internet Registry – RIR*), koji su organizovani po kontinentima. Da bi se koristile određene IP adrese, potrebno ih je najpre zakupiti kod pripadajućeg RIR-a, što tipično rade velike organizacije, kompanije i Internet provajderi. Manje organizacije i pojedinci se povezuju na lokalne Internet provajdere, koji im dodeljuju deo svojih ranije zakupljenih adresa. Iako se inicijalno smatralo da dužina IP adrese od 32 bita daje dovoljno veliki adresni prostor od oko 4.3 milijarde adrese, danas je situacija takva da su skoro sve adrese zauzete.

Čak ni sve adrese u klasama A, B i C nisu dozvoljene za javno korišćenje, već su rezervisane za posebne namene. Adrese koje su namenjene za izolovano korišćenje nezavisno od ostatka Interneta, nazivaju se privatne adrese i obuhvataju sledeće opsege:

- U klasi A – opseg od 10.0.0.0 do 10.255.255.255.
- U klasi B – opseg od 172.16.0.0 do 172.31.255.255.
- U klasi C – opseg od 192.168.0.0 do 192.168.255.255.

Takođe, u klasi A rezervisani su i sledeći opsezi za posebne namene:

- Opseg od 0.0.0.0 do 0.255.255.255 za adresiranje tzv. predefinisanih adresa (*default*).
- Opseg od 127.0.0.0 do 127.255.255.255 za implicitno adresiranje na lokalnom računaru (*loopback*).

Može se primetiti da *classful* adresiranje veoma nesrazmerno uspostavlja veličinu mrežnog i host dela adrese, posebno u klasama A i B, koje imaju preveliki host deo. Kako bi se veličina IP mreže što bolje prilagodila stvarnim potrebama (broju uređaja u mreži), potrebna je mogućnost fleksibilnog definisanja granice između mrežnog i host dela adrese. Ovo se postiže konceptom koji se naziva *Classless Inter-Domain Routing* (CIDR), uvođenjem novog podatka koji se pridružuje IP adresi, a koji se naziva maska [16].

U originalnoj interpretaciji maska se predstavlja kao niz od 32 bita, gde biti koji pripadaju mrežnom delu adrese sadrže jedinice, dok preostali biti u host delu sadrže nule. Naziv maska je proizašao iz ovakvog „binarnog maskiranja“ mrežnog dela i host dela IP adrese, a radi potrebe za mašinski brzim pretvaranjem IP adrese uređaja u adresu mreže. To se postiže primenom binarne operacije *AND* nad uparenim bitima IP adrese i maske, čime mrežni deo adrese ostaje nepromenjen, dok se biti u host delu postavljaju na nule. Potrebu za ovim imaju

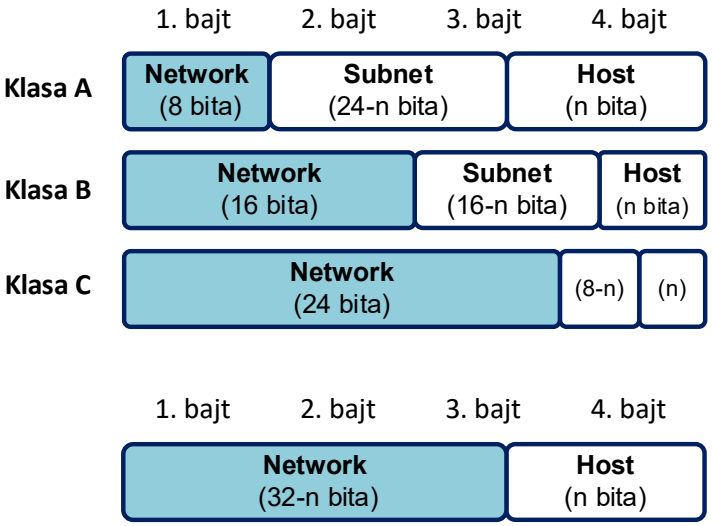
ruteri, koji za odredišne IP adrese uređaja iz zaglavlja paketa treba na odrede IP adrese pripadajućih mreža, kako bi pakete mogli da prosleđuju prema tim mrežama.

Kada se maska prikazuje zajedno sa IP adresom, uobičajena oznaka je oblika „a.b.c.d/n“, odnosno da se nakon IP adrese postavlja znak kose crte („/“) i broj bitnskih jedinica u masci (tzv. dužina maske). Primer adrese sa maskom je 147.91.11.241/27.

Maska se često prikazuje i odvojeno od IP adrese, npr. u konfiguracijama ili statusnim porukama. Tada se maska uobičajeno prikazuje u *dotted decimal* notaciji na isti način kao i IP adresa. Na primer, implicitno podrazumevane maske pojedinačnih klasa u *classful* interpretaciji IP adresa su:

- Klasa A: 255.0.0.0 (maska sa 8 binarnih jedinica)
- Klasa B: 255.255.0.0 (maska sa 16 binarnih jedinica)
- Klasa C: 255.255.255.0 (maska sa 24 binarne jedinice)

Ipak, pravi smisao maske je da se mrežne adrese uspostavljaju nezavisno od klasa. Tom prilikom maska može da sadrži proizvoljan broj jedinica, pa time granica između mrežnog i host dela može da bude unutar bilo kog bajta. U kontekstu klasa A, B i C, maska može da preklopi host deo, koji se time smanjuje na račun dela koji je interpretiran kao „podmreža unutar mreže koju određuje klasa“ (*subnet*), što ilustruje Slika 5.9. Ipak, koncept klasa je ubrzo odbačen, tako da veličinu mrežnog i host dela određuje isključivo maska.



Slika 5.9. Classless adresiranje – proizvoljna podela na mrežni deo i host deo

Za prikazivanje maske čija dužina ne odgovara granici između bajtova IP adrese, potrebno je određeno poznavanje binarno-dekadne konverzije. Primera radi, ako maska sadrži 27 binarnih jedinica i 5 preostalih nula, poslednji bajt ima binarnu vrednost „11100000“. Pretvaranje u dekadnu vrednost se dobija sabiranjem eksponenata broja 2 na mestima sa binarnim jedinicama, odnosno:

$2^7+2^6+2^5 = 128+64+32 = 224$

Na ovaj način se dobija maska 255.255.255.224. Do istog rezultata, ali sa manje binarno-dekadne aritmetike, dolazi se „inverznom“ računicom posmatranjem broja bitskih nula u bajtu gde se maska završava. Ovaj broj se tretira kao eksponent broja 2, što se oduzme od broja

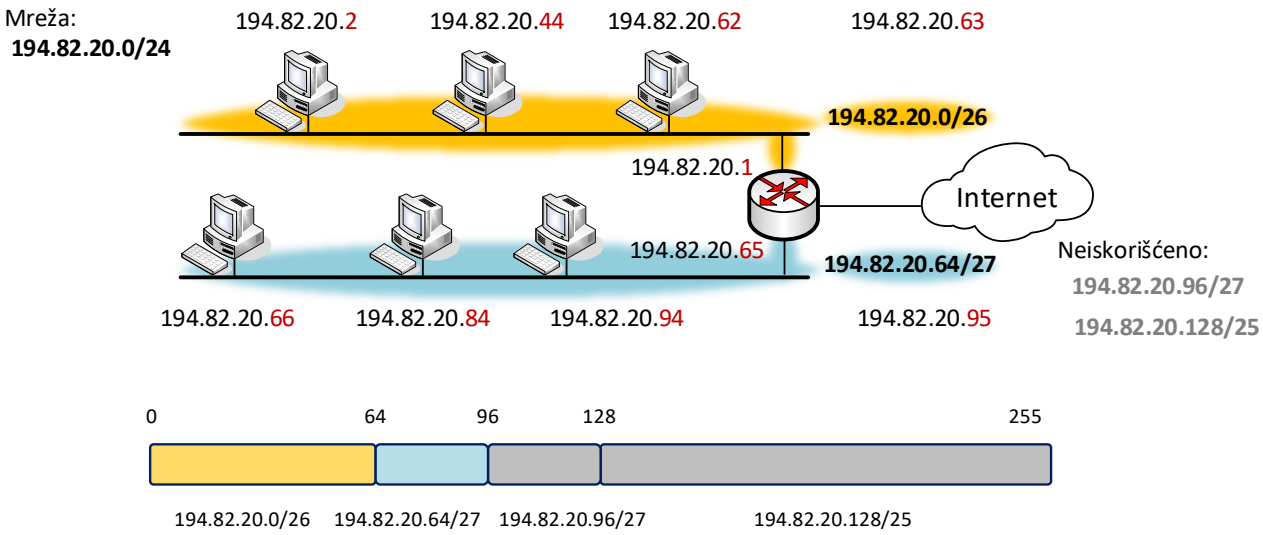
256. U prethodnom primeru maska sadrži 5 nula u poslednjem bajtu i koristi se kao eksponent broja 2, za koji se umanjuje broj 256, odnosno:

$256 - 2^5 = 256-32 = 224$

Koncept podele veće mreže na više manjih podmreža se zadržao, budući da se povećavanjem dužine maske inicijalna mreža deli na više manjih jednakih delova, a na račun host dela koji se za toliko smanjuje. Na ovaj način se mrežne IP adrese mogu hijerarhijski organizovati. Unutar dodeljenog adresnog prostora na nivou određene organizacije, dužina maske pojedinačnih IP mreža može da bude različita, što predstavlja koncept pod nazivom varijabilna dužina maske (*Variable Length Subnet Mask – VLSM*).

Generalno posmatrano, raspoloživi adresni opseg se posmatra kao kontinuirani blok adresa, koji se deli prema potrebama na manje IP mreže različitih viličina. Pri tome veličina IP mreže, izražena kroz broj mogućih IP adresa u mreži, nije proizvoljna, već predstavlja eksponent broja 2: npr. 16, 32, 26, 128 itd. Takođe, najmanji adresni blok koji se može dodeliti nekoj mreži ima masku dužine 30, odnosno sadrži dva bita u host delu. Time se može adresirati ukupno 4 adrese, od kojih prva adresa predstavlja IP adresu mreže, poslednja adresa je brodkast adresa, dok se samo preostale dve adrese mogu koristiti za uređaje u mreži. Ograničeno na samo dve efektivne adrese, maska dužine 30 se po pravilu koristi za adresiranje direktnih *point-to-point* linkova između rutera.

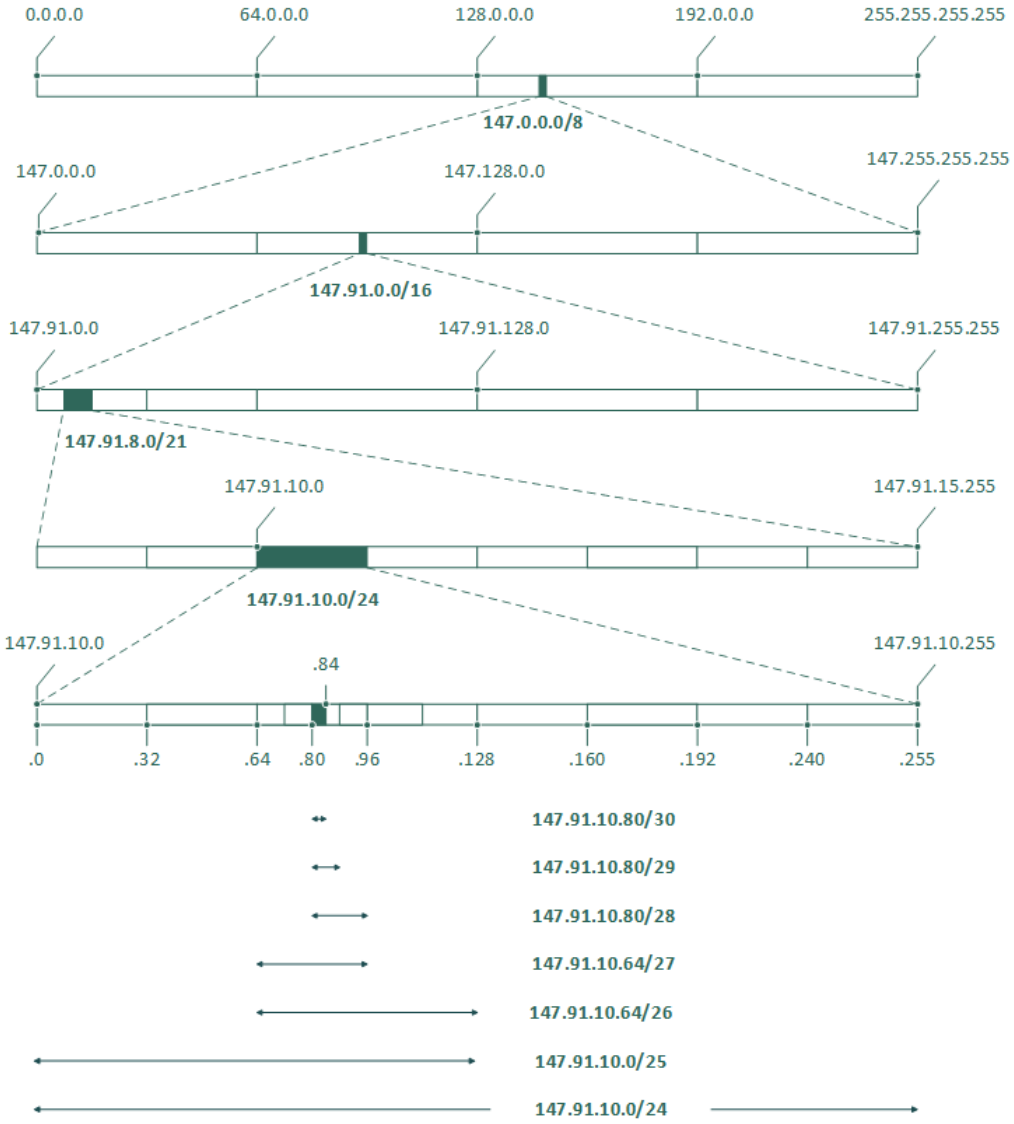
Posmatrajmo dalje primer kada je na raspolaganju mrežna IP adresa 194.82.20.0/24, koja predstavlja blok od 256 adresa, i potrebu da se realizuju dve međusobno povezane, ali nezavisne mreže sa po 50 i 20 uređaja. Najefikasnije korišćenje adresa se postiže ako se prvo alocira blok od 64 adrese za mrežu sa 50 uređaja, a zatim blok od 32 adrese za mrežu sa 20 uređaja. Time se dobijaju dve podmreže sa različitim maskama: 194.82.20.0/26 i 194.82.20.64/27. Neiskorišćene adrese su u opsegu od 194.82.20.96 do 194.82.20.255, ali se one ne mogu objediniti jednom mrežnom adresom, već se predstavljaju preko dva preostala bloka: 194.82.20.96/27 I 194.82.20.128/25 (Slika 5.10).



Slika 5.10. Primer podele adresnog bloka 194.82.20.0/24 na dve IP mreže

Prethodni primer ujedno demonstrira da se IP podmreže, odnosno manji blokovi IP adresa, čak ni sa maskama ne mogu proizvoljno formirati, već njihova granica može da bude samo na nekom od umnožaka eksponenta broja dva (u gornjem primeru 64, 96, 128). Bilo koji adresni

blok, uključujući i celokupni adresni prostor, je potrebno posmatrati kao hijerarhiju blokova, od kojih je svaki nastao podelom na manje segmente iste veličine, čiji broj predstavlja eksponent broja 2 (npr. podela na 2, 4, 8, 16 itd.). Slika 5.11 prikazuje mali adresni blok 147.91.10.80/30, koji pripada IP mreži Elektrotehničkog fakulteta Univerziteta u Beogradu 147.91.8.0/21, kao deo većeg adresnog bloka 147.91.0.0/16 koji pripada Akademskoj mreži Srbije (AMRES), što sve zajedno čini tek delić hijerarhije celokupnog adresnog prostora.



Slika 5.11. Hijerarhija adresnog prostora na primeru Elektrotehničkog fakulteta Univerziteta u Beogradu

Kao što se povećanjem dužine maske odgovarajući adresni blok deli na manje celine (podmreže), tako se i smanjenjem dužine maske više podmreža objedinjuje u jednu veću zajedničku celinu, što se naziva **agregacija**. Kao i kod podele na podmreže, ni agragacija se ne može sprovoditi na proizvoljan način, već se mora poštovati hijerarhija koja proizilazi iz prethodno opisane podele blokova bazirane na eksponentima broja 2.

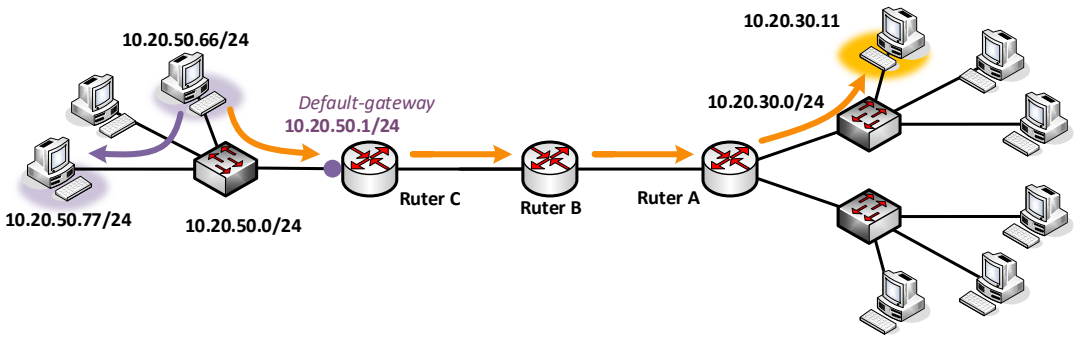
Uvođenje maske kroz koncept koji je nazvan *Classless Inter-Domain Routing*, ukazuje na osnovnu primenu za potrebe rutiranja – procesa koji IP pakete sprovodi kroz mrežu, prosleđujući ih do krajnjeg odredišta, što je predmet narednog poglavlja.

6. Principi rutiranja

Da bi komunicirao na IP nivou svaki uređaj mora da ima dodeljenu IP adresu i masku. Ipak, da bi se poslao IP paket do odredišnog uređaja, potrebno je znati samo njegovu IP adresu, ali ne i masku. Izvorišni uređaj preko svoje IP adrese i maske poznaje IP mrežu kojoj pripada, na osnovu čega utvrđuje da li odredište pripada istoj ili različitoj IP mreži. Kada se formira IP paket, on se enkapsulira u Eternet okvir, pa je potrebno odrediti i MAC adresu koja se postavlja u polje odredišne adrese Eternet zaglavlja.

U prvom slučaju, kada izvorište i odredište pripadaju istoj IP mreži, koja je ujedno i ista LAN mreža, za odredišnu adresu u Eternet zaglavlju se postavlja MAC adresa odredišnog uređaja, pa se IP paket posredstvom Eternet komunikacije direktno prenosi do odredišta.

U drugom slučaju, kada se prepozna da je odredište u drugoj IP mreži, a time i u drugom LAN-u, komunikacija se ne može direktno ostvariti preko Eterneta. Paket u tom slučaju mora da izađe iz izvorišne LAN mreže i da u ostatku „IP prostora“ nađe put do odredišta. Izlaz iz LAN mreže je interfejs rutera na koji je mreža povezana, a čija IP adresa pripada istoj IP mreži kao i ostali uređaji u LAN mreži. Za sve uređaje u posmatranoj LAN mreži ovaj interfejs rutera se naziva „difoltni gejtvej“ (***default gateway***). IP paket koji se šalje za odredište van LAN mreže postavlja se u Eternet okvir koji za odredišnu adresu ima MAC adresu difoltnog gejtveja, dok se u IP zaglavlju navodi IP adresa odredišta.



Slika 6.1. Primer komunikacije sa uređajima unutar iste i različite LAN mreže

U primeru koji demonstrira Slika 6.1, računar sa adresom 10.20.50.66/24 direktno šalje pakete do računara 10.20.50.77, budući da se nalaze u istoj LAN mreži, odnosno dele istu IP mrežu. Sa druge strane, IP adresa 10.20.30.11 pripada drugoj IP mreži, pa se IP paket, enkapsuliran u Eternet paket, šalje najpre do difoltnog gejtveja, sa adresom 10.20.50.1. Nakon toga ruter preuzima Eternet okvir, gleda sadržaj pripadajućeg IP paketa i odlučuje na koji izlazni interfejs treba da ga prosledi. Ovaj proces se naziva **rutiranje** (*routing*).

Podsetimo se da svičevi prosleđuju Eternet pakete na osnovu odredišne MAC adrese iz zaglavlja paketa i bridžing tabele, koja za svaku MAC adresu u LAN mreži ukazuje na port sviča koji vodi do nje. Proces rutiranja koji sprovode ruteri se takođe sprovodi na osnovu odredišne adrese, što se naziva ***destination-based* rutiranje**. Slično svičevima, i ruteri imaju tabele na osnovu kojih se IP paketi prosleđuju prema odredištima, koje se nazivaju tabele rutiranja, odnosno **ruting tabele** (*routing table*).

Jedan red u ruting tabeli naziva se **ruta** (*route*) i sastoji se od sledeća dva osnovna podatka.

- IP adresa mreže, čiji je integralni deo i maska.

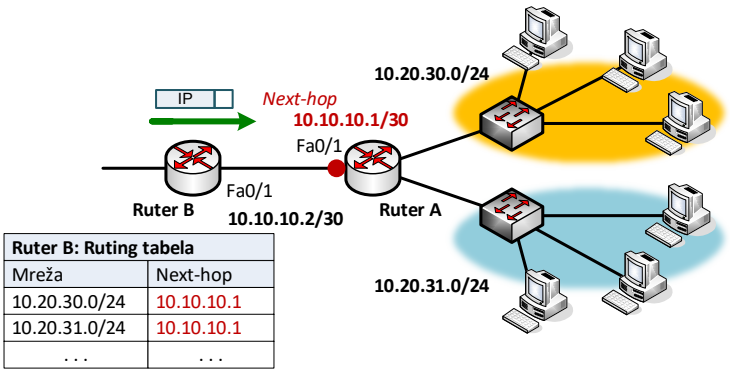
- IP adresa interfejsa sledećeg rutera na putu koji vodi do odredišne mreže, što se naziva ***next-hop*** adresa.

Na osnovu ruta iz ruting tabele, ruter zna za postojanje određenih IP mreža, kao i preko kog narednog rutera vodi put do njih. Na ovaj način, svaki ruter iz svoje pozicije „sagledava“ ostatak mreže, pa stoga i svaki ruter ima jedinstvenu ruting tabelu, različitu od drugih.

Osim što ruting tabele sadrže informacije na IP nivou, primetimo da postoje još dve razlike u odnosu na bridžing tabele.

- Ruting tabela ne sadrži adrese odredišnih uređaja, već adrese odredišnih mreža.
- *Next-hop* podatak nije izlazni interfejs pripadajućeg rutera (kao što je to port na sviču), već interfejs sledećeg rutera na putu koji vodi do odredišne mreže. Razlog je što se na izlaznom segmentu može nalaziti LAN mreža sa više rutera, pa je potrebno specificirati koji od njih predstavlja sledeći korak na putu.

Na primeru koji ilustruje Slika 6.2, Ruter B na osnovu svoje ruting tabele zna da postoje mreže 10.20.30.0/24 i 10.20.31.0/24, ne tačno i gde se one nalaze, ali da put do njih vodi preko *next-hop* adrese 10.10.10.1, koja predstavlja direktno povezani interfejs narednog Rutera A na putu do njih.



Ruter B: Ruting tabela

Mreža	Next-hop
10.20.30.0/24	10.10.10.1
10.20.31.0/24	10.10.10.1
. . .	. . .

Slika 6.2. Ruting tabela

Proces rutiranja sa aspekta jednog rutera svodi se na sledeće:

- Za svaki L2 okvir koji pristigne na neki od interfejsa, izdvaja se pripadajući IP paket.
- Iz zaglavlja IP paketa uzima se odredišna IP adresa, koja se uparuje sa odgovarajućom IP adresom mreže iz ruting tabele.
- Za nađenu IP adresu mreže uzima se pripadajuća *next-hop* adresa.
- IP paket se prosleđuje na *next-hop* adresu, posredstvom novog L2 okvira u koji je enkapsuliran.

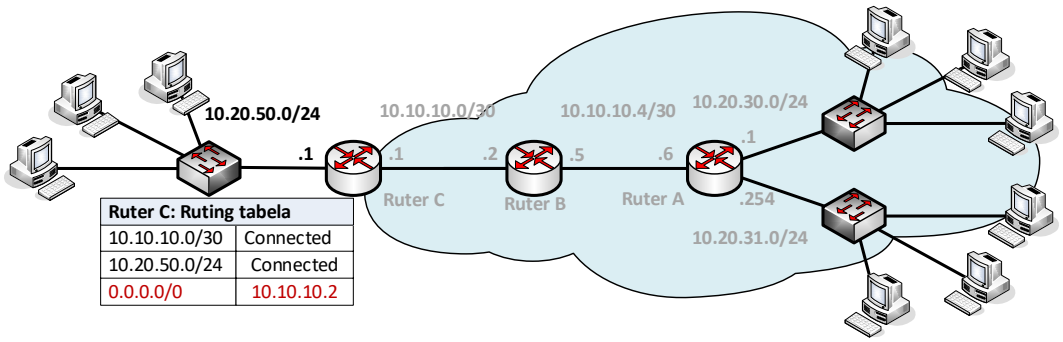
Na ovaj način svaki ruter nezavisno utiče na prosleđivanje paketa. Pod pretpostavkom da svi ruteri poseduju usaglašene i tačne podatke o IP mrežama, svaki paket će kroz niz koraka da pronađe put do svog odredišta.

Prethodno je istaknuta hijerarhijska organizacija IP adresa, gde jedna veća IP mreža, obuhvata više manjih. U slučaju da se u ruting tabeli nađu ovakve IP mreže, gde jedna sadrži drugu, odredišna adresa može da bude uparena sa obe IP mreže, pa se postavlja pitanje koja od ovih ruta će da se koristi?

Pravilo je da ako odredišna adresa zadovoljava više IP mreža u ruting tabeli, tada se bira „**najspecifičnija ruta**“ (*more specific route*), odnosno mreža koja ima najveću masku (*the longest match*). Ovaj princip je i logičan, budući da manja podmreža donosi veći nivo specifičnosti i preciznosti u odnosu na veću mrežu koja je obuhvata.

Primetimo da najveća moguća IP mreža, koja agregira sve ostale IP mreže, kao i sve host adrese, obuhvata celokupni adresni prostor, i predstavljena je preko adrese 0.0.0.0/0. Prema prethodnom pravilu najspecifičnije rute, ova mreža je najmanje specifična, a ruta u kojoj se ona javlja će se koristiti jedino ako ne postoji ni jedna druga ruta koja može da se upari. Ovakva ruta se naziva **difoltna ruta** (*default route*).

Difoltna ruta je izuzetno korisna, jer omogućava „podrazumevano rutiranje“ za sve mreže koje nemaju svoje rute u ruting tabeli, čime se ruting tabela može značajno smanjiti. U mnogim slučajevima, fizički put do većine, ako ne i svih preostalih mreža, vodi samo preko jedne veze, pa sve ostale rute mogu da se zamene sa difoltnom rutom (Slika 6.3).



Slika 6.3. Difoltna ruta za rutiranje prema svim ostalim mrežama

Naredno pitanje koje se postavlja je kako ruteri uspostavljaju svoje ruting tabele?

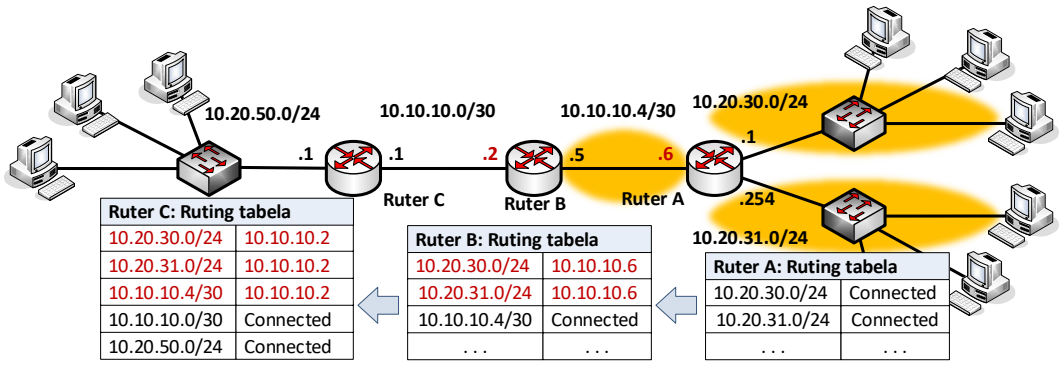
Najpre napomenimo da za razliku od svičeva, ruteri zahtevaju određeno inicijalno podešavanje, koje se sprovodi određenim konfiguracionim komandama. Minimalno je potrebno postaviti IP adrese i maske na interfejse rutera. Na osnovu toga ruter će da prepozna IP mreže na koje je direktno povezan i najpre će njih da upiše u ruting tabelu. Ovakve mreže nemaju *next-hop* atribut koji ukazuje na drugi ruter, već imaju poseban status koji ukazuje da je mreža direktno povezana na ruter (*connected*).

Postoji mogućnost da mrežni administratori kroz konfiguraciju rutera dodaju željenu rutu u ruting tabelu, navodeći IP adresu mreže i *next-hop*. Ovakve rute se nazivaju **statičke rute**, a njihovo korišćenje se naziva **statičko rutiranje**. Ovaj pristup je ipak nepodesan na iole većim mrežama, budući da zahteva manuelni rad koji je podložan greškama. Ipak, daleko veći problem je „statičnost“ statičkih ruta, budući da su one fiksne i ne mogu se prilagoditi eventualnim promenama u mreži. Statičke rute će uvek da prosleđuju pakete na definisanu putanju, čak iako je neka veza na putanji u prekidu, što će dovesti do gubitka paketa, odnosno prekida u komunikaciji, iako možda postoji druga putanja do odredišta.

Daleko veća fleksibilnost se postiže automatskim prepoznavanjem gde se nalazi koja IP mreža, kroz proces „učenja“ ruta i popunjavanja ruting tabela, što se naziva **dinamičko rutiranje**. Tom prilikom ruteri međusobno komuniciraju i razmenjuju informacije o rutama posredstvom određenih protokola rutiranja. Iako se protokoli rutiranja detaljno opisuju u

posebnom poglavlju, na ovom mestu ćemo ilustrovati osnovni princip razmene ruta putem ruting protokola (Slika 6.4).

Preko adresa svojih interfejsa, ruteri inicijalno poznaju samo direktno povezane IP mreže. U posmatranom primeru, Ruter A obaveštava Ruter B o sadržaju svoje ruting tabele, koja se sastoji od direktno povezanih mreža (10.20.30.0/24 i 10.20.31.0/24). Ruter B upisuje ove mreže u svoju ruting tabelu i pridružuje im *next-hop* adresu preko koje ih je naučio, u ovom slučaju adresu 10.10.10.6 (na slici je radi preglednosti za adrese interfejsa prikazan samo poslednji bajt). Ovaj proces oglašavanja ruta se nastavlja i prema Ruteru C, uključujući i rute za direktno povezane mreže na Ruteru B (10.10.10.4/30).



Slika 6.4. Dinamičko oglašavanje ruta posredstvom protokola rutiranja

Velika prednost je što se kod dinamičkog rutiranja rute automatski razmenjuju, što dovodi do samostalnog uspostavljanja ruting tabela. Ova osobina ih čini skalabilnim, jer se mreža može jednostavno proširivati dodavanjem novih rutera, veza i IP mreža. Još značajnije, što opravdava naziv „dinamičko“, je činjenica da se ruting tabele adaptiraju na promene u mreži. U slučaju prekida pojedinih veza, ruting tabele će da uspostave novo stanje koje omogućava pronalaženje alternativnih putanja do pojedinih odredišta, ako postoje.

U slučaju postojanja redundantnih veza u mreži, primenjeni ruting protokol će otkriti sve putanje do odredišnih mreža. Da bi se odlučilo koju putanju zadržati u ruting tabeli, potrebno je uvesti meru njihovog kvaliteta, odnosno značaja za rutiranje paketa. Ovaj parametar se naziva **metrika**. Metrika može biti izražena preko različitih veličina, a najznačajnije su sledeće:

- *Hop-count* – broj koraka na putanji, odnosno broj veza ili rutera do odrešne mreže. Manja vrednost je bolja, zato što odražava kraću putanju.
- *Bandwidth* – propusni opseg ili kapacitet prenosa podataka po putanji. Veća vrednost je bolja, jer omogućava protok veće količine podataka u jedinici vremena.
- *Cost* – cena putanje, koja se obično izvodi iz protoka kao zbir obrnuto proporcionalnih vrednosti svih linkova na putanji, ali može biti i posebno postavljena na proizvoljnu vrednost. Kao i svaka cena, manja vrednost je bolja.
- *Delay* – kašnjenje koje postoji na vezi ili putanji, što zavisi od fizičke dužine i realizacije veze. Primera radi, satelitske veze imaju veće kašnjenje od zemaljskih veza, nezavisno od kapaciteta prenosa podataka.
- *Load* – opterećenje veze u određenom periodu, kako bi se dao veći prioritet manje opterećenim vezama, a više opterećene veze rasteretile.

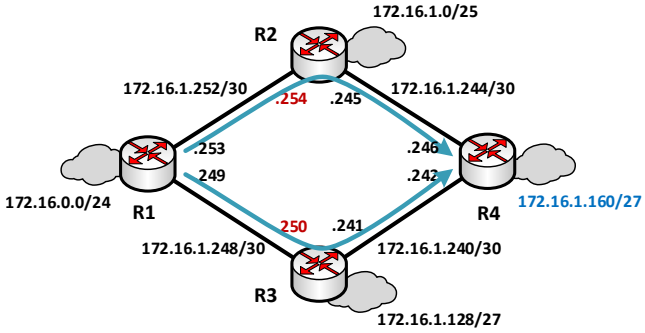
- *Reliability* – pouzdanost veze, koja se smanjuje u slučaju kada na vezi postoje učestali problemi.

Prvobitni ruting protokoli su obično primenjivali *hop-count* metriku zbog svoje jednostavnosti i intuitivnosti. Ipak, ova veličina je za korisnike jedva primetna i manje značajna, za razliku od kapaciteta prenosa podataka, koji se direktno odražava na kvalitet komunikacije. Iz toga razloga popularniji su ruting protokoli koji za metriku imaju cenu (*cost*) koja se izvodi iz kapaciteta, ali se može i dodatno podešavati. Takođe neki ruting protokoli mogu kombinovati više metrika sa određenim težinskim faktorima, čime se dobija kompozitna metrika u apstraktnim jedinicama.

Imajući u vidu konkretnu metriku, pravilo je da u slučaju više putanja, odnosno ruta, ruting protokol zadržava samo onu rutu koja ima najbolju vrednost metrike (najmanju ili najveću vrednosti, u zavisnosti od primenjene metrike) i samo nju upisuje u ruting tabelu.

U slučaju da više ruta imaju istu i najbolju vrednost metrike, tada će sve one da budu upisane u ruting tabelu. U ruting tabeli će ova situacija da se prepozna tako što će jedna IP mreža imati dva ili više *next-hop* atributa.

R1: Ruting tabela		
Mreža	Met	Next-hop
172.16.0.0/24	0	Connected
172.16.1.0/25	1	172.16.1.254
172.16.1.128/24	1	172.16.1.250
172.16.1.160/27	2	172.16.1.250
		172.16.1.254



Slika 6.5. Balansiranje saobraćaj kada više ruta imaju iste vrednosti metrike

U tom slučaju saobraćaj će da se podeli, odnosno balansira približno ravnomerno na višestruke rute iz ruting tabele, što se naziva **balansiranje saobraćaja** (*load balancing*). Ova mogućnost je generalno korisna, jer se rasterećuju pojedinačni linkovi, ali može delimično i da oteža praćenje rada mreže i dijagnosticiranje eventualnih problema.

Jedan ruter može istovremeno da koristi više ruting protokola, a svi oni postavljaju najbolje rute u jedinstvenu i zajedničku ruting tabelu. Problem koji se tom prilikom javlja je što primenjeni ruting protokoli mogu da imaju različite metrike koje nisu uporedive (npr. broj koraka i cena). U slučaju da različiti ruting protokoli pronalaze rute do iste IP mreže, metrika je neupotrebljiva veličina, jer se ne može odlučiti koja ruta je bolja. U takvim slučajevima koristi se generalno ustanovljen prioritet ruting protokola, što se naziva **administrativna distanca** (*administrative distance*). Manja vrednost administrativne distance ima veći prioritet, što je dodeljeno protokolima koji se smatraju za pouzdanije.

Pravilo je da se u slučaju poređenja ruta iz više ruting protokola, u ruting tabeli kao najbolja ruta zadržava samo jedna i to ona koja je naučena od ruting protokola sa najmanjom administrativnom distancom.

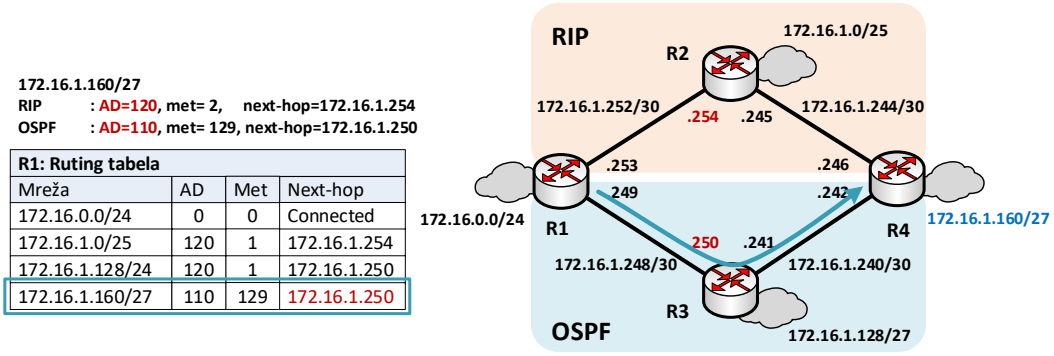
Tabela 6.1 prikazuje vrednosti administrativnih distanci za pojedine ruting protokole. Primećuje se da najbolju administrativnu distancu imaju rute za mreže koje su direktno povezane na ruter

(*connected*). Nakon toga sledeće su statičke rute koje za administrativnu distancu imaju vrednost 1. Tek nakon toga slede stvarni protokoli rutiranja.

Ruting protocol	Administrativna distanca
Connected	0
Static	1
EIGRP summary	5
BGP external	20
EIGRP internal	90
IGRP	100
OSPF	110
RIP	120
EIGRP external	170
BGP internal	200

Tabela 6.1. Administrativne distance za pojedine ruting protokole

Na primeru koji pokazuje Slika 6.6, ruter R1 za mrežu 172.16.1.160/27 dobija dve rute – jednu preko RIP protokola, a drugu preko OSPF protokola, koji su tema narednih poglavlja. U ruting tabelu se upisuje samo jedna ruta i to iz OSPF protokola, budući da OSPF ima manju administrativnu distancu u odnosu na RIP.



Slika 6.6. Izbor samo jedne rute iz ruting protokola koji ima bolju administrativnu distancu

U iole složenijim mrežama koristi se dinamičko rutiranje, odnosno uspostavljanje određenog protokola rutiranja. I pred toga, za pojedine izolovane slučajeve mogu se dodatno koristiti i pojedinačne statičke rute. U oba slučaja, rezultat je popunjavanje ruting tabele odgovarajućim rutama.

Primer ruting tabele i komande koja je izlistava (*show ip route*) na ruterima proizvođača *Cisco Systems* prikazuje Slika 6.7.

```
Router>show ip route
Routing entry for 147.91.0.0/16, 304 known subnets
  Attached (5 connections)
  Variably subnetted with 12 masks
O E1    212.200.228.92 [110/32] via 147.91.7.65, 00:00:35, Ethernet1/0
O E2 212.62.33.0/24 [110/20] via 147.91.7.77, 00:00:35, Ethernet1/0
     147.91.0.0/16 is variably subnetted, 304 subnets, 12 masks
O IA       147.91.200.0/22 [110/12] via 147.91.7.93, 18:49:41, Ethernet1/0
O IA    147.91.132.64/28 [110/12] via 147.91.7.92, 18:49:41, Ethernet1/0
O IA    147.91.4.192/28 [110/11] via 147.91.7.65, 18:49:41, Ethernet1/0
C       147.91.7.64/26 is directly connected, Ethernet1/0
O IA    147.91.217.128/25 [110/11] via 147.91.7.96, 18:49:43, Ethernet1/0
O E2    147.91.221.128/25 [110/20] via 147.91.7.65, 02:39:31, Ethernet1/0
O*E2 0.0.0.0/0 [110/1] via 147.91.7.65, 00:04:37, Ethernet1/0
O       147.91.0.83/32 [110/11] via 147.91.7.117, 18:49:43, Ethernet1/0
O IA    147.91.212.128/26 [110/12] via 147.91.7.65, 18:49:43, Ethernet1/0
```

Slika 6.7. Primer ruting tabele

Ako se posmatra ruta za mrežu 147.91.132.64/28, ona sadrži informacije koje prikazuje Tabela 6.2.

Labela	Značenje
O IA	Labela koja ukazuje kako je ruta naučena (u ovom slučaju preko OSPF ruting protokola („O“), tzv. *inter-area* ruta („IA“)
147.91.132.64/28	IP adresa mreže na koju se ruta odnosi
[110/12]	Prvi broj označava administrativnu distancu (110 za OSPF). Drugi broj označava metriku.
via 147.91.7.92	*Next-hop* adresa
18:49:41	Vreme kada je ruta upisana u tabelu
Ethernet1/0	Interfejs rutera koji vodi do *next-hop* adrese

Tabela 6.2. Opis informacija iz ruting tabele

Prethodno prikazani ruting protokoli se odnose na mreže pojedinačnih organizacija, koje mogu biti i veće korporacije, internet provajderi, državne ili akademske mreža na nacionalnom nivou i slično, a koje se nazivaju **autonomni sistemi** (*Autonomous Systems* - AS). Oni se odlikuju po tome što pripadaju zajedničkoj organizaciji i imaju usaglašene politike upravljanja i principe rada od strane zajedničkog tima mrežnih administratora, dele određeni blok IP adresa, imaju usaglašen rad protokola rutiranja, koji su efikasni i skalabilni čak i velikim mrežama, od nekoliko stotina ili hiljada rutera. Autonomni sistemi koji su povezani na Internet imaju svoje jedinstvene identifikacije u vidu celobrojnih vrednosti veličine dva bajta, koji se nazivaju brojevi autonomnih sistema (*AS number*).

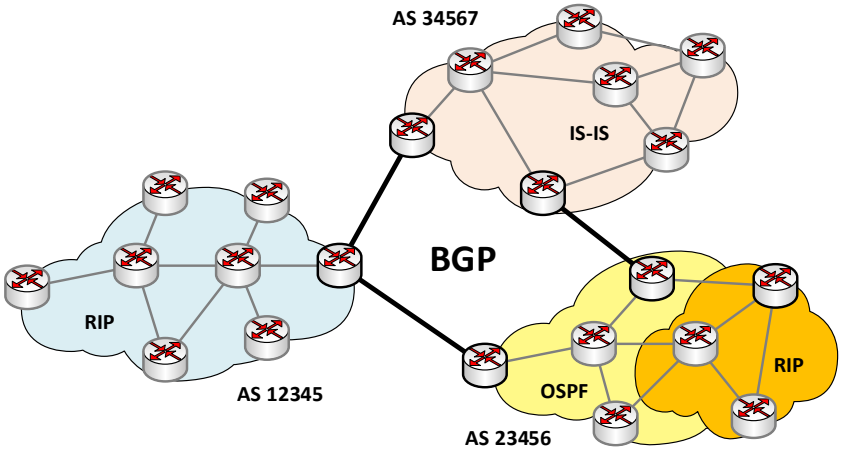
Čak i velike mreže pojedinačnih autonomnih sistema su zanemarljive u odnosnu na celokupni Internet. Takođe je i upravljanje mrežama unutar autonomnog sistema u odnosu na upravljanje na Internetu značajno različito, budući da je Internet u velikoj meri decentralizovan i deljen od strane velikog broja nezavisnih mreža. Uspostavljanje i deljenje ruta na Internetu stoga zahteva nešto drugačiji koncept, odnosnu drugačiji protokol rutiranja u odnosu na rutiranje unutar autonomnih sistema.

Imajući u vidu prethodno izneto, ruting protokoli se dele na dve osnovne grupe (Slika 6.8):

- **Interni ruting protokoli** – Ruting protokoli namenjeni za rad unutar jednog autonomnog sistema. Otkrivaju i prenose rute za sve mreže iz pripadajućeg autonomnog sistema, ali ne

van njega. U jednom autonomnom sistemu mogu da koegzistiraju više internih ruting protokola, a oblast koju pokriva jedan protokol naziva se **ruting domen**.

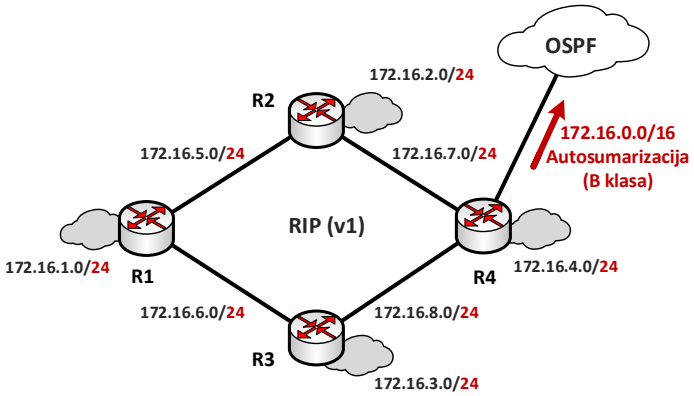
- **Eksterni ruting protokoli** – Ruting protokoli namenjeni za rad na celokupnom Internetu, posmatrajući ga kao mrežu velikog broja povezanih autonomnih sistema. Rute sadrže i informaciju o autonomnim sistemima na putanjama do odredišnih mreža, koje obično predstavljaju agregirane IP blokove celih autonomnih sistema.



Slika 6.8. Interni protokoli rutiranja unutar autonomnih sistema i eksterni protokoli rutiranja između njih

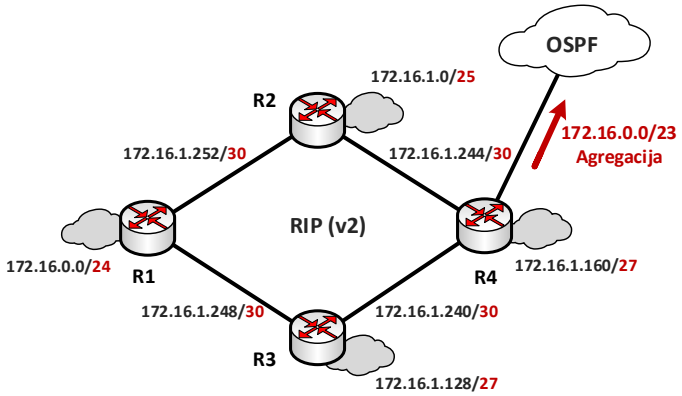
Interni ruting protokoli su raznovrsniji, a najpoznatiji primeri su RIP, OSPF, IS-IS, EIGRP itd. Sa druge strane, na Internetu je u primeni praktično samo jedan eksterni ruting protokol, a to je BGP (*Border Gateway Protocol*). BGP protokol izlazi van okvira ove knjiga, pa se u nastavku razmatraju samo interni protokoli rutiranja.

U „pionirskim“ danima IP komunikacija, u doba podele adresa na klase A, B i C, prvobitne verzije pojedinih protokola rutiranja uz IP adresu mreže nisu prenosili pripadajuću masku. Ova vrsta ruting protokola se naziva ***classful***. Istina, mreže su mogle da koriste masku i time se podele na manje celine, ali je maska morala da bude ista za sve mreže u tom ruting domenu. Konkretnu dužinu maske ruteri su mogli da prepoznaju na osnovu IP adresa svojih interfejsa, a koje sadrže masku. Implicitna zavisnost od pripadajuće klase adresa je ipak ostala pri povezivanju sa drugim ruting protokolom. Tada se IP adrese iz celog ruting domena agregiraju u jednu adresu mreže koja zaista ima masku pripadajuće klase (npr. maska dužine 16 za klasu B, što prikazuje Slika 6.9). Ovakva implicitna dodela maske prema pripadajućoj klasi se naziva **automatska sumarizacija adresa** (*autosummary*). Ne samo što automatska sumarizacija nije optimalna, već može da izazove i koliziju sa sličnim mrežama iz iste klase.



Slika 6.9. Classful ruting protokol i automatska sumarizacija adresa sa maskom pripadajuće klase

Primena varijabilne dužine maske (VLSM) čini da maska postaje sastavni deo IP adrese mreže, čime se i ona mora razmenjivati preko ruting protokola. Tada nema razloga ni za automasku sumarizaciju adresa na nivou klase, već se „izvoz“ IP adrese celokupne mreže sprovodi eksplicitnom agregacijom u optimalni adresni blok koji ih objedinjuje (Slika 6.10). Ruting protokoli ovog tipa nazivaju se ***classless***.



Slika 6.10. Classless ruting protokol i podrška za varijabilnu dužinu maske

Nije teško zaključiti da se već dugo koriste isključivo *classless* ruting protokoli, koji se dele na sledeće dve osnovne grupe:

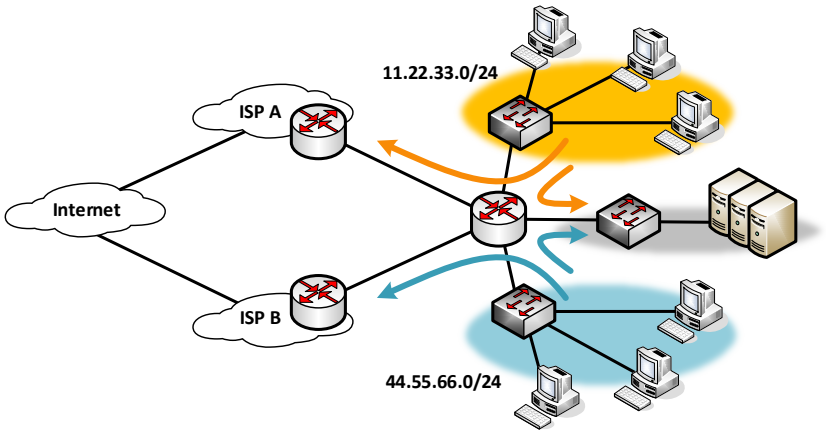
- ***Distance-Vector*** protokoli rutiranja i
- ***Link-State*** protokoli rutiranja.

Protokoli rutiranja iz ovih grupa imaju suštinski različite principe rada, koji se detaljno objašnjavaju u narednim poglavljima, ali je njihov cilj isti – uspostavljanje ruting tabele.

Za kraj razmatranja koncepta ruting protokola, ruting tabele i uobičajenog principa **rutiranja na osnovu odredišta** (*destination-based routing*), napomenimo da se u pojedinim slučajevima može primeniti i **rutiranje na osnovu izvorišta** (*source-based routing*). Tom prilikom se korišćenjem posebnih pravila prepoznaju izvorišne adrese i na osnovu toga paketi prosleđuju na odgovarajući izlazni link nezavisno od sadržaja ruting tabele.

Na primeru koji prikazuje Slika 6.11, dve IP mreže pripadaju različitim organizacijama i koriste isti ruter za izlazak na Internet, ali preko različitih provajdera. Korišćenjem klasičnog rutiranja

ne bi mogla da se napravi željena razlika u prosleđivanju odlaznog saobraćaja preko odgovarajućeg provajdera. Zato je potrebno posebnim komandama saobraćaj sa mreže 11.22.33.0/24 preusmeriti na provajdera ISP A, dok saobraćaj sa mreže 44.55.66.0/24 na provajdera ISP B. Nakon toga, na ostalim ruterima saobraćaj se normalno rutira prema odredištu. Takođe, saobraćaj između ove dve mreže može normalno da se rutira, uključujući i pristup deljenom internom segmentu sa namenskim serverima.

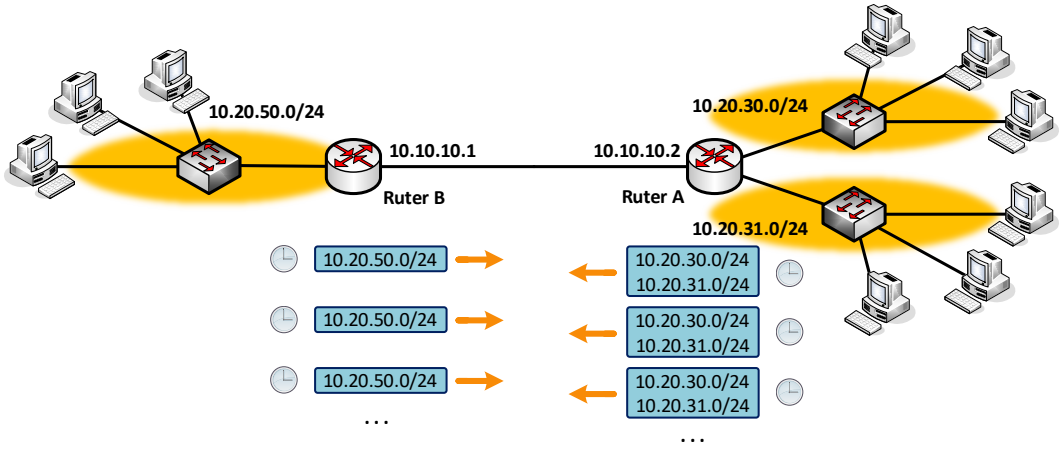


Slika 6.11. Primer rutiranja na osnovu izvorišne adrese (source-based routing)

7. *Distance-Vector* protokoli rutiranja

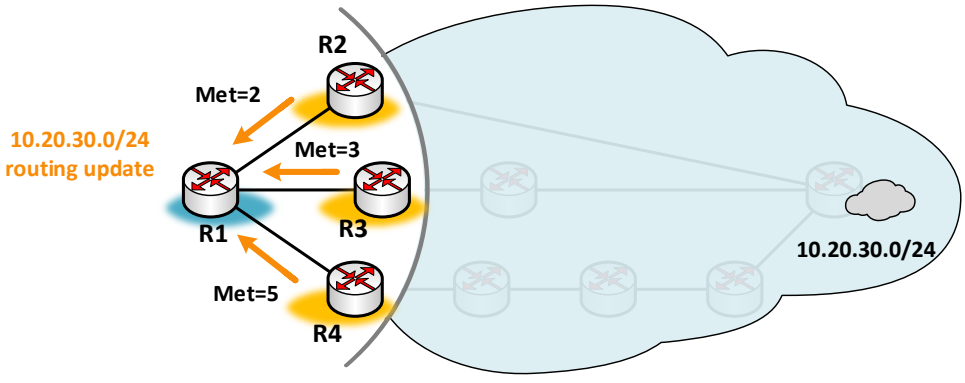
7.1 Princip rada

Protokoli rutiranja ne služe za rutiranje paketa, već za uspostavljanje tabela rutiranja na osnovu kojih se paketi rutiraju. Kod protokola rutiranja *Distance-Vector* tipa to se postiže tako što susedni ruteri između sebe razmenjuju rute koje trenutno poseduju u ruting tabelama, što se naziva oglašavanje ruta, odnosno ruting apdejt (*routing update*), kao što ilustruje Slika 7.1. Oglašavanje ruta se sprovodi periodično, uobičajeno na 30 sekundi, čak iako u ruting tabelama nema promena.



Slika 7.1. Periodično oglašavanje ruta kod distance-vector ruting protokola

Na ovaj način ruteri „uče“ na kojoj strani se nalazi koja mreža („*vector*”) i kolika je njena udaljenost („*distance*”), što je dovoljno za formiranje ruting tabele. Osim ovih informacija, za rutere je ostatak mreže nepoznat – ne poznaju se ostali ruteri u mreži, kao ni njihov broj, topologija povezanosti, odnosno tačne putanje do udaljenih mreža, broj i kapacitete pojedinih linkova itd. (Slika 7.2).



Slika 7.2. Sagledavanje mreže sa aspekta jednog rutera

U slučaju dobijanja više različitih ruta za istu IP mrežu, ruter će izabrati samo onu koja ima najbolju metriku i nju će da upiše u ruting tabelu. U slučaju da takvih ruta ima više, sve će se upisati u ruting tabelu, čime se podržava balansiranje saobraćaja (*load balancing*).

Regularna situacija na nivou cele mreže podrazumeva da su svi ruteri u stabilnom i konzistentnom stanju. Stanje rutera je stabilno ako se ruting tabela ne menja sa pristizanjem

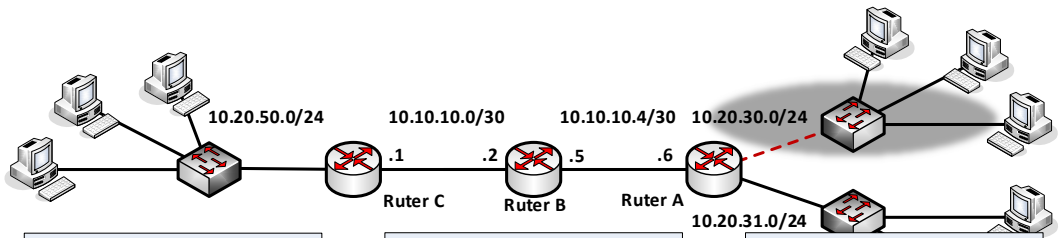
periodičnih ruting apdejta. Konzistentno stanje podrazumeva da ruter poseduje sve potrebne i tačne informacije koje obezbeđuju ispravno rutiranje prema svim mrežama u ruting domenu.

U slučaju promena u mreži, usled prekida ili dodavanja pojedinih linkova ili LAN mreža, dolazi do promene stanja – postojeće mreže mogu da postanu nedostupne ili da se promeni putanja do njih, a mogu da se pojave i nove mreže. Informacije o tome moraju da propagiraju do svih rutera u što kraćem periodu, kako bi se uspostavilo novo stacionarno stanje. Ovaj proces se naziva **konvergencija**.

Za ruting protokole je razmena ruta relativno jednostavan posao, ali je izazov obezbediti stalnu konzistentnost ovih informacija na nivou celog ruting domena. Ovo je posebno osetljivo tokom konvergencije, kada nemaju svi ruteri ažurno i ispravno stanje, što može, makar i privremeno, izazvati određene neregularne situacije. Jedna od najvećih neregularnosti je nastanak petlji pri rutiranju, što dovodi do kruženja paketa. Ruting petlje mogu da nastanu čak i na jednom linku između samo dva rutera, ako za neku mrežu ovi ruteri međusobno ukazuju jadan na drugog preko *next-hop* atributa. Paketi ipak ne kruže beskonačno, kao što je to slučaj kod Eternet petlji, budući da IP paketi imaju ograničeni „životnog veka“ kroz TTL mehanizam (*Time-to-Live*). To predstavlja krajnju meru zaštite od ruting petlji, ali nikako i rešenje problema.

Pre nego što se prikaže rešenje za eliminaciju ruting petlji koje primenjuju protokoli rutiranja *Distance-Vector* tipa, sledeći primer demonstrira nastanak petlji ako se od susednih rutera prihvataju sve rute bez dodatnih pravila i mera zaštite.

U stacionarnom stajnu, sva tri rutera koje prikazuje Slika 7.3 imaju ispravne ruting tabele za sve mreže, uz metriku koja predstavlja broj koraka (*hop-count*). Nakon toga nastaje prekid veze sa mrežom 10.20.30.0/24, što detektuje direktno povezani Ruter A i njegovu rutu, koja je bila tipa *connected*, briše iz ruting tabele. Ova informacija još uvek nije propagirala do ostalih rutera, koji i dalje imaju nepromenjene tabele rutiranja, a time i nekonzistentno stanje.



Ruter C: Ruting tabela

Mreža	Met	Next-hop
10.10.10.0/30	0	Connected
10.10.10.4/30	1	10.10.10.2
10.20.30.0/24	2	10.10.10.2
10.20.31.0/24	2	10.10.10.2
10.20.50.0/24	0	Connected

Ruter B: Ruting tabela

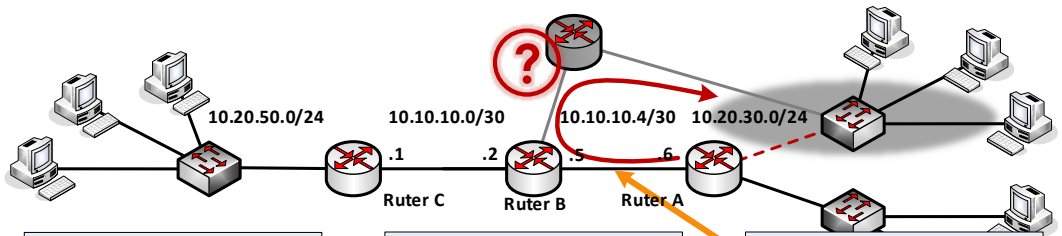
Mreža	Met	Next-hop
10.10.10.0/30	0	Connected
10.10.10.4/30	0	Connected
10.20.30.0/24	1	10.10.10.6
10.20.31.0/24	1	10.10.10.6
10.20.50.0/24	1	10.10.10.1

Ruter A: Ruting tabela

Mreža	Met	Next-hop
10.10.10.0/30	1	10.10.10.5
10.10.10.4/30	0	Connected
~~10.20.30.0/24~~	~~0~~	~~Connected~~
10.20.31.0/24	0	Connected
10.20.50.0/24	2	10.10.10.5

Slika 7.3. Nastanak ruting petlji – prekid veza Rutera A sa mrežom 10.20.30.0/24

Pre nego što Ruter A obavesti Ruter B da mreža više nije dostupna, može da nastupi periodično oglašavanje ruta od Rutera B, koji za ovu mrežu i dalje ima prethodnu vrednost metrike koja iznosi 1. Kada ova ruta dospe do Rutera A, ona će da ukazuje da Ruter B i dalje ima vezu do mreže 10.20.30.0/24 i to sa korakom 1, npr. preko nekog trećeg rutera (Slika 7.4). Stoga će Ruter A da prihvati rutu i poveća metriku na vrednost 2, a za *next-hop* vrednost postavlja adresu Rutera B, čime će ruteri međusobno da ukazuju jedan na drugog.



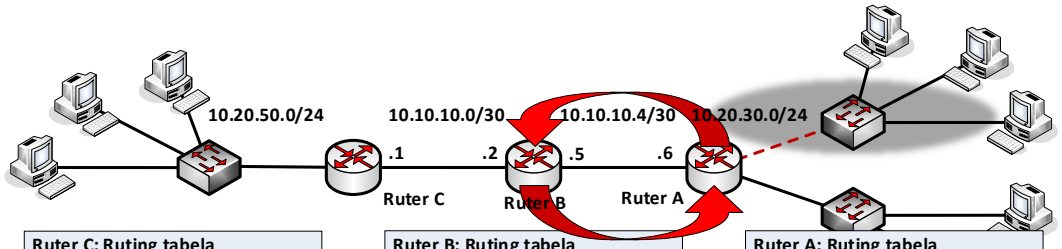
Ruter C: Ruting tabela		
Mreža	Met	Next-hop
10.10.10.0/30	0	Connected
10.10.10.4/30	1	10.10.10.2
10.20.30.0/24	2	10.10.10.2
10.20.31.0/24	2	10.10.10.2
10.20.50.0/24	0	Connected

Ruter B: Ruting tabela		
Mreža	Met	Next-hop
10.10.10.0/30	0	Connected
10.10.10.4/30	0	Connected
10.20.30.0/24	1	10.10.10.6
10.20.31.0/24	1	10.10.10.6
10.20.50.0/24	1	10.10.10.1

Ruter A: Ruting tabela		
Mreža	Met	Next-hop
10.10.10.0/30	1	10.10.10.5
10.10.10.4/30	0	Connected
10.20.30.0/24	2	10.10.10.5
10.20.31.0/24	0	Connected
10.20.50.0/24	2	10.10.10.5

Slika 7.4. Nastanak ruting petlji – Ruter B oglašava rutu koja više nije ispravna

U ovakvom stanju ako neki paket iz ostatka mreže pristigne za mrežu 10.20.30.0/24, Ruter B će da ga prosledi Ruteru A, koji ga vraća nazad Ruteru B, čime nastaje petlja pri rutiranju između Rutera A i Rutera B (Slika 7.4)



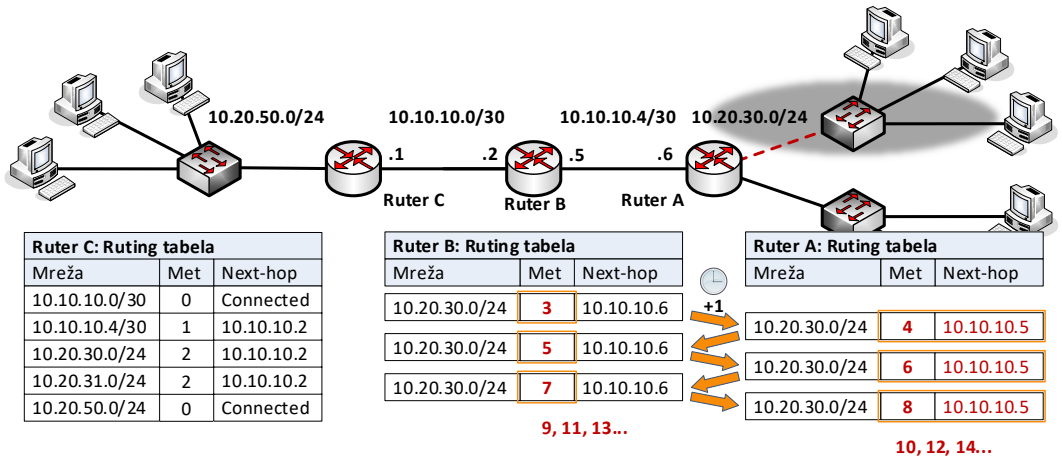
Ruter C: Ruting tabela		
Mreža	Met	Next-hop
10.10.10.0/30	0	Connected
10.10.10.4/30	1	10.10.10.2
10.20.30.0/24	2	10.10.10.2
10.20.31.0/24	2	10.10.10.2
10.20.50.0/24	0	Connected

Ruter B: Ruting tabela		
Mreža	Met	Next-hop
10.10.10.0/30	0	Connected
10.10.10.4/30	0	Connected
10.20.30.0/24	1	10.10.10.6
10.20.31.0/24	1	10.10.10.6
10.20.50.0/24	1	10.10.10.1

Ruter A: Ruting tabela		
Mreža	Met	Next-hop
10.10.10.0/30	1	10.10.10.5
10.10.10.4/30	0	Connected
10.20.30.0/24	2	10.10.10.5
10.20.31.0/24	0	Connected
10.20.50.0/24	2	10.10.10.5

Slika 7.5. Nastanak ruting petlji – kruženje paketa u petlji između Rutera A i Rutera B

Ali proces oglašavanja pogrešnih ruta koji je izazvao ruting petlu time se ne završava. U narednom ruting apdejtu, Ruter A ponovo obaveštava Ruter B da postoji veza sa tom mrežom, ali ovog puta sa metrikom koja ima vrednost 2. Ruter B prihvata rutu i u svojoj ruting tabeli ažurira vrednost metrike na 3, nakon čega na njega dolazi red za oglašavanje ruta. Time je proces oglašavanja pogrešnih ruta ušao u petlju, gde Ruter A i Ruter B naizmenično povećavaju metrike za datu rutu. Ova pojava se naziva „brojanje do beskonačnosti“ (***count to infinity***). Srećom, „beskonačnost“ u ovom slučaju predstavlja konačnu vrednost koja se tretira kao neregularna, a koja čak nije mnogo ni velika – kod RIP ruting protokola ova vrednost iznosi 16. Ipak, sa periodom oglašavanja ruta od 30 sekundi i povećavanjem vrednosti za 2, ova vrednost će da se dostigne za oko 4 minuta, tokom kog perioda će sve vreme postojati petlja pri rutiranju. Tek kada se dostigne ova najveća vrednost, koja se smatra neregularnom, ruteri će da zaključe da je mreža nedostupna i rutu će konačno da izbrišu iz ruting tabela.



Slika 7.6. Nastanak ruting petlji – count to infinity problem

7.2 Tehnike zaštite od neregularnosti

Ruting protokoli *Distance-Vector* tipa izbegavaju ovu i slične neregularne situacije primenom dodatnih tehnika koje koriste pri oglašavanju, koje se opisuju u nastavku.

- ***Route Poisoning***

  Umesto brisanja rute za mrežu koja više nije dostupna, ova tehnika podrazumeva da ruter zadrži takvu rutu u ruting tabeli, ali da joj pridruži maksimalnu i nevalidnu vrednost metrike (broj 16). Brisanjem rute gube se informacije o mreži, dok se na ovaj način dodaje informacija da mreža koja je postojala u ruting tabeli više nije dostupna. Prilikom oglašavanja ove rute sa nevalidnom metrikom, drugi ruteri će takođe da zadrže i rutu i nevalidnu vrednost metrike.

- ***Triggered Update***

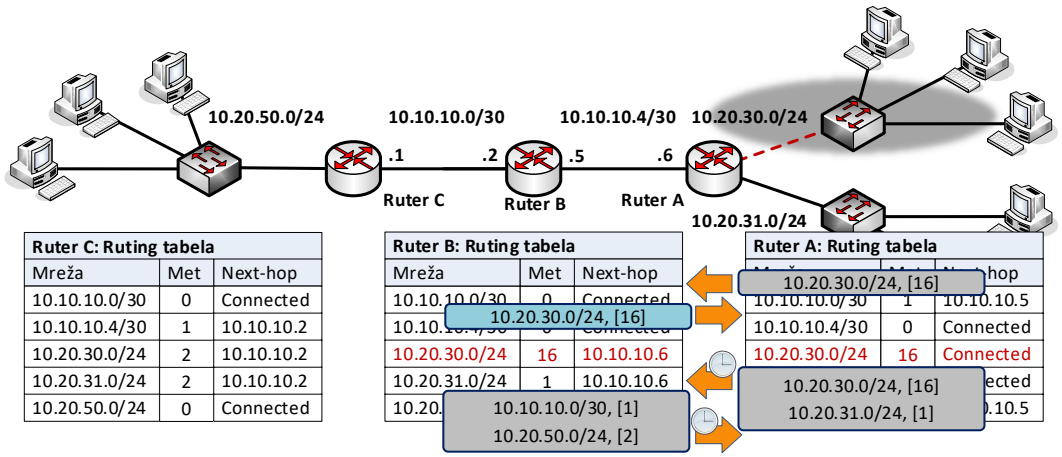
  Ova tehnika podrazumeva da kada neka ruta postane nedostupna i dodeli joj se nevalidna metrika, neće se čekati sledeći regularni periodični apdejt, već će se ova ruta istog trenutka oglasiti drugim ruterima i time označiti da je nevalidna. Pri tome se ostale rute iz ruting tabele ne oglašavaju, već čekaju sledeći period oglašavanja.

- ***Split Horizon***

  Pravilo *Split Horizon* nalaže da se nikada ne oglašava ruta na interfejs preko koga je ta ruta pristigla. Drugim rečima, za najmerodavniji izvor informacija smatra se strana sa koje su rute pristigle, tako da rute ne treba vraćati prema originalnom „izvoru“, već ih treba prenositi na ostale interfejse. Primetimo da je kršenje ovog pravila bio glavni razlog za nastanak ruting petlje u prethodnom primeru (Slika 7.4).

- ***Poison Reverse***

  Postoji izuzetak kod prethodno opisanog *Split Horizon* pravila, a koji se sprovodi samo za nevalidne rute sa maksimalnom vrednošću metrike. Kada ruter primi nevalidnu rutu, on će je ipak vratiti nazad na stranu odakle je prvobitno pristigla, što se naziva *Poison Reverse*, kršeći generalno *Split Horizon* pravilo. Ruta je i dalje nevalidna i ne može izazvati problem, ali se time dodatno postiže da ruter potvrđuje da do date mreže nema bolju rutu. Ovo oglašavanje se sprovodi neposredno po prijemu originalne rute prema *Triggered Update* principu, nakon čega se oglašavanje nastavlja u redovnom periodičnom režimu (Slika 7.7).



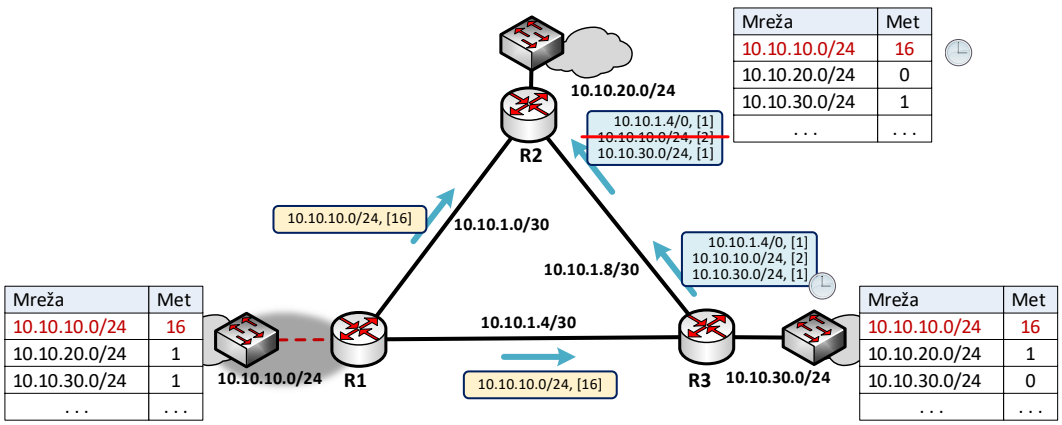
Slika 7.7. Split Horizon and Poison Reverse

- ***Holddown timer***

Proces konvergencije se sprovodi distribuirano na svim ruterima u mreži i traje određeno vreme. Čak i uz primenu prethodnih pravila koje obezbeđuju tačnost informacija i ubrzavaju propagaciju ruta, u određenim situacijama može doći do prijema ruta koje su u međuvremenu postale netačne.

U primeru koji demonstrira Slika 7.8, LAN mreža sa adresom 10.10.10.0/24 je postala nedostupna na ruteru R1, koji o tome obaveštava rutere R2 i R3 posredstvom *Triggered Update* mehanizma. Na osnovu toga ruter R2 će u ruting tabelu za ovu rutu da postavi nevalidnu vrednost metrike. Međutim, u određenom kratkom intervalu može da se desi da ruter R3 pre nego što je dobio ovu nevalidnu rutu, regularnim periodičnim oglašavanjem pošalje prethodnu, sada već zastarelu i pogrešnu informaciju da je mreža dostupna preko rutera R1. Kada ova ruta dođe do rutera R2, on će ponovo da je upiše kao validnu rutu, čime se ulazi u nekonzistentno i pogrešno stanje.

Da bi se sprečile promene stanja ruta u bliskim vremenski intervalima, što može da bude pogrešno usled zakasnelih poruka, rutama u ruting tabelama se pridružuju tajmeri, koji tokom njihovog trajanja zabranjuju ponovno menjanje stanja ruta. Na taj način se daje dovoljno vremena da se oglašavanje u mreži stabilizuje sa ispravnim informacijama. Ovaj mehanizam se naziva *Holddown timer.*



Slika 7.8. Holddown timer

7.3 RIP ruting protokol

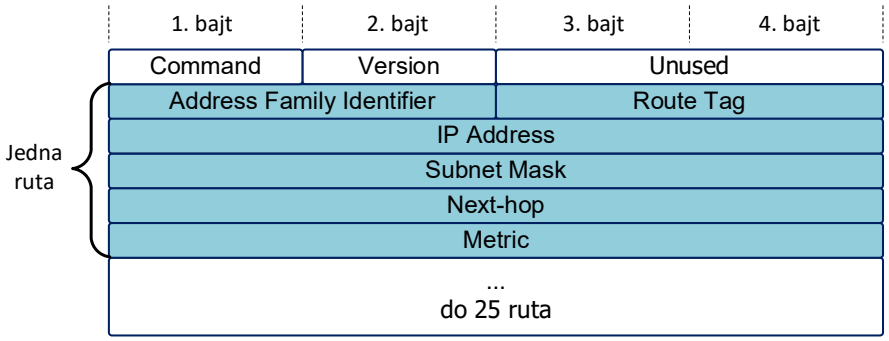
Najpoznatiji primer ruting protokola *Distance-Vector* tipa je RIP protokol (*Routing Information Protocol*), kome je dodeljena administrativna distanca od 120. RIP koristi *hop-count* metriku sa maksimalnom vrednošću 16, koja označava nevalidne rute. Prvobitna verzija 1 je pripadala *classful* tipu ruting protokola [17], da bi tek verzija 2 podržala prenos maski pri oglašavanju ruta i omogućava promenljivu dužinu maske za mreže u ruting domenu (VLSM), čime se svrstava u *classless* tip [18]. RIP protokol implementira prethodno opisane mehanizme koji obezbeđuju ispravno oglašavanje ruta: *Route Poisoning, Triggered update, Split Horizon, Holddown Timer*, uključujući i *Poison Reverse.*

Oglašavanje ruta preko RIP protokola se sprovodi periodično na 30 sekundi i to u dva koraka, putem poruka koje se prenose na aplikativnom nivou.

U prvom koraku ruter zahteva rute od susednih rutera slanjem ***RIP Request*** poruke. U verziji 1 ova poruka se šalje na brodkast adresu 255.255.255.255, dok se u verziji 2 koristi multikast adresa 224.0.0.9, za koju pakete primaju svi RIPv2 ruteri. U ovom zahtevu se može specificirati i adresa mreže za koju se traže rute, a obično se navodi adresa 0.0.0.0 koja se odnosi na sve mrežne adrese, odnosno sve rute.

U drugom koraku ruteri odgovaraju sa ***RIP Response*** porukom koja se šalje na unikast IP adresu rutera koji je inicirao zahtev. U odgovoru se šalje celokupna ruting tabela, sem ruta koje su naučene od rutera koji je poslao zahtev, a shodno *Split Horizon* pravilu. Jedna *RIP Response* poruka može da sadrži do 25 ruta, a u slučaju većeg broja ruta šalju se dodatne poruke.

Slika 7.9 prikazuje format *RIP Response* poruke. Polje *Version* ukazuje na verziju RIP protokola, polje *Command* na tip poruke koja se šalje (vrednost 1 za *RIP Request*, vrednost 2 za *RIP Response*). Osnovni podaci za jednu rutu se odnose na IP adresu mreže, masku, *next-hop* atribut i metriku. Osim toga polje *Address Family Identifier* ukazuje na vrstu adresa koje se oglašavaju, koja za IP adrese ima vrednost 2, ostavljajući mogućnost primene i za druge protokole mrežnog nivoa.



Slika 7.9. Format RIP poruke

RIP protokol je bio jedan od prvih široko rasprostranjenih protokola rutiranja. Razlog tome leži u činjenici da je jednostavan za implementaciju, konfigurisanje i održavanje, ne zahteva velike procesorske i memorijske resurse, a čak i uz stalno periodično oglašavanje, opterećuje mreže je zanemarljivo.

Sa druge strane, najveći nedostatak RIP protokola je *hop-count* metrika koja ne odražava na adekvatan način kvalitet putanja za komunikaciju prema odredištima. Osim toga, RIP ne

ispoljava dovoljnu skalabilnost u mrežama sa velikim brojem rutera (npr. preko 1000), kada ni konvergencija nije dovoljno brza, čak ni uz *Triggered Update* mehanizam. Ovo su ujedno i opšti nedostaci *Distance-Vector* protokola rutiranja, a što se prevazilazi korišćenjem *Link-State* protokola rutiranja.

8. Link-State protokoli rutiranja

8.1 Princip rada

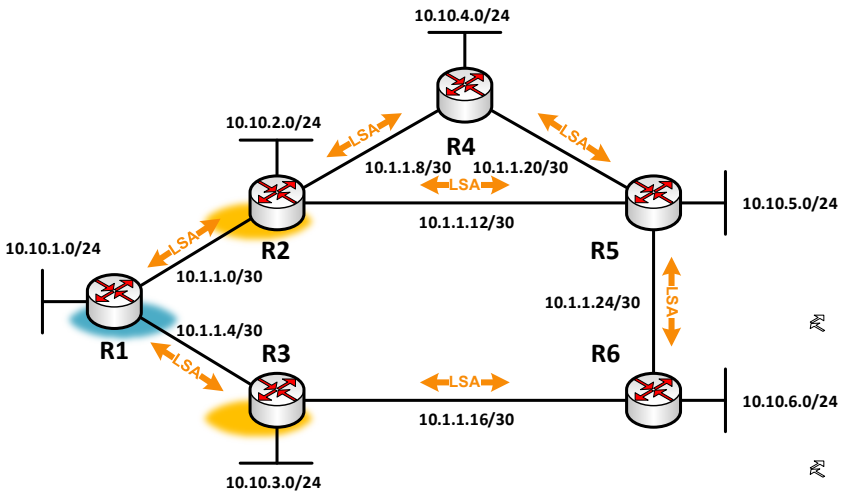
Ruteri koji koriste *Distance-Vector* protokole rutiranja učestalo razmenjuju rute sa susedima, na osnovu čega kreiraju ruting tabele, ali uz veoma limitirano sagledavanje ostatka mreže (Slika 7.2). *Link-State* protokoli rutiranja se takođe baziraju na razmeni poruka između susednih rutera, sa istim krajnjim ishodom, a to je ispravno popunjena ruting tabela, ali je njihov rad značajno drugačiji.

Najpre istaknimo da u ovoj terminologiji „*link*“ označava interfejs rutera, a „*link-state*“ označava podatke koji se odnose na interfejs, kao što su: IP adresa i maska interfejsa, tip i naziv, brzina protoka podataka, adrese susednih rutera itd.

Proces uspostavljanja ruting tabela sprovodi se u četiri koraka, kao što sledi.

- **1. Korak – Intenzivna razmena informacija**

  Ruteri između sebe ne oglašavaju rute, već razmenjuju sve relevantne informacije o svojim interfejsima (*link-state*) kroz proces koji se naziva ***Link-State Advertisement*** (LSA), a koristeći pakete koji se nazivaju **LSA paketi**. Šta više, ruteri sve ove informacije, dalje prenose do ostalih rutera u mreži, posredstvom novih LSA paketa, što predstavlja dosta intenzivan, ali kratkotrajan proces, koji se naziva ***flooding*** (Slika 8.1). Kao rezultat, svi ruteri će da poseduje iste informacije o svim ruterima i njihovim interfejsima u mreži, koje predstavljaju internu bazu informacija pod nazivom ***Link-State Database*** (LSDB). Budući da se dele iste informacije, LSDB je isti na svakom ruteru.

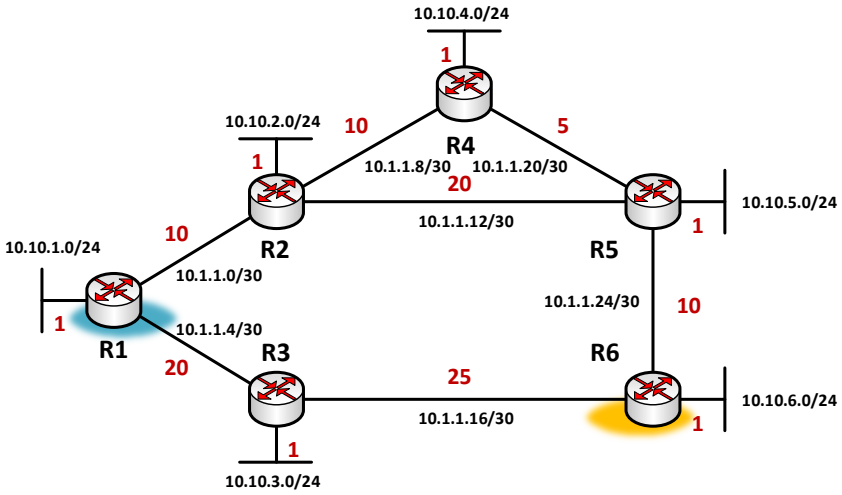


Ruter	LSA	
R1	R1 --- : 10.10.1.0/24	[0]
R1	R1 --- : 10.1.1.0/30	[0]
R1	R1 --- : 10.1.1.4/30	[0]
R2	R2 --- : 10.10.2.0/24	[1]
R2	R2 => R1: 10.1.1.0/30	[10]
R2	R2 => R4: 10.1.1.8/30	[10]
R2	R2 => R5: 10.1.1.12/30	[20]
R3	R3 --- : 10.10.3.0/24	[1]
R3	R3 => R1: 10.1.1.4/30	[20]
R3	R3 => R6: 10.1.1.16/30	[25]
R4	R4 --- : 10.10.4.0/24	[1]
R4	R4 => R2: 10.1.1.8/30	[10]
R4	R4 => R5: 10.1.1.20/30	[5]
R5	R5 --- : 10.10.5.0/24	[1]
R5	R5 => R2: 10.1.1.12/30	[20]
R5	R5 => R4: 10.1.1.20/30	[5]
R5	R5 => R6: 10.1.1.24/30	[10]
R6	R6 --- : 10.10.6.0/24	[1]
R6	R6 => R3: 10.1.1.16/30	[25]
R6	R6 => R5: 10.1.1.24/10	[10]

Slika 8.1. Intenzivna razmena LSA poruka (flooding) i kreiranje baze informacija o mreži (Link-State Database)

- **2. Korak – Kreiranje topologije**

  Na osnovu informacija iz LSDB, ruteri su u stanju da sagledaju celokupnu mrežu. Polazeći od svoje pozicije u mreži, svaki ruter rekurzivno dodaje susedne rutere i rekonstruiše izgled mreže u obliku grafa povezanosti rutera i njihovih interfejsa. Osim topologije povezanosti, ovaj logički model sadrži sve relevantne informacije, uključujući i IP adrese svih mreža, kojima je pridružena metrika u obliku cene, a koja je obrnuto proporcionalna brzini protoka na pripadajućim interfejsima (Slika 8.2).

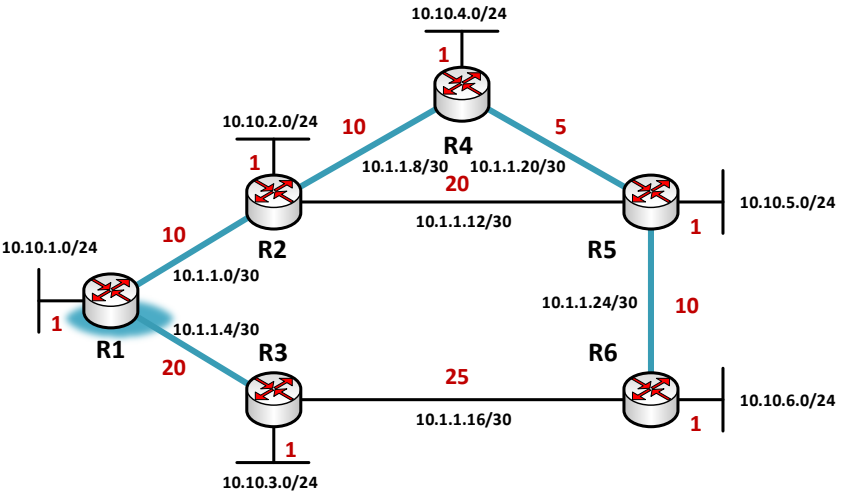


Slika 8.2. Kreiranje topologije mreže sa cenama

- **3. Korak – Nalaženje najkraćih putanja**

Svaki ruter iz svoje pozicije u grafu posmatra sve IP mreže, za koje sagledava sve moguće putanje, a kao najbolju bira putanju sa najmanjom cenom, koja predstavlja zbir cena svih pripadajućih grana (Slika 8.3). Ovo se postiže metodom nalaženja najkraćeg puta u grafu (*Shortest Path First*), imajuću u vidu cenu kao metriku, a realizuje se dobro poznatim i veoma efikasnim *Dijkstra* algoritmom [19], uz složenost O(*n* log(*n*)). I ovog puta, ako postoji više najboljih putanja koje dele najmanju cenu, sve se uzimaju kao najbolje, čime se podržava balansiranje saobraćaja.

R1: Ruting tabela		
Mreža	**Next-hop**	**Met**
10.1.1.4/30	Connected	[0]
10.1.1.0/30	Connected	[0]
10.1.1.4/30	Connected	[0]
10.10.2.0/24	10.1.1.2	[11]
10.1.1.8/30	10.1.1.2	[20]
10.1.1.12/30	10.1.1.2	[10]
10.10.3.0/24	10.1.1.6	[21]
10.1.1.16/30	10.1.1.6	[45]
10.10.4.0/24	10.1.1.2	[21]
10.1.1.20/30	10.1.1.2	[25]
10.10.5.0/24	10.1.1.2	[26]
10.1.1.24/30	10.1.1.2	[35]
10.10.6.0/24	10.1.1.2	[36]

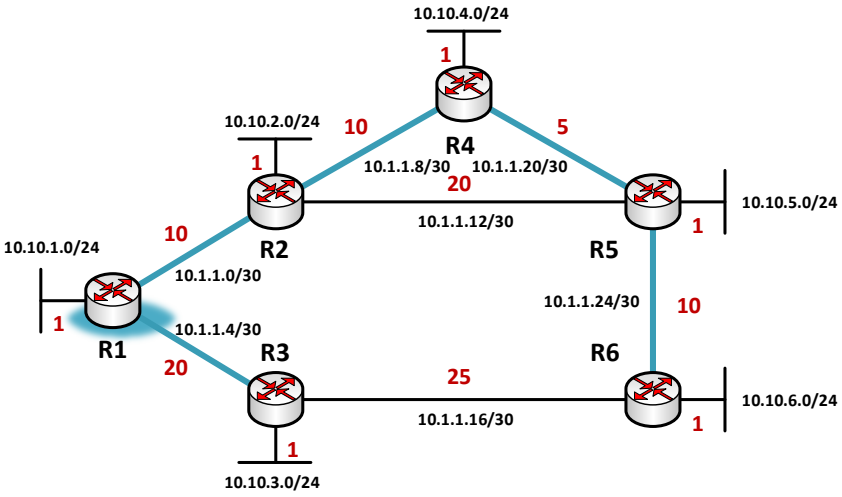


Slika 8.3. Nalaženje najkraćih putanje od rutera R1 do svih IP mreža

- **4. Korak – Kreiranje ruting tabele**

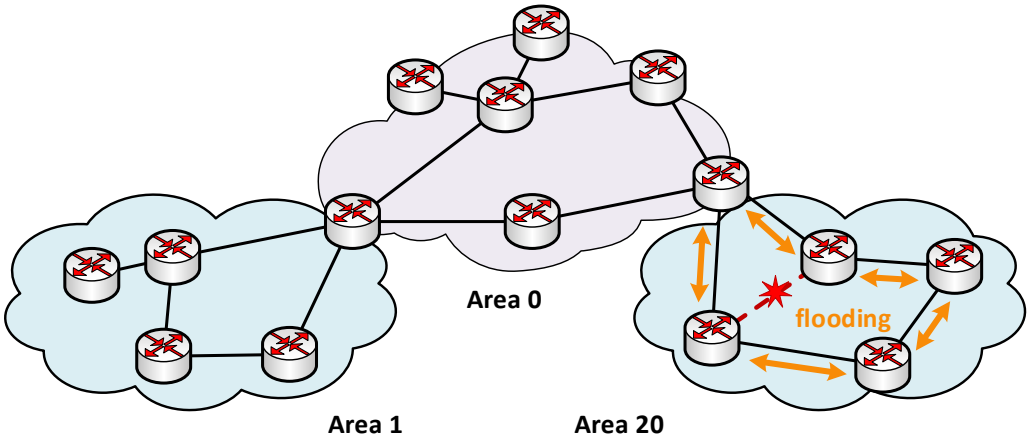
Kad ruter zna najbolje putanje do svih IP mreža, kreiranje ruting tabele je trivijalan posao, koji podrazumeva da se za svaku IP mrežu kao *next-hop* atribut uzme prvi naredni interfejs na putanji (Slika 8.4).

R1: Ruting tabela		
Mreža	Next-hop	Met
10.1.1.4/30	Connected	[0]
10.1.1.0/30	Connected	[0]
10.1.1.4/30	Connected	[0]
10.10.2.0/24	10.1.1.2	[11]
10.1.1.8/30	10.1.1.2	[20]
10.1.1.12/30	10.1.1.2	[10]
10.10.3.0/24	10.1.1.6	[21]
10.1.1.16/30	10.1.1.6	[45]
10.10.4.0/24	10.1.1.2	[21]
10.1.1.20/30	10.1.1.2	[25]
10.10.5.0/24	10.1.1.2	[26]
10.1.1.24/30	10.1.1.2	[35]
10.10.6.0/24	10.1.1.2	[36]



Slika 8.4. Kreiranje ruting tabele na ruteru R1

Navedeni postupak konvergencije traje kratko, ali je dosta intenzivan u pogledu komunikacije između rutera (*flooding* proces) i zauzeća memorijskih i procesorskih resursa. On se sprovodi pri svakoj promeni u mreži, što utiče na sve rutere u mreži. Da bi se ograničio uticaj pojedinačnih promena na ostatak mreže, što je posebno izraženo u slučaju nestabilnih veza, ruting domen se može podeliti na više celina, koje se nazivaju oblasti (***area***). Promena stanja u jednoj oblasti će izazvati *flooding* samo u toj oblasti, bez prenosa konvergencije u naredne oblasti, koje ostaju stabilne (Slika 8.51



Slika 8.5. Povećanje skalabilnosti podelom ruting domena na oblasti

Iako je konvergencija dosta buran proces, ona je ipak brza i generalno traje kraće nego što je to slučaj sa *Distance-Vector* ruting protokolima. Takođe, u stacionarnom stanju *Link-State* ruting protokoli su dosta mirni, a ruteri periodično razmenjuju samo kratke poruke za očuvanje susedstava (*keepalive*). Cena ovih dobrih osobina *Link-State* ruting protokola je veće zauzeće memorijskih i procesorskih resursa rutera, kao i veće opterećenje mreže tokom konvergencije.

Dve vrste *Link-State* ruting protokola su široko prihvaćene:

- **IS-IS** (***Intermediate System-Intermediate System***), standardizovan od strane ISO organizacije [20]
- **OSPF** (***Open Shortest Path First***), standardizovan od strane IETF organizacije [21].

Nastavak poglavlja detaljnije opisuje OSPF protokol.

8.2 OSPF ruting protokol

Inicijalna verzija OSPF ruting protokola datira iz 1991. godine, dok je aktuelna verzija 2 standardizovana 1998. godine [21].

OSPF poruke se prenose direktno unutar IP paketa, a za identifikaciju u polju *Protocol* u IP zaglavlju rezervisan je broj 89. Paketi se razmenjuju samo između susednih rutera, koristeći dve multikast adrese: 224.0.0.5 (tzv. *AllSPFRouters*) i 224.0.0.6 (tzv. *AllDRouters*), a čije su namene prikazane u nastavku.

Pri komunikaciji unutar OSPF protokola, ruteri moraju da imaju jedinstvene identifikatore na nivou celog ruting domena (***Router ID*** – RID). Podrazumevana vrednost identifikacije rutera je najveća IP adresa logičkog interfejsa (*loopback*), ako se koristi, odnosno fizičkog interfejsa u suprotnom slučaju.

Loopback predstavlja logičku instancu interfejsa, koja se definiše u konfiguraciji rutera i ima IP adresu i masku koje mogu da služe za komunikaciju sa ruterom. U odnosu na ostale fizičke interfejse, bitno je istaći sledeće dve karakteristike *loopback* interfejsa:

- *Loopback* interfejs je uvek operativan (u „UP“ stanju), za razliku od fizičkog interfejsa koji može da bude i neoperativan (u „DOWN“ stanju), bilo da je administrativno ugašen pri konfiguraciji rutera ili da ne ispunjava određene fizičke uslove (npr. kabl nije povezan).
- Kao logički interfejs, bez stvarne mreže povezane na njega, *loopback* interfejs može da egzistira i singularno („sam za sebe“), pa se za njega dodatno dozvoljava i maska dužine 32 bita, što se često koristi radi uštede adresa.

Zbog ovih osobina, *Loopback* interfejsi imaju prednost pri izboru OSPF identifikacije rutera, a njihovo korišćenje za ovu namenu predstavlja dobru praksu.

Komunikacija između OSPF rutera se sprovodi razmenom različitih vrsta OSPF poruka, a krajnji cilj je da se upotpune (sinhronizuju) LSDB tabele (*Link-State Database*) razmenom nedostajućih LSA paketa, što se sprovodi u više sukcesivnih faza.

- ***Hello* – Prepoznavanje susedstva**

  Nije dovoljno da ruteri budu samo fizički povezani, već je potrebno i da se oni međusobno prepoznaju kao OSPF ruteri, razmene i usaglase sve obavezne parametre (RID, identifikaciju oblasti itd.).

  Tom prilikom ruteri prolaze kroz sledeća tri stanja.

  - ***Down*** – početno stanje.
  - ***Init*** – među-stanje inicijalizovane, ali ne i završene komunikacije.
  - ***2*-way** – stanje u koje ulazi ruter nakon što je dobio potvrdu da ga je druga strana prepoznala.

  Ovaj proces inicijalnog prepoznavanja sprovodi se u tri koraka (Slika 8.6):

  - **1. korak**: Privi ruter šalje tzv. *Hello* poruku na multikast adresu 224.0.0.5, u kojoj označava svoju identifikaciju, nakon čega ulazi u *Init* stanje.

- **2. korak**: Drugi ruter odgovara sa novom *Hello* porukom, u kojoj u posebnom polju, pod nazivom *Seen,* upisuje prethodno dobijen RID, nakon čega takođe prelazi u *Init* stanje. Kada prvi ruter primi *Hello* poruku i u *Seen* polju prepozna svoju identifikaciju, to je potvrda da ga je drugi ruter prepoznao, nakon čega ulazi u *2-way* stanje.
- **3. korak**: Prvi ruter šalje novu *Hello* poruku, a u polje *Seen* dodaje RID drugog rutera. Po prijemu ove poruke, drugi ruter prepoznaje svoju identifikaciju u polju *Seen*, i takođe ulazi u *2-way* stanje.

2-way stanje na oba rutera označava da su se ruteri međusobno prepoznali kao susedni OSPF ruteri, ali tek treba da pristupe razmeni LSA podataka, što se sprovodi kroz naredne korake. U slučaju nastanka greške kada se razmena LSA ne može ostvariti, ruteri ostaju u *2-way* stanju.

Hello poruke nastavljaju da se periodično šalju čak i ako se ruteri uspešno prepoznaju, što služi za održavanje uspostavljenog 2-way stanja. Predefinisana vrednost periodičnog oglašavanja *Hello* poruka iznosi 10 sekundi, tzv. ***Hello Interval***. U slučaju izostanka određenog broja uzastopnih *Hello* poruka, smatraće se da je susedni ruter postao neaktivan i raskinuće se *2-way* stanje. Predefinisani broj izostalih *Hello* poruka poruka nakon kojih se raskida *2-way* stanje je 4. Imajući u vidu *Hello Interval* od 10 sekundi, predefinisana vrednost za interval detekcije neaktivnog rutera iznosi 40 sekundi, što se naziva *Dead Interval*. Predefinisane vrednosti *Hello* i *Dead* intervala se mogu menjati, ali to nije preporučljivo, budući da ovi parametri moraju da budu isti na svim ruterima kako bi se uspostavilo susedstvo, što je osnovni preduslov da OSPF ispravno radi.

♦ ***ExStart* – Priprema za razmenu**

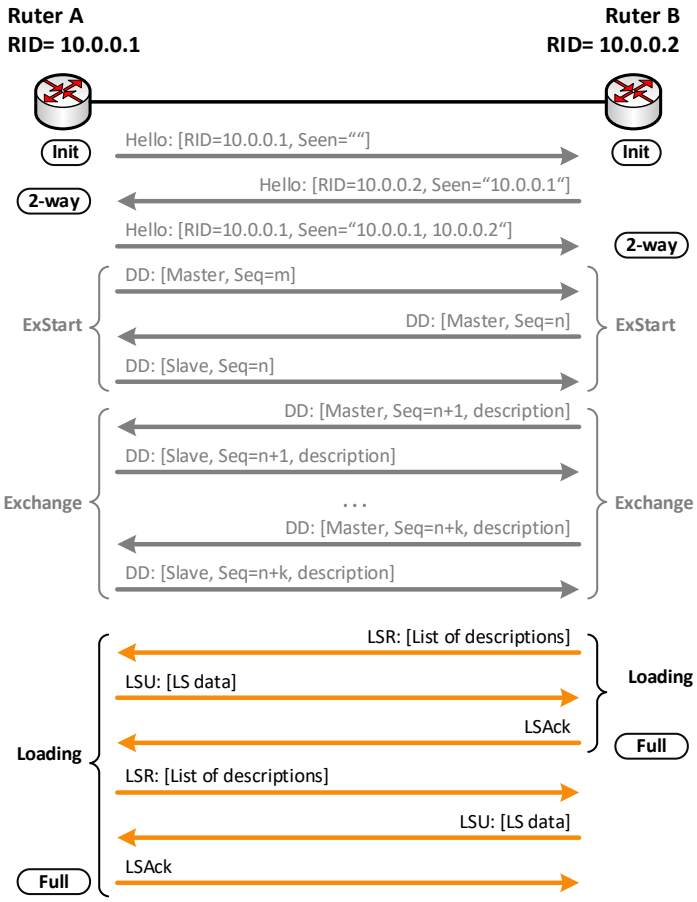
U ovoj fazi ruteri se dogovaraju ko će prvi da šalje podatke, tzv. ***master***, a ko šalje nakon toga, tzv. ***slave***. Svaki ruter može sebe da delegira da bude *master*, što drugi ruter može da prihvati ili odbije. Pravilo je da *maser* postaje ruter sa većom identifikacijom. Pri razmeni ovih poruka, inicijalizuju se početne vrednosti tzv. brojača sekvence (*Sequence Number*), koji služi za praćenje razmene paketa, što sledi u narednoj fazi.

♦ ***Exchange* – Razmena deskriptora**

Da bi se LSDB tabele sinhronizovale na optimalan način, potrebno je razmeniti samo nedostajuće podatke, a da bi se to sprovelo, najpre je potrebno da se utvrdi šta nedostaje na svakoj strani. U tom cilju ruteri najpre razmenjuju kratke opise sadržaja LSDB tabela, tzv. deskriptore (***Database Descriptor***), što prvo sprovodi *master*, a nakon toga i *slave* ruter.

♦ ***Loading* – Razmena svih podataka**

U ovoj završnoj fazi se konačno prenose nedostajući sadržaji iz LSDB tabela, a koji su utvrđeni u prethodnoj fazi. Najpre *master* šalje poruku ***Link-State Request*** (LSR), u kojoj navodi listu deskriptora za nedostajuće podatke. *Slave* ruter šalje zahtevane podatke u jednoj ili više ***Link-State Update*** poruka (LSU), što master potvrđuje slanjem ***Link-State Acknowledgement*** poruke (LSAck). *Master* ruter tada ulazi u stanje ***full***, koje podrazumeva da je LSDB tabela na ruteru u potpunosti ažurirana. Nakon toga na isti način i *slave* zahteva nedostajuće podatke, a po njihovom prijemu od *master* rutera, i *slave* ruter ulazi u ***full*** stanje. Time je proces konvergencije okončan, a ruteri koji su međusobno prošli ovu fazu smatraju se za „sinhronizovane OSPF susede“ (***adjacent***).

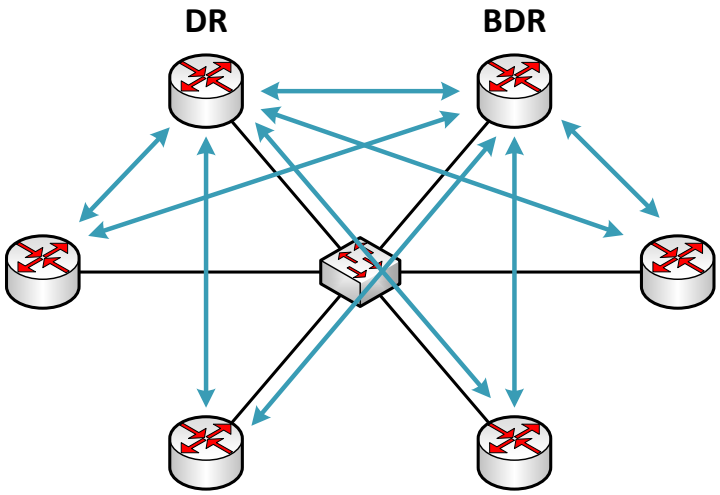


Slika 8.6. Razmena podataka i uspostavljanje adjacent odnosa između dva rutera

Opisani proces razmena LSA paketa i uspostavljanja *adjacent* odnosa između dva rutera je očigledan u slučaju direktnih veza (*point-to-point*). Ipak, stvari se komplikuju u slučaju segmenata sa više učesnika, tzv. brodkast mreža, kakva je LAN mreža Enternet tipa. Uspostavljanje *adjacent* odnosa između svih parova rutera bi bilo neskalabilno, jer njihov broj raste sa kvadratnom zavisnošću u odnosu na broj rutera (za n rutera postoji $n(n-1)/2$ parova).

Da bi se izbegla direktna komunikacija „svako-sa-svakim“, a time smanjilo opterećenje i rutera i mreže, u mreži Eternet tipa se bira jedan centralni ruter, tzv. ***Designated Router*** (**DR**), sa kojim svi ostali ruteri uspostavljaju *adjacent* odnos. Kako ovaj ruter ne bi bio jedinstvena tačka otkaza (*single point of failure*), bira se i rezervni centralni ruter, tzv. ***Backup Designated Router*** (BDR), sa kojim ostali ruteri takođe uspostavljaju *adjacent* odnos. U slučaju prekida rada DR, njegov ulogu automatski preuzima BDR, nakon čega se traži novi BDR. Broj veza je time sveden na linearnu zavisnost u odnosu na broj rutera (Slika 8.7). Ostali ruteri u OSPF terminologiji se nazivaju ***DR Other***.

Podrazumevano pravilo je da DR i BDR postaju ruteri sa najvećim identifikatorima (RID). Ovaj izbor se može kontrolisati konfiguracijom prioriteta u vrednosti od 0 do 255, gde veća vrednost ima veći prioritet pri izboru. Vrednost prioriteta od 0 isključuje ruter iz postupka izbora DR i BDR. Ruteri koji postanu DR ili BDR, zadržavaju to stanje čak iako se u mreži pojavi ruter koji ima veći prioritet, odnosno identifikaciju. Ovo pravilo je uspostavljeno kako bi se izbegla ponovna konvergencija uz intenzivni *flooding* proces.



Slika 8.7. Uspostavljanje adjacent veza u Eternet mreži sa više rutera

Razmena LSA paketa i sinhronizacija LSDB tabela između svih rutera u Eternet mreži se sprovodi posredno preko DR, i to u dva koraka:

- Svi ruteri šalju LSA pakete na multicast adresu 224.0.0.6, koja se simbolički označava sa *AllDRouters*, jer na njoj saobraćaj pirmaju samo DR i BDR ruteri.
- DR i BDR ruteri primaju ove LSA pakete, ali samo DR ih dalje prosleđuje na multicast adresu 224.0.0.5, koja se simbolički označava sa *AllSPFRouters*, jer na njoj slušaju svi OSPF ruteri. Na taj način svi poslati LSA paketi indirektno preko DR pristižu do svih ostalih rutera.

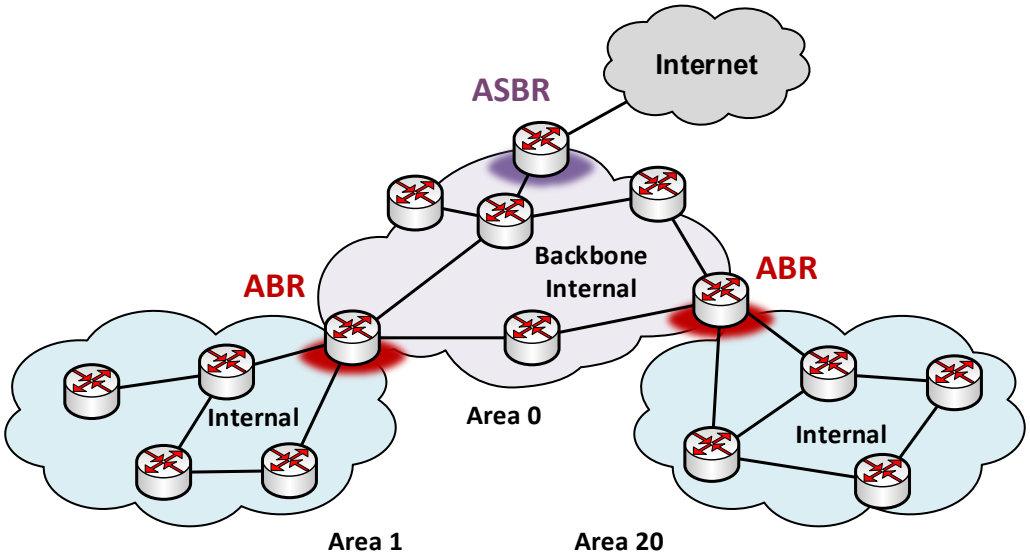
OSPF protokol za metriku koristi cenu, a koja se obrnuto proporcionalno izvodi iz brzine prenosa podataka na interfejsu. Referentna vrednost od koje se deli konkretna brzina interfejsa je ostavljena na izbor proizvođačima rutera. Proizvođač *Cisco Systems* za referentnu vrednost uzima 100 Mbps, tako da *FastEthernet* i brže tehnologije imaju najmanju cenu od 1. Osim ovakvog implicitnog uspostavljanja cene, na svakom interfejsu je moguće i ručno konfigurisati proizvoljnu vrednost cene i time dodatno podešavati prioritete pojedinih linkova u izboru najbolje putanje.

Kao što je prethodno istaknuto, skalabilnost *Link-State* ruting protokola se postiže podelom ruting domena na oblasti (Slika 8.5), i to na jednu centralnu oblast (***backbone area***), na koju se povezuje proizvoljan broj perifernih oblati (***peripheral area***). Dodatno ograničenje je da se periferne oblasti mogu direktno povezivati samo na centralnu, ali ne i na druge periferne oblasti. Oblasti se označavaju celobrojnim vrednostima, gde centralna oblasti ima vrednost 0, a periferne oblasti proizvoljne vrednosti veće od nule (npr. *Area 0, Area 20, Area 100*). Osim toga, dozvoljeno je da oznake oblasti budu u *dotted-decimal* formi kao i IP adrese, u kom slučaju centralna oblast ima oznaku *Area 0.0.0.0.* Ovo ipak nije formalno dodeljivanje IP adrese mreže, već samo simbolički naziv, koji je koristan ako cela oblast deli jedinstvenu agregiranu IP adresu.

Periferne oblasti se na centralnu oblast povezuju preko jednog ili više rutera. Ruteri mogu da pripadaju jednoj ili više oblasti, ali interfejsi rutera, linkovi i LAN mreže mogu da pripadaju samo jednoj oblasti. Na osnovu toga, prema mestu i ulozi rutera u oblasti, ruteri imaju sledeće uloge (Slika 8.8):

- ***Area Border Router*** *(*ABR) – granični ruter između centralne i periferne oblasti.

- ***Autonomous System Boundary Router*** (ASBR) – granični ruter između OSPF domena i nekog drugog ruting domena.
- ***Internal Router*** – interni ruter koji pripada samo jednoj oblasti.
- ***Backbone Router*** – interni ruter koji pripada centralnoj oblasti.



Slika 8.8. Tipovi rutera prema mestu i ulozi u oblasti

Da bi se minimizirao *flooding* proces između oblasti, potrebno je različito tretirati pojedine LSA zapise, pa se razlikuju sledeće vrste LSA paketa prema načinu oglašavanja unutar oblasti i prenošenju između oblasti (Slika 8.9):

- ***Router LSA*** (*Type 1*)

  LSA koje generišu svi ruteri za oglašavanje informacija o pripadajućim interfejsima, a propagiraju samo unutar jedne oblasti, bez prenosa se u druge oblasti, tzv. ***Intra-Area LSA***. U ruting tabeli označene su sa "O".

- ***Network LSA*** (*Type 2*)

  LSA koje generiše samo DR ruter, čime se oglašava pripadajuća Eternet mreža prema ostalim ruterima u oblasti. Takođe propagira samo unutar jedne oblasti, bez prenosa u druge oblasti, tzv. *Intra-Area LSA*. U ruting tabeli označene su sa "O".

- ***Summary LSA*** (*Type 3* i *Type 4*)

  - *Type 3* – LSA u koje se pretvaraju *Router LSA* i *Network LSA* na ABR ruteru. Na ovaj način se Informacije o lokalnim linkovima i mrežama iz jedne oblasti preko ABR rutera prenose najpre u Area 0, a zatim preko drugih ABR unose se u druge oblasti, tzv. ***Inter-Area LSA***.
  - *Type 4* – LSA koje oglašava ASBR ruter za svoje interfejse, čime se u OSPF ruting domen prenose informacije kako pristupiti ASBR ruteru za vezu sa drugim ruting domenima.

  Konverzijom *Router LSA* i *Network LSA* u *Summary LSA* postiže se smanjenje *flooding* efekata, jer promene u jednoj oblasti ne izazivaju konvergenciju u drugim oblastima.

Dodatno je poželjno dizajnirati adresni prostor tako da je moguće agregirati sve IP mreže iz jedne oblasti i time koristiti samo jednu *Summary LSA* za sve mreže iz oblasti.

U ruting tabeli označene su sa “O IA”

- ***External LSA*** (*Type 5*)

  LSA koje generiše ASBR ruter i ubacuje u OSPF domen za prenos informacija o mrežama van OSPF domena. Tom prilikom na dva načina se mogu tretirati cene.

  - Na metriku iz drugog ruting domena dodaje se cena u OSPF domenu (kumulativna cena) U ruting tabeli označene su sa “O E1”.
  - Zadržava se metrika iz drugog ruting domena i nepromenjena vrednost se prenosi u svim oblastima bez dodavanja OSPF cena. U ruting tabeli označene su sa “O E2”.

U svakoj oblasti se interno prenose *Router LSA* i *Network LSA* generisani od strane pripadajućih „unutrašnjih“ rutera, koje se pri izlasku iz oblasti agregiraju i konvertuju u *Summary LSA*. Sa druge strane, tipovi LSA kojima se dozvoljava ulazak u oblast određuju koje će rute biti generisane u ruting tabelama, na osnovu čega se oblasti dele na sledeće vrste:

- ***Standard Area*** (*Ordinary*)

  Ovo je podrazumevana vrsta oblasti, u koju se dozvoljava ulazak svih LSA koje su za to namenjene, kao što su *Summary LSA i External LSA.* Centralna oblast je uvek ovog tipa, budući da mora da preuzme i prenese sve LSA pakete.

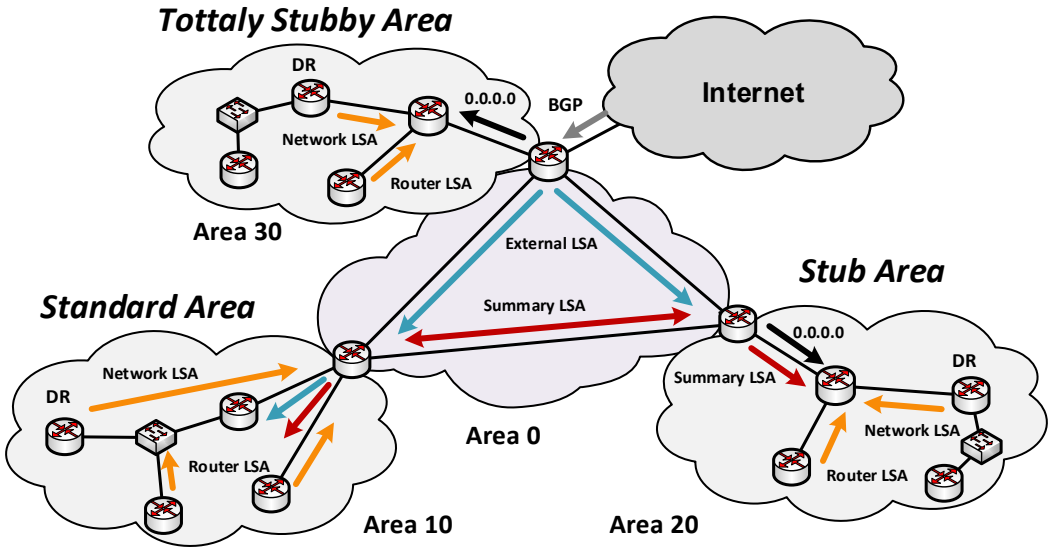
- ***Stub Area***

  Ova periferna oblast dozvoljava ulazak samo *Summary LSA* paketa, ali ne i *External LSA* paketa. Dodatno, kako bi se obezbedilo slanje IP paketa prema odredištima van OSPF domena, ABR ruter, na koji je ova periferna oblast povezana, automatski ubacuje difoltnu rutu u oblast.

  Pripadnost *stub* oblasti je potrebno naglasiti u konfiguraciji OSPF protokola na svakom ruteru u oblasti, tako što se pri definiciji oblasti, uz dodeljenu identifikaciju (broj oblasti) naznači i tip „*stub*“. Najme, prilikom razmene *Hello* poruka, jedan od podataka koji moraju biti usaglašeni i isti na svim ruterima u ovoj vrsti oblasti je i polje „*stub flag”*, što u ovom slučaju mora biti setovano (vrednost *true*). U slučaju da neki ruter oglašava *Hello* pakete sa resetovanim ovim flegom, OSPF susedstvo se neće uspostaviti, a time ni razmeniti LSA paketi.

- ***Totally Stubby Area***

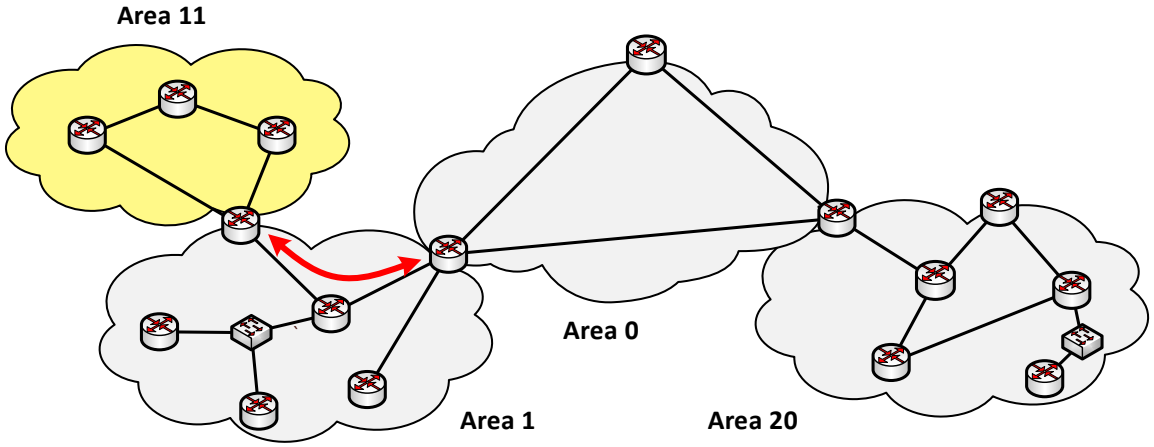
  Ova periferna oblast ne dozvoljava ulazak ni *External LSA* ni *Summary LSA* paketa. Umesto toga, kao i kod *stub* oblasti, ABR ruter automatski ubacuje difoltnu rutu u oblast ovog tipa. Ovo se obično postavlja za oblasti koje su samo preko jednog ABR rutera povezane sa centralnom oblasti i ostatkom mreže, pa je difoltna ruta dovoljna da zameni rute ka svim mrežama van oblasti. Time se znatno smanjuje veličina ruting tabele, a uz zadržavanje potpuno funkcionalnog rutiranja. I ovom prilikom na svim ruterima tip oblasti mora biti definisana kao *stub area,* kako bi se postavio *stub* fleg u *Hello* porukama između svih rutera u oblasti, dok se samo na ABR ruteru oblast proglašava za tip *„Totally Stubby“.*



Slika 8.9. Tipovi LSA prema načinu prenosa u oblastima

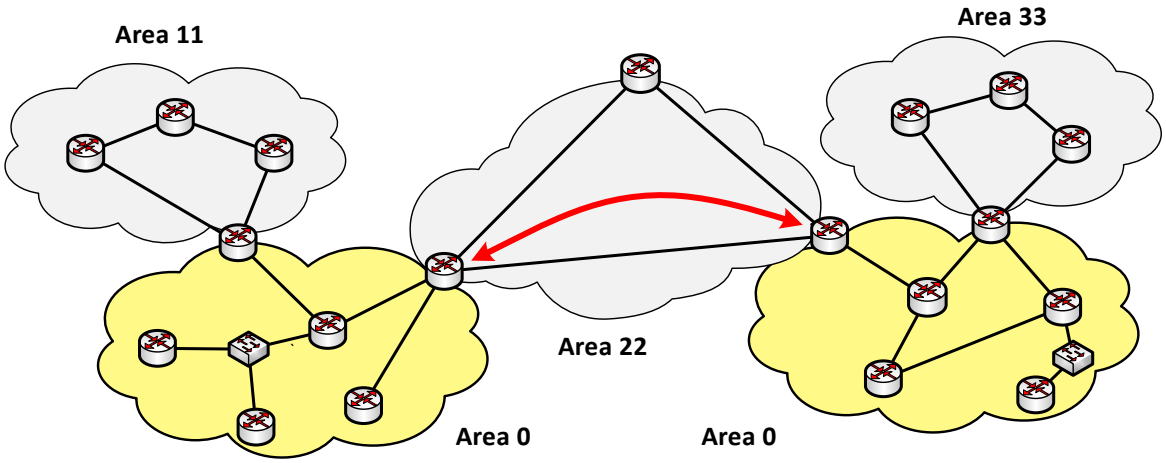
Oblasti u OSPF ruting domenu obično se dele po fizičkoj izdvojenosti i lokalnoj koncentrisanosti. Prethodno je naznačeno da sve periferne oblasti moraju da budu direktno povezane na centralnu oblast, čime se onemogućava da se jedna periferna oblast poveže na drugu. Realnost ponekad nameće da deo mreže, koji bi bilo prirodno izdvojiti u posebnu oblast, ne može fizički direktno da se poveže na centralnu oblast, već samo na obližnju, takođe perifernu oblast.

Kako bi se izbeglo „veštačko“ spajanje ovakve dve susedne periferne oblasti u jednu, uvodi se koncept **virtualnog linka** (***virtual link***), koji omogućava logičko (virtualno) i potpuno funkcionalno povezivanje periferne oblasti na centralnu oblast, ali fizički preko druge periferne oblasti (Slika 8.10). Na ovaj način jedna periferna oblast se povezuje na drugu, kroz koju se uspostavlja virtuelni link do ABR rutera i centralne oblasti. Zadatak rutera koji uspostavljaju virtuelni link je da u svemu simuliraju razmenu odgovarajućih LSA paketa i drugih funkcionalnih elemenata OSPF protokola kao da se radi o jednom ABR ruteru.



Slika 8.10. Virtuelni linkovi – logičko povezivanje periferne oblasti na centralnu posredstvom druge periferne oblasti

Budući da se na ovaj način obezbeđuje logička veza između dva rutera, moguće je čak preko virtuelnog linka spojiti dva fizički razdvojena dela centralne oblasti u funkcionalno jedinstvenu logičku centralnu oblast (Slika 8.11). Primer za ovo egzotično spajanje je integracija dva odvojena OSPF ruting domena u jedan.

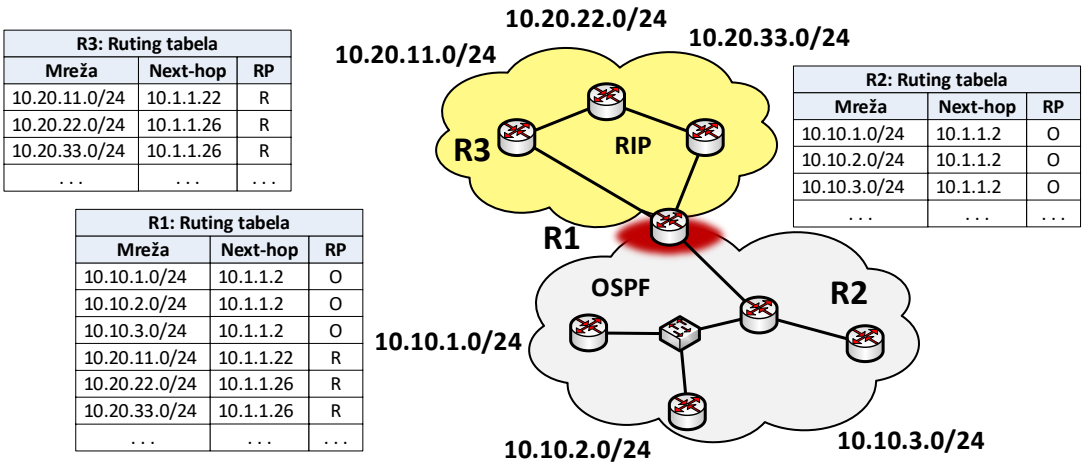


Slika 8.11. Virtuelni linkovi – logičko spajanje dva dela centralne oblasti posredstvom fizičke periferne oblasti

8.3 Redistribucija ruta između ruting domena

Rute za direktno povezane mreže (*connected* rute) automatski se integrišu u OSPF domen, obuhvatanjem interfejsa rutera u odgovarajuću OSPF oblast. One se oglašavaju kao *Router LSA* i *Network LSA*, a u drugim oblastima se prenose kao *Summary LSA*. Sve ostale mreže van OSPF domena su inicijalno nepoznate za OSPF protokol, čak iako se njihove rute nalaze u ruring tabeli nekog rutera unutar OSPF ruting domena. Ovo se odnosi na rute iz drugih ruting protokola, statičke rute ili čak rute iz drugog nezavisnog OSPF ruting domena. Da bi ove spoljašnje rute postale vidljive u posmatranom OSPF ruting domenu, one moraju da se eksplicitno prihvate, što se postiže kroz proces koji se naziva **redistribucija ruta**.

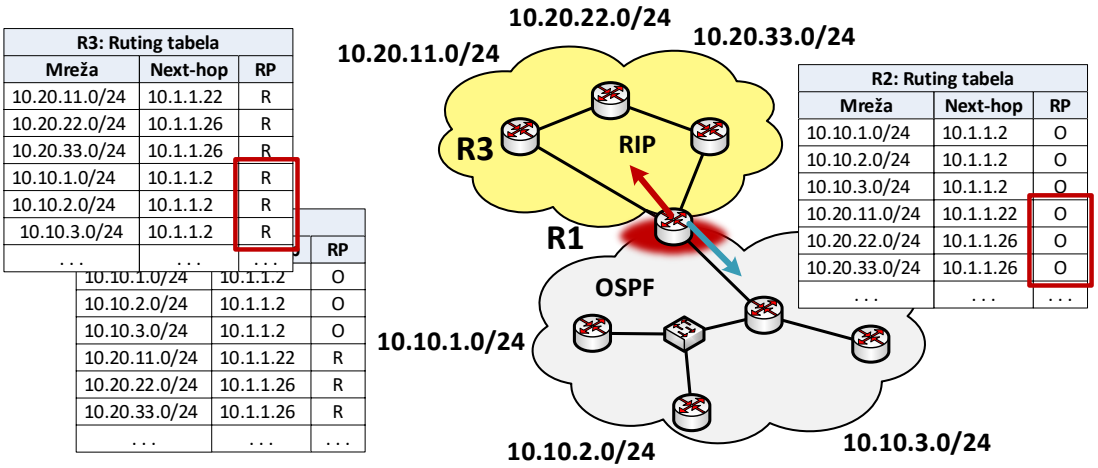
Prethodni princip tretiranja spoljašnjih ruta generalno važi za sve ruting protokole. Na primeru koji pokazuje slika Slika 8.12, RIP i OSPF ruting domeni su fizički spojeni preko rutera R1, ali na kome nije sprovedena redistribucija ruta. Posledica toga je da ruter R1 poseduje rute iz oba ruting protokola i može da komunicira sa svim IP mrežama, ali ostali ruteri poseduju rute samo iz ruting protokola kome u potpunosti pripadaju. Zbog toga je komunikacija između ruting domena praktično onemogućena.



Slika 8.12. Povezivanje različitih ruting domena - stanje bez redistribucije ruta

Da bi se omogućila potpuna komunikacija na nivou cele fizičke mreže, potrebno je na ruteru R1 konfigurisati redistribuciju ruta, o to u oba smera: iz RIP u OSPF, kao i iz OSPF u RIP ruting protokol (Slika 8.13). RIP rute iz ruting tabele na ruteru R1 se tada prihvataju u OSPF domen i pretvaraju u *External LSA* poruke, koje prenose informacije o mrežama iz RIP ruting domena. One će na ostalim ruterima u OSPF domenu da generišu OSPF rute eksternog tipa (E1 ili E2).

Na sličan način, redistribucijom OSPF ruta u RIP domen, ruter R1 počinje da oglašava IP mreže iz OSPF domena, ali u potpunosti preko RIP protokola. Ostali ruteri iz RIP ruting domena će da prihvate ove informacije kao regularne RIP rute, budući da RIP ne prepoznaje razliku između originalnih ruta nastalih unutar RIP protokola i ruta za mreže preuzetih iz drugog ruting protokola.



Slika 8.13. Povezivanje različitih ruting domena - stanje nakon međusobne redistribucije ruta

Prilikom redistribucije ruta, korisno je da se IP adrese mreža agregiraju, ako je to izvodljivo. Time se smanjuje broj ruta koje se unose u drugi ruting domen, rasterećuju ruteri, a ruting tabele čine preglednijim prilikom administracije mreže.

Napomenimo i to da nema uvek potrebe za uzajamnom redistribucijom. Tipično je da se rute iz manje mreže redistribuiraju u ruting domen veće mreže, dok se u obrnutom smeru u manjoj mreži može definisati difoltna statička ruta koja ukazuje na „izlazni“ put preko veće mreže. Ova

statička ruta takođe mora da se redistribuira, što neki ruting protokoli sprovode automatski kroz određene konfiguracione komande. Sličan princip važi i za povezivanje mreže na Internet, koji sadrži mnogo više ruta (preko milion), što prevazilazi funkcionalnost internih protokola rutiranja i performanse rutera koji se za ovu namenu koriste. Granični ruter može da koristi eksterni protokol rutiranja (BGP) i da poseduje čak sve Internet rute, ali njih nipošto ne treba učitavati u interni protokol rutiranja. Umesto toga, dovoljno je da granični ruter generiše difolntu rutu koja se propagira do svih rutera unutra ruting domena. Time će ruteri u internom ruting domenu da nauče gde je izlaz za saobraćaj prema svim spoljašnjim mrežama.

9. ARP protokol

9.1 Uparivanje IP adresa i MAC adresa

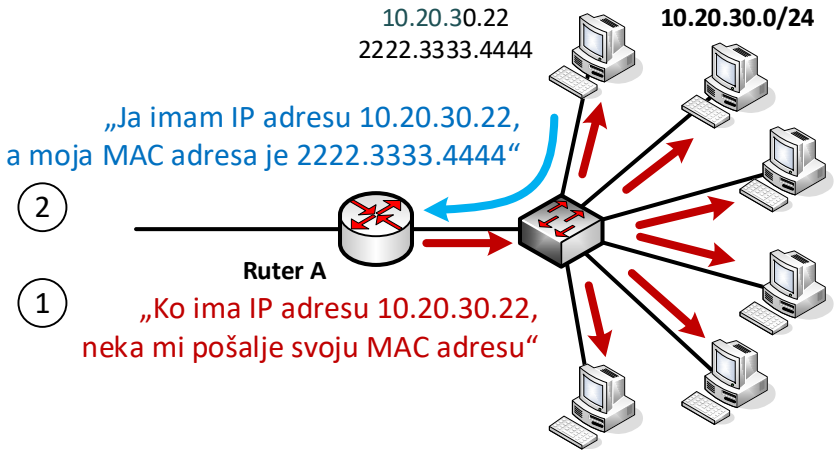
Komunikacija između krajnjih uređaja se obavlja prenosom IP paketa kroz mrežu preko niza rutera. Posmatrano na IP nivou, u svakom koraku IP paketi se prenose do naredne IP adrese: u prvom koraku izvorišni uređaj šalje paket na difoltni gejtvej, zatim ruteri šalju na *next-hop* adresu susednog rutera, dok se u poslednjem koraku IP paket isporučuje na odredišnu IP adresu. IP adresa difoltnog gejtveja je sastavni deo IP podešavanja svakog uređaja na LAN mreži, IP adresa za *next-hop* ruter je naučena posredstvom ruting protokola, dok je IP adresa krajnjeg odredišta navedena u zaglavlju IP paketa. IP paketi se prenose u okvirima L2 nivoa, za čije je prosleđivanje potrebna MAC adresa uređaja kome se okviri šalju. Stoga je na segmentima drugog sloja, a posebno u LAN mrežama Eternet tipa sa više učesnika, potreban mehanizam prepoznavanja MAC adresa na osnovu odgovarajućih IP adresa. Ovo je posao koji sprovodi **ARP protokol** (***Address Resolution Protocol***) [22].

Svi uređaji na LAN mreži Eternet tipa u pozadini, bez potrebe da se bilo šta podešava, sprovode ARP protokol. Kao i svaki protokol, informacije se razmenjuju putem posebno formatiranih poruka, a koje se u slučaju ARP-a prenose u Eternet paketima. Kao i IP protokol, i ARP ima registrovani identifikacioni broj koji se navodi u polju *type* u Eternet zaglavlju, a koji iznosi 806_{hex} [15].

ARP protokol se sprovodi svaki put kada je potrebno isporučiti IP paket, gde se za određenu IP adresu na istoj LAN mreži otkriva MAC adresa, što se sprovodi u dva koraka (Slika 9.1).

- **Korak 1**
  - Uređaj koji traži nedostajuću MAC adresu šalje poruku uputa (***ARP Request***) na Eternet brodkast adresu (FFFF.FFFF.FFFF) u kojoj navodi IP adresu za koju se traži MAC adresa. Slika 9.1 prikazuje upit koji šalje Ruter A da isporuči pristigli IP paket, ali to može biti bilo koji uređaj u LAN mreži.
  - Svi uređaji u LAN mreži primaju Eternet brodkast paket, iz koga izdvajaju APR poruku, uzimaju navedenu IP adresu i upoređuju sa svojom IP adresom.
- **Korak 2**
  - Samo uređaj koji prepozna svoju IP adresu u poruci ARP upita, uzvratiće sa porukom odgovora (***ARP Reply***), u koju postavlja svoju MAC adresu.
  - *ARP Reply* poruka se postavlja u novi Eternet okvir koji se šalje na unikast MAC adresu uređaja koji je poslao *ARP Request* poruku. Ova MAC adresa se takođe saznaje iz inicijalne *ARP Request* poruke, a ne iz Eternet okvira, budući da se ovaj posao obavlja na nivou ARP-a, a ne na nivou Eternet protokola.
  - Inicijalni uređaj prima Eternet okvir koji sadrži ARP odgovor, iz koga preuzima traženu MAC adresu.
  - Novootkrivena MAC adresa se zajedno sa odgovarajućom IP adresom upisuje u internu ARP tabelu (***ARP Table***), kako bi se ona mogla koristiti i za naredne komunikacije.
  - U slučaju da nijedan uređaj ne prepozna svoju IP adresu u poruci ARP upita, niko neće poslati odgovor. Posle određenog vremena čekanja (*timeout*), obično u trajanju od

nekoliko sekundi, uređaj koji je inicirao ARP upit će IP protokolu prijaviti grešku da ne može da isporuči IP paket.



Slika 9.1. Ilustracija sprovođenja ARP protokola u dva koraka

Na navedeni način uređaji po potrebi otkrivaju MAC adrese u LAN mreži i održavaju ARP tabelu sa njima pridruženim IP adresama. Prilikom potrebe za slanjem IP paketa, najpre će da se proverava sadržaj ARP tabele i jedino ako u tabeli ne postoji tražena IP adresa, sprovešće se slanje ARP upita. Svaki red u ARP tabeli ima svoj interni *timeout* period važenja, nakon čega se podatak briše, uz kasnije ponovno sprovođenje ARP upita u slučaju potrebe.

Format ARP paketa prikazuje Slika 9.2. U ARP terminologiji, oznaka „HA“ se odnosi na „*Hardware Address*“, odnosno MAC adresu (6 bajtova). Nazivi „*Sender*“ i „*Target*“ označavaju izvorišnu i odredišnu stranu za pojedinačne ARP poruke. Polje „*Protocol Type*“ tipično nosi vrednost 800_{hex}, što označava IP protokol, a potencijalno se može koristi i za druge protokole mrežnog nivoa.

1. bajt	2. bajt	3. bajt	4. bajt
Hardware Type		Protocol Type (0x0800)	
HLEN	PLEN	Operation	
Sender HA (1..4)			
Sender HA (5..6)		Sender IP (1..2)	
Sender IP (3..4)		Target HA (0..1)	
Target HA (3..6)			
Target IP (1..4)			

Slika 9.2. Format ARP paketa

9.2 Primer IP komunikacije

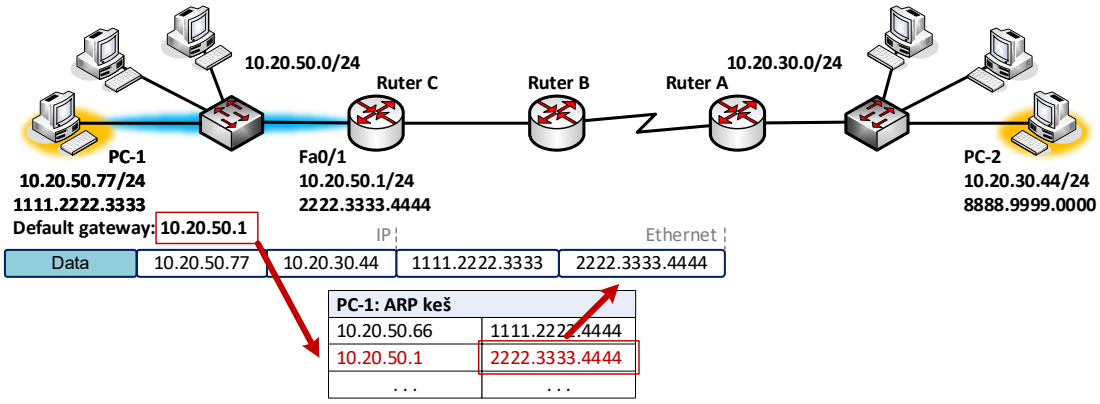
Na početku ovog poglavlja Slika 5.2 je ilustrovala komunikaciju na IP nivou koja se sprovodi između dva uređaja na različitim LAN mrežama, a koja se odvija preko rutera. U narednom primeru ćemo detaljno demonstrirati sve korake koji se sprovode u svakom od segmenata na putu do odredišta, i to sa aspekta sadržaja zaglavalja IP paketa i okvira na L2 nivou.

Istaknimo ponovo da se tom prilikom IP adrese izvorišta i odredišta ne menjaju tokom celog puta. Sa druge strane, MAC adrese izvorišta i odredišta su na svakom segmentu različite, budući da su od interesa samo za taj lokalni L2 segment. Ipak, na jednom L2 segmentu MAC adrese u Eternet okviru se ne menjaju prilikom prolaska paketa kroz svičeve.

Primer: Uređaj PC-1 sa adresom 10.20.50.77/24 šalje IP paket za PC-2 sa adresom 10.20.30.44/24

♦ **Korak 1**

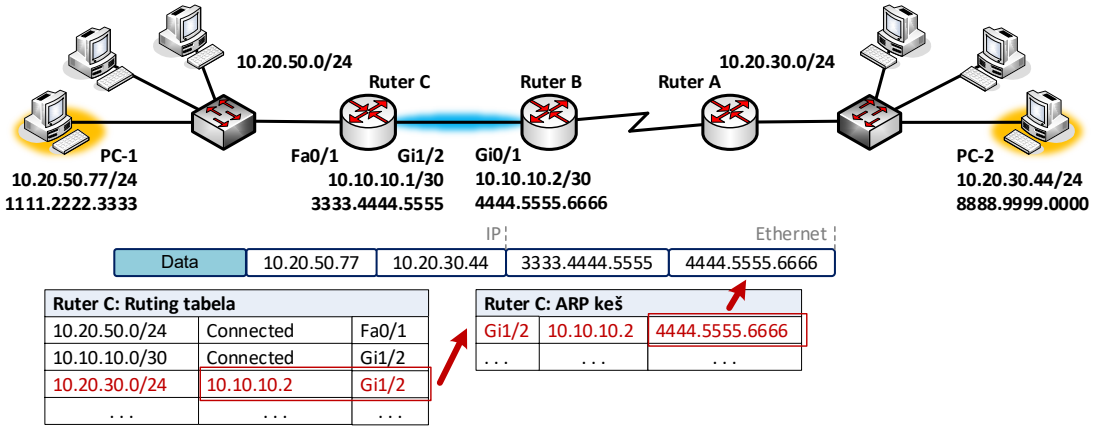
- PC-1 prepoznaje da odredišna IP adresa ne pripada istoj IP mreži 10.20.50.0/24, pa zaključuje da mora da ga pošalje na difoltni gejtvej.
- PC-1 u svojoj ARP tabeli traži i nalazi IP adresu difoltnog gejtveja, iz koje uzima njegovu MAC adresu. U slučaju da ona ne postoji u ARP tabeli, pokreće se ARP protokol.
- PC-1 formira Eternet okvir, gde za odredišno polje navodi nađenu MAC adresu difoltnog gejtveja.
- IP paket se stavlja u Eternet okvir, koji se preko sviča prenosi do Rutera C.



Slika 9.3. Primer IP komunikacije, korak 1

♦ **Korak 2**

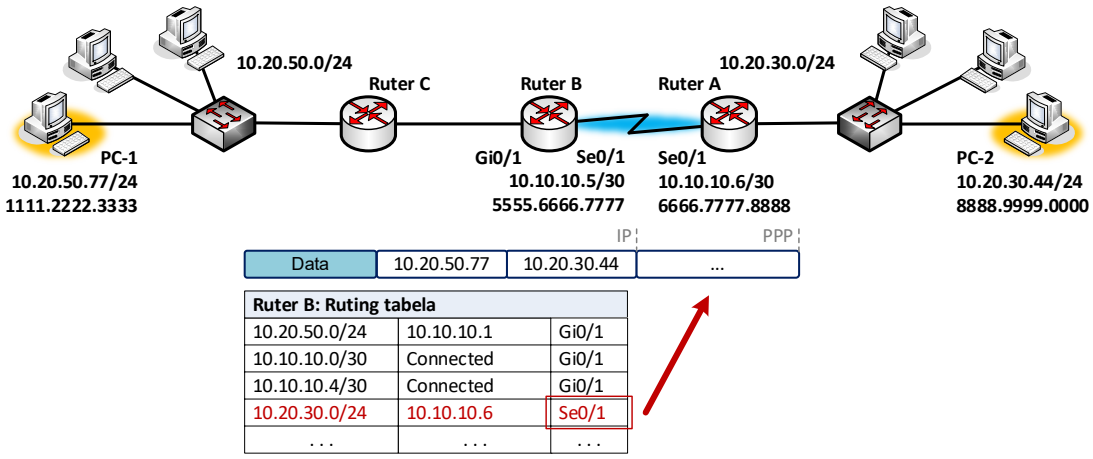
- Ruter C prihvata Eternet okvir i iz njega uzima enkapsulirani IP paket.
- Na osnovu odredišne IP adrese iz zaglavlja IP paketa u ruting tabeli se traži najspecifičnija IP adresa mreže, koja je u ovom slučaju 10.20.30.0/24.
- Iz nađene rute se uzima *next-hop* adresa narednog rutera (10.10.10.2).
- IP adresa za *next-hop* uređaj se ne koristi za slanje poruke na IP nivou, već se koristi da bi se iz ARP tabele Rutera C našla odgovarajuća MAC adresa, koja se koristi kao odredište novog Eternet okvira u koji se enkapsulira originalni IP paket.
- Eternet okvir se preko direktnog *point-to-point* linka šalje ka Ruteru B.



Slika 9.4. Primer IP komunikacije, korak 2

♦ **Korak 3**

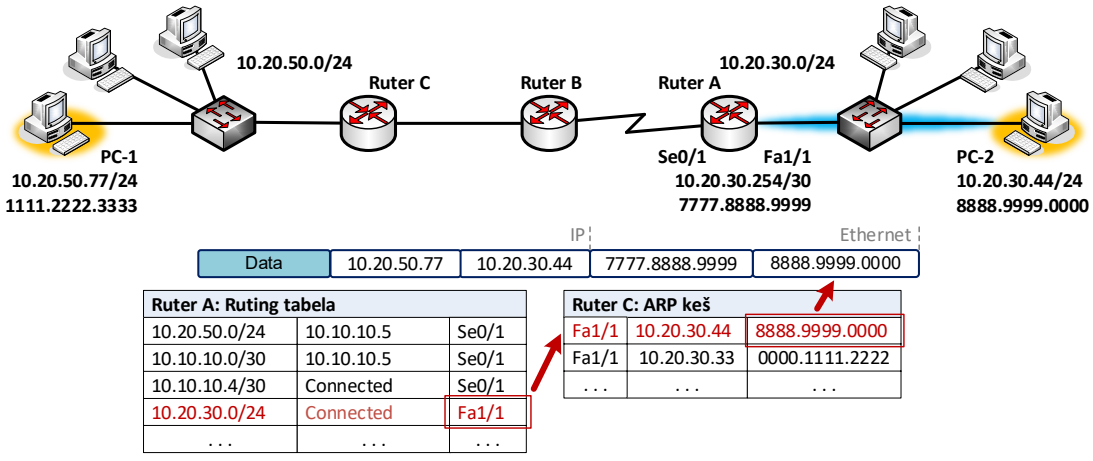
- Ruter B prihvata Eternet okvir i iz njega uzima enkapsulirani IP paket.
- Na osnovu odredišne IP adrese iz zaglavlja IP paketa u ruting tabeli se traži najspecifičnija IP adresa mreže (10.20.30.0/24).
- Iz nađene rute se uzima *next-hop* adresa narednog rutera (10.10.10.6).
- Segment prema narednom ruteru nije Eternet, već tzv. direktna serijska *point-to-point* veza, za čiji se prenos ne koristi MAC adresa (npr. *Point-to-Point Protocol*). IP paket se enkapsulira u novi okvir i direktno šalje do narednog Rutera A, u ovom slučaju bez potrebe za korišćenjem APR protokola.



Slika 9.5. Primer IP komunikacije, korak 3

♦ **Korak 4**

- Ruter A prihvata okvir na L2 nivou i iz njega uzima enkapsulirani IP paket.
- Na osnovu odredišne IP adrese iz zaglavlja IP paketa u ruting tabeli se traži najspecifičnija IP adresa mreže (10.20.30.0/24).
- Ovoj mreži odgovara *connected* ruta, iz čega se zaključuje da je LAN mreža direktno povezana na interfejs rutera, pa IP paket treba isporučiti krajnjem odredištu na toj LAN mreži.
- Iz IP paketa se uzima adresa odredišta, koja se traži u ARP tabeli, da bi se dobila uparena MAC adresa. U slučaju da IP adresa ne postoji u ARP tabeli, pokreće se ARP protokol.
- Ruter A formira Eternet okvir, gde za odredišno polje navodi nađenu MAC adresu odredišnog uređaja PC-2, a u telo okvira enkapsulira IP paket.
- Eternet okvir se preko sviča prenosi do uređaja PC-2.
- Uređaj PC-2 prihvata Eternet okvir, iz njega uzima enkapsulirani IP paket, prepoznaje svoju IP adresu u odredišnom polju, preuzima podatke iz tela IP poruke i prosleđuje protokolu višeg nivoa, čija je identifikacija navedena u polju *Protocol* u zaglavlju IP paketa.



Slika 9.6. Primer IP komunikacije, korak 4

Uvid u sadržaj ARP tabele moguće je ostvariti odgovarajućim komandama koje su dostupne na svim popularnim operativnim sistemima i ruterima raznih proizvođača, kao što ilustruje Slika 9.7.

```
C:\>arp -a
Interface: 147.91.4.8 --- 0x60003
  Internet Address      Physical Address      Type
  147.91.4.1            00-1b-90-41-78-00     dynamic
  147.91.4.48           00-e0-18-3e-09-c9     dynamic
  147.91.4.50           00-0b-cd-38-79-cb     dynamic
  147.91.4.58           00-60-b0-ef-90-ba     dynamic
```

```
[linux]$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
proxy3.amres.ac.rs       ether   90:e2:ba:d5:b6:f0   C                     eth0
vlan9-gw.rcub.bg.ac.rs   ether   00:1b:90:41:78:00   C                     eth0
147.91.1.51              ether   f4:a7:39:30:67:c2   C                     eth0
ns.rcub.bg.ac.rs         ether   32:63:35:37:37:31   C                     eth0
gaea.rcub.bg.ac.rs       ether   06:69:45:b0:2f:af   C                     eth0
helpdesk.amres.ac.rs     ether   00:50:56:89:ea:b8   C                     eth0
tesla.rcub.bg.ac.rs      ether   22:8a:15:f3:73:5d   C                     eth0
proxy1.amres.ac.rs       ether   90:e2:ba:d5:b8:18   C                     eth0
```

```
cisco:> show arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  147.91.108.160           0   000c.4212.23c9  ARPA   FastEthernet3/41
Internet  147.91.108.180          20   0017.a4d5.9b88  ARPA   FastEthernet3/41
Internet  147.91.108.181           3   0019.e010.a2c3  ARPA   FastEthernet3/41
Internet  147.91.108.177          10   0004.7612.e85b  ARPA   FastEthernet3/41
```

Slika 9.7. Primeri ARP tabela na Windows i Linux operativnim sistemima i Cisco uređajima

10. ICMP protokol

Činjenica da IP protokol nije pouzdan ne znači da ne radi dobro, već da ne garantuje da će svaki paket biti isporučen na odredište. Paketi koji nisu isporučeni su odbačeni na nekom delu puta, a ponekad i na samom odredišnom uređaju. IP protokol je u takvim situacijama nemoćan, ali postoji drugi mehanizam koji može da obavesti izvorišni uređaj da IP paket nije isporučen. Ovaj posao, kao i još neke pomoćne funkcije sprovodi novi protokol, koji se naziva *Internet Control Message Protocol*, poznatiji po skraćenom nazivu ICMP.

10.1 Vrste ICMP poruka

ICMP poruke se enkapsuliraju i prenose posredstvom IP poruka, pa se mogu slati između bilo kojih proizvoljno udaljenih IP uređaja. Standardizovana identifikacija ICMP protokola je „1“, što se u IP zaglavlju navodi u polju *Protocol*. U slučaju nastanka greške, uređaj koji je detektovao grešku, šalje ICMP poruku o grešci (***Error message***) uređaju koji je poslao originalni IP paket. Tom prilikom se prvih 100 bajtova originalnog IP paketa, uključujući i zaglavlje, postavlja u telo ICMP poruke, kako bi izvorišni uređaj mogao da prepozna koji paket nije isporučen.

Uzroci grešaka mogu biti različiti, pa su time različite i ICMP poruke koje se generišu, kao što sledi.

- ***Destination Unreachable***

  Poruke ovog tipa ukazuju da IP paket nije isporučen na odredište. Razlozi za to mogu biti različiti, što dovodi do sledećih podtipova poruka:

  - ***Can’t fragment***

    IP paket je trebao da se fragmentira, ali je u njemu bio postavljen „*Don't fragment*“ fleg koji je to sprečio.

  - ***Network unreachable***

    Određeni ruter na putu do odredišta u svojoj ruting tabeli nema IP adresu mreže kojoj pripada odredišna adresa, uključujući i difoltnu rutu. Ruter stoga nije imao informaciju na koju stranu da prosledi paket, pa je morao da ga uništi.

  - ***Host unreachable***

    Poslednji ruter na putu prepoznaje da odredišna IP adresa pripada mreži na koju je on povezan, ali se odredište ne odaziva na ARP upit.

  - ***Protocol unreachable***

    Paket je stigao do odredišnog uređaja i njegovog IP nivoa, ali protokol višeg sloja, koji je naveden u polju *Protocol* IP zaglavlja, nije aktivan i paket nije mogao da mu se isporuči.

  - ***Port unreachable***

    Paket je stigao do odredišnog uređaja i protokola na četvrtom nivoa (*transport layer*), ali aplikacija nije aktivna, a koja se identifikuje kroz polje *Port*[10] zaglavlja četvrtog nivoa, pa paket nije mogao da se isporuči.

[10] Funkcionalnost protokola četvrtog nivoa i značenje pojedinih polja iz zaglavlja se objašnjava u trećem delu knjige, a do tada ne treba mešati *Port* kao identifikaciju aplikacije na krajnjem uređaju sa fizičkim mrežnim portom na sviču.

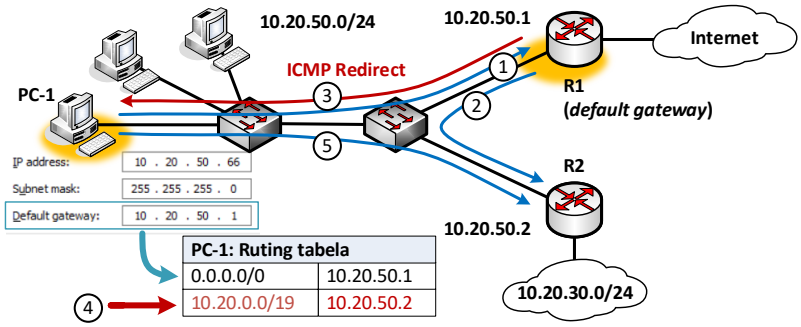
- ***Time exceeded***

  Pri prosleđivanju IP paketa ruteri smanjuju vrednost TTL polja (*Time to Live*) za 1, a ako ova vrednost u nekom trenutku postane nula, paket će da bude uništen, uz slanje ove vrste ICMP poruke.

- ***Redirect***

  Za komunikaciju izvan LAN mreže uređaji šalju IP pakete najpre difoltnom gejtveju. Ipak, to ne mora da bude jedini ruter povezan na posmatranu LAN mrežu, pa time ne nužno i najbolji izlaz iz mreže. Ako difoltni gejtvej prepozna da ruta za određeni IP paket ukazuje na *next-hop* koji je na istoj LAN mreži, tada se paket vraća kroz isti interfejs i prosleđuje do drugog rutera (Slika 10.1, korak 2). Iako paket nije uništen, već ispravno prosleđen, ruter prepoznaje da ova putanja nije optimalna i da uređaj za odgovarajuću mrežu može direktno da koristi drugi ruter, o čemu obaveštava izvorišni uređaj slanjem ove vrste ICMP poruke (Slika 10.1, korak 3).

  Ovom prilikom istaknimo da i računari poseduju ruting tabele koje koriste za slanja IP paketa. Šta više, konfigurisanjem difoltnog gejtveja na računaru se kreira difoltna ruta za mrežu 0.0.0.0/0. ICMP *Redirect* poruka koja pristigne na uređaj izazvaće da se za određenu mrežu u ruting tabelu upiše specifičnija ruta koja ukazuje na drugi ruter koji za tu IP mrežu predstavlja izlaz iz LAN mreže (Slika 10.1, korak 4). Nakon toga će naredni IP paketi da slede ovu novu rutu i direktno da se šalju na drugi ruter. Primetimo da u ovom slučaju IP paket nije izgubljen, pa se ova ICMP poruka više može smatrati za poruku obaveštenja, nego poruku greške.



Slika 10.1. Primer slanja ICMP Redirect poruke

Komande za listanje ruting tabele na *Linux* i *Windows* operativnim sistemima za prethodni primer ilustruje naredna slika. Primetimo da postoje i rute za mrežne adrese posebnih namena, kao što su lokalne adrese uređaja, tzv. *loopback* (127.0.0.0/8), multikast adrese (224.0.0.0/4), i broadkast (10.20.255.255, 255.255.255.255).

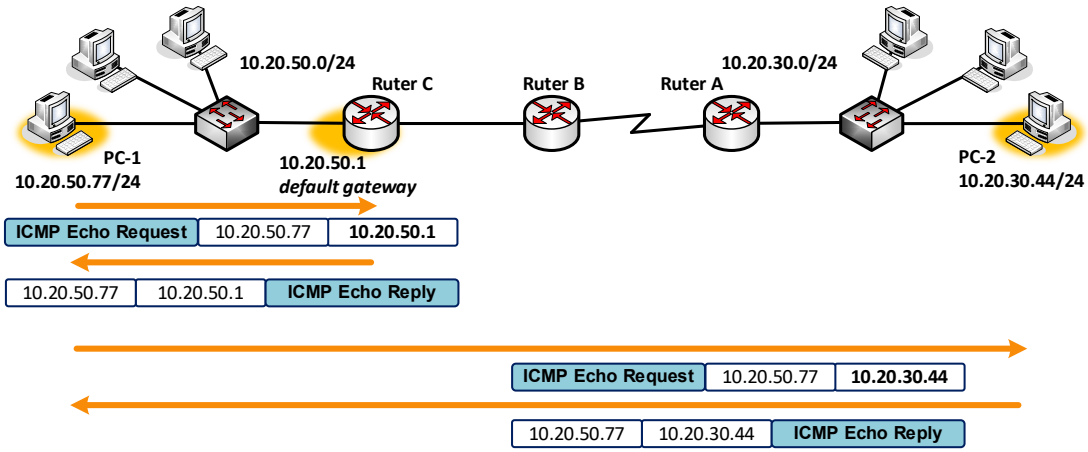
```
Linux# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.20.0.0       10.20.50.2      255.255.224.0   U     0      0        0 eth0
default         10.20.50.1      0.0.0.0         UG    0      0        0 eth0
```

```
C:\Documents and Settings>route print
Active Routes:
Network Destination        Netmask          Gateway       Interface   Metric
          0.0.0.0          0.0.0.0      10.20.50.1     10.20.50.66       20
        127.0.0.0        255.0.0.0        127.0.0.1      127.0.0.1        1
         10.20.0.0    255.255.224.0      10.20.50.2     10.20.50.66       20
    10.20.255.255  255.255.255.255      10.20.50.2     10.20.50.66       20
        224.0.0.0        240.0.0.0      10.20.50.2     10.20.50.66       20
  255.255.255.255  255.255.255.255      10.20.50.2     10.20.50.66       20
Default Gateway:       10.20.50.1
```

Slika 10.2. Primer ruting tabela na Linux i Windows operativnim sistemima

Osim poruka o greškama, koje nastaju na osnovu određenih događaja koji nastanu u mreži i šalju se u jednom smeru prema izvorišnom uređaju, ICMP podržava i poruke komunikacije u oba smera, slanjem upita na koji se dobija odgovor (***Query message***).

- *Echo Request*, *Echo Reply* – Svaki uređaj može da pošalje ICMP poruku upita (*Echo Request*) bilo kom drugom uređaju na mreži, na osnovu koje se uzvraća ICMP odgovorom (*Echo Reply*). Ni upit ni odgovor ne nose posebno korisne informacije, ali je zato izuzetno korisna sama činjenica da su poruke razmenjene, čime se potvrđuje da između uređaja postoji IP veza, što predstavlja i njihovu osnovnu namenu (Slika 10.3).



Slika 10.3. ICMP Echo Request i Echo Reply poruke

10.2 Primena ICMP protokola

Komanda *ping*

Podaci u računarskim mrežama se tipično razmenjuju između aplikacija krajnjih uređaja. U slučaju eventualnih grešaka i odbacivanja IP paketa, poslaće se odgovarajuće ICMP poruke o grešci, a stvar je implementacije softvera na krajnjem uređaju kako će se ove poruke tretirati. U slučaju greške u komunikaciji na IP nivou, korisničke aplikacije obično samo ispoljavaju prekid u radu i bez informacije o eventualno primljenim ICMP porukama o greškama. Korisnici sa malo više znanja iz osnova računarskih mreža u takvim situacijama imaju mogućnosti da ručno provere konektivnost krajnjih uređaja, odnosno njihovu dostupnost na IP nivou. Ovo se

postiže korišćenjem ICMP *Echo Request* i *Echo Reply* poruka, koje su u okviru različitih operativnih sistema standardno implementirane kao posebna komanda pod nazivom ***ping***.

Ping komanda predstavlja prvi i osnovni korak u dijagnosticiranju problema u radu računarskih mreža. U svom osnovnom obliku komanda šalje ICMP *Echo Request* poruke do navedene IP adrese i detektuje prijem odgovarajućih ICMP *Echo Reply* poruka. U slučaju da odgovori izostanu, u rezultatu će biti prikazan gubitak paketa. To se često tumači da navedena IP adresa nije dostupna, što ne mora uvek da bude tačno, jer paketi mogu da stignu do krajnjeg uređaja, ali da povratni paketi budu izgubljeni u suprotnom smeru. Tačnije interpretacija je da ne postoji dvosmerna komunikacija sa navedenom IP adresom.

Greška u komunikaciji može da bude u bilo kojoj tački na putu između krajnjih uređaja, a taj put ne mora da bude isti u odlaznom i dolaznom smeru. Zato je korisno da se proveri dostupnost i nekih drugih uređaja u mreži, dalje ili bliže u odnosu na korisnikov računar. Nisu retke ni greške u LAN mreži samog korisnika, pa je uvek korisno proveriti i dostupnost prve tačke u IP komunikaciji, a to je difoltni gejtvej.

Primeri izvršavanja ping komande su dati u nastavku, i to na *Windows* i *Linux* računarima, kao i na ruteru (Slika 10.4). Osim izveštaja o prijemu odgovara za svaki poslati paket, ping komanda daje i druge korisne informacije, kao što su vreme propagacije u oba smera (*round trip time* – RTT), TTL, procenat izgubljenih paketa itd. U navedenim primerima nije bilo gubitka paketa, a korišćena je IP adresa *google* veb sajta, što je moglo da se zadaje i u simboličkom obliku sa „ping www.google.com“.

```
C:\Windows\System32>ping 216.58.214.196
Pinging 216.58.214.196 with 32 bytes of data:
Reply from 216.58.214.196: bytes=32 time=39ms TTL=50
Reply from 216.58.214.196: bytes=32 time=39ms TTL=50
Reply from 216.58.214.196: bytes=32 time=39ms TTL=50
Reply from 216.58.214.196: bytes=32 time=39ms TTL=50
Ping statistics for 216.58.214.196:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 39ms, Maximum = 39ms, Average = 39ms
```

```
Linux# ping -c 3 216.58.214.196
PING 216.58.206.68 56(84) bytes of data.
64 bytes from mil07s08-in-f4.1e100.net (216.58.206.68): icmp_seq=1 ttl=53 time=22.4 ms
64 bytes from mil07s08-in-f4.1e100.net (216.58.206.68): icmp_seq=2 ttl=53 time=22.5 ms
64 bytes from mil07s08-in-f4.1e100.net (216.58.206.68): icmp_seq=3 ttl=53 time=22.5 ms
--- www.google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 22.490/22.556/22.621/0.048 ms
```

```
Router>ping 216.58.214.196
Sending 5, 100-byte ICMP Echos to 216.58.214.196, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 32/33/36 ms
```

Slika 10.4. Ping komanda na sistemima Windows, Linux i na ruteru

U slučaju opterećenja određenih linkova ili rutera na putanji do odredišta, pojedini paketi mogu biti značajnije usporeni, a neki i odbačeni. Periodičnim slanjem veće količine ping paketa i uvidom u vreme propagacije i procenat eventualno izgubljenih paketa, ping komanda može da pruži i osnovne indikacije o kvalitetu veze.

Potrebno je naglasiti da čak iako se ne dobije odgovor posredstvom ping komande, to ne mora nužno da znači da postoji problem u konekciji na IP nivou. Pojedini ruteri zabranjuju prolazak ICMP *Echo Request* i *Echo Reply* poruka, a često ni serveri ne odgovaraju na ping zahteve,

kako se ne bi izlagali dodatnom opterećenju. Opterećenje je zanemarljivo pri normalnom povremenom korišćenju, ali se ova mogućnost može zloupotrebiti generisanjem enormno velikog broja paketa. Oni se mogu slati sa jedne adrese, što predstavlja tzv. DoS napad (*Denial of Service*) ili distribuirano sa većeg broja uređaja, tzv. *botnet* mreže, što se označava kao DDoS napad (*Distributed DoS*).

Komanda *traceroute*

Ping komanda proverava konektivnost na IP nivou, ali u slučaju greške u komunikaciji, ona ne daje informacije gde se javlja greška na putu do odredišta. Za to služi komanda *traceroute*[11], koja na indirektan način otkriva sve rutere na putanji do odredišta, odnosno do tačke u kojoj se paketi eventualno odbacuju.

Komanda *traceroute* do odredišnog uređaja u više iteracija šalje po tri *ICMP Echo Request* poruke, a u pojedinim implementacijama paketi se šalju i na aplikativnom nivou. Specifičnost je u tome što se inicijalna vrednost TTL polja (*Time-to-Live*) u IP zaglavlju postavlja na 1. Rezultat je da poruke dođu do prvog rutera na putu, koji ih uništava zato što je „istekao" TTL, a izvorištu se šalje ICMP poruka „*Time exceeded*". Na osnovu toga se prepoznaje ko je prvi ruter na putu, kao i to da je on dostupan na IP nivou. Nakon toga se u svakom sledećem koraku TTL vrednost poveća za 1, čime se dosеže, a ujedno i detektuje, naredni ruter na putu, sve dok se ne pristigne do odredišta. U slučaju da se paketi gube na nekom ruteru ili linku, neće se dobiti povratne ICMP „*Time exceeded*" poruke, što će se detektovani nakon predefinisanog *timeout* perioda.

```
C:\Windows\System32>tracert www.google.com
Tracing route to www.google.com [216.58.214.196]
over a maximum of 30 hops:
  1    <1 ms    <1 ms    <1 ms  vlan10-gw.rcub.bg.ac.rs [147.91.4.1]
  2    <1 ms    <1 ms    <1 ms  amres-j-r.amres.ac.rs [147.91.6.129]
  3    <1 ms    <1 ms    <1 ms  amres-mpls-core.amres.ac.rs [147.91.5.144]
  4    <1 ms    <1 ms    <1 ms  amres-ip-core.amres.ac.rs [147.91.5.145]
  5     7 ms    10 ms     7 ms  it2.it1.eumedconnect.net [83.97.88.5]
  6    21 ms    21 ms    21 ms  62.40.98.45
  7    21 ms    21 ms    21 ms  ae8.mx1.mil2.it.geant.net [62.40.98.188]
  8    22 ms    22 ms    28 ms  72.14.203.32
  9    21 ms    20 ms    21 ms  108.170.245.73
 10    26 ms    26 ms    25 ms  72.14.233.132
 11    43 ms    41 ms    41 ms  108.170.236.250
 12    40 ms    40 ms    40 ms  74.125.242.225
 13    41 ms    41 ms    41 ms  72.14.233.75
 14    39 ms    39 ms    39 ms  bud02s23-in-f4.1e100.net [216.58.214.196]
Trace complete.
```

Slika 10.5. Traceroute komanda na Windows operativnom sistemu

Komanda *traceroute* omogućava da se otkrije putanja kojom paketi propagiraju prema odredištu, kako prikazuje Slika 10.5. Napomenimo da putanja u suprotnom smeru može biti drugačija, ali se to može detektovati izvršavanjem *traceroute* komande na udaljenom računaru.

[11] Na *Windows* operativnim sistemima naziv komande je „*tracert*"

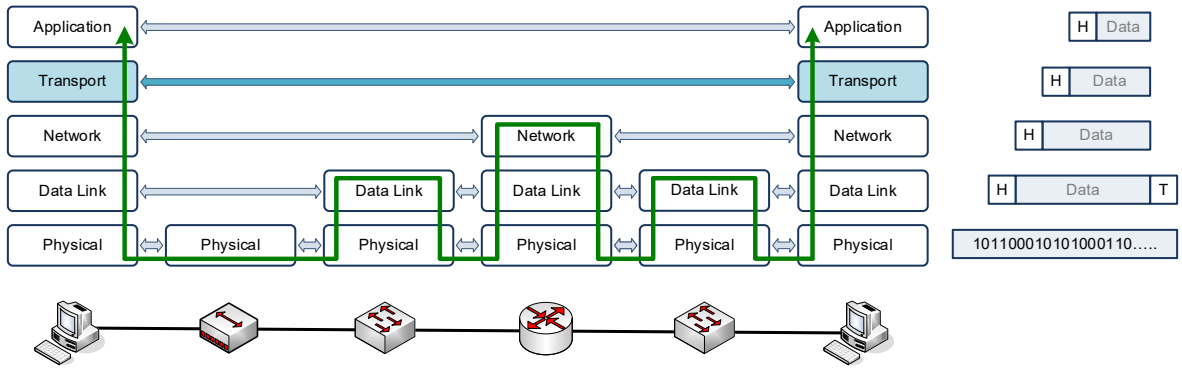
Treći deo:

Viši slojevi

11. Transportni sloj

11.1 Opšte funkcije transportnog sloja

Eternet tehnologija i ostali protokoli na drugom nivou podatke prenose u lokalnom domenu između manjeg broja učesnika. IP protokol na mrežnom nivou omogućava prenos podataka između proizvoljnog broja učesnika na proizvoljnim rastojanjima, što omogućava komunikaciju na globalnom nivou u Internet mreži. Treba imati u vidu da se razmena podataka se ne sprovodi između krajnjih uređaja, već između aplikacija na tim uređajima. Aplikacije su raznovrsne i i više njih se može istovremeno koristiti na jednom uređaju. Sve one dele određene zajedničke potrebe u pogledu slanja podataka na mrežu i prijema sa mreže, a koje nisu podržane IP protokolom. Kako ove potrebe ne bi obezbeđivala svaka aplikacija za sebe, one su objedinjene u četvrtom, tzv. transportnom sloju, koji omogućava aplikacijama da koriste mrežne usluge. Drugim rečima, transportni sloj omogućava transport aplikativnih podataka sa-kraja-na-kraj na nivou cele mrežne infrastrukture, bez obzira na njenu veličinu, strukturu, organizaciju i primenjene tehnologije nižih nivoa (Slika 11.1).

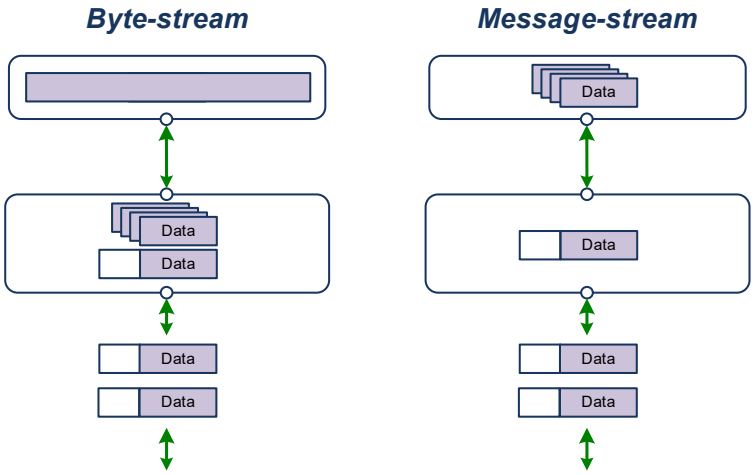


Slika 11.1. Transport aplikativnih podataka

Posmatrano u kontekstu podele na slojeva, transportni sloj na uređaju preuzima podatke od više aplikacija, enkapsulira ih u svoje poruke, koje predaje IP protokolu da se pošalju do udaljenog uređaja. Na prijemnoj strani transportni sloj radi obrnuti proces – od IP sloja preuzima poruku transportnog sloja, dekapsulira je i predaje podatke odgovarajućoj aplikaciji.

Razmena podataka sa aplikacijama se može sprovoditi u dva režima:

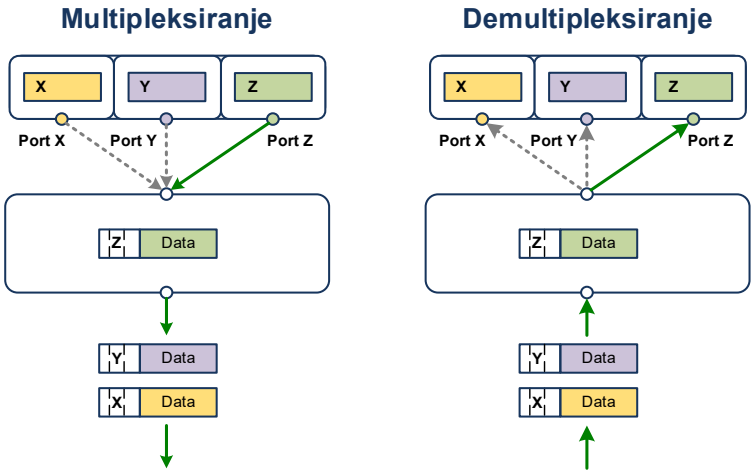
- ***Byte-stream*** – I aplikacija i transportni sloj podatke tretiraju kao niz bajtova, koji se razmenjuju u manjim celinama. Nezavisno od aplikacije, transportni sloj sprovodi podelu primljenih podataka na delove određene dužine, tzv. segmente, koje razmenjuje sa uparenim transportnim slojem na drugoj strani komunikacije. Ovaj proces se naziva **segmentacija**. Na prijemnoj strani segmenti se objedinjavaju uz odražavanje originalnog poretka i u nezavisnim celinama prenose odredišnoj aplikaciji.
- ***Message-stream*** – Aplikacija samostalno priprema, odnosno deli podatke u manje celine i prosleđuje ih transportnom sloju, koji ih u celini prenosi drugoj strani u komunikaciji. Prijemna strana primljene podatke u celini predaje odredišnoj aplikaciji, koja ih objedinjuje u originalne aplikativne podatke.



Slika 11.2. Enkapsulacija i transport aplikativnih podataka

Transportni sloj preuzima i prosleđuje podatke od više aplikacija istovremeno (*multipleksiranje*), a na prijemnoj strani podatke razvrstava na aplikacije kojima su podaci i namenjeni (*demultipleksiranje*). Stoga je potrebno da svaka aplikacija ima svoju jedinstvenu identifikaciju, koja se kao podatak prenosi u zaglavlju paketa transportnog nivoa (Slika 11.3). Ova identifikacija se naziva **port** (*port*) i izražava se kao celobrojna vrednost u opsegu od 0 do 65.535.

Potrebno je istaći da se portovi ne odnose na celokupni transportni sloj, već na određeni protokol transportnog nivoa, a kojih ima više, a čije se identifikacije takođe navode u zaglavlju nižeg sloja, u ovom slučaju u polju *Protocol* u zaglavlju IP paketa.



Slika 11.3. Transport aplikativnih podataka

Za razliku od protokola drugog i trećeg nivoa, koji za multipleksiranje i demultipleksiranje u zaglavljima u oba smera prenose iste identifikacije protokola višeg nivoa, čije su vrednosti standardom ustanovljene, aplikacije koje međusobno komuniciraju imaju različite vrednosti svojih portova. Tom prilikom se razlikuju dve uloge u komunikaciji: aplikacija koja inicira komunikaciju, tzv. **klijent** (*client*) i aplikacija koja odgovara na zahteve klijenata, tzv. **server** (*server*).

Da bi klijenti mogli da iniciraju komunikaciju i pristupe severima[12], serverske aplikacije su stalno dostupne na fiksnim i unapred poznatim portovima, na kojima mogu da ostvare istovremene komunikacije sa velikim brojem različitih klijenata. Budući da klijenti iniciraju komunikaciju, oni mogu da koriste proizvoljne vrednosti portova, na osnovu koji će ih serveri prepoznavati.

Iz navedenih razloga, brojevi portova se dele u sledeće opsege:

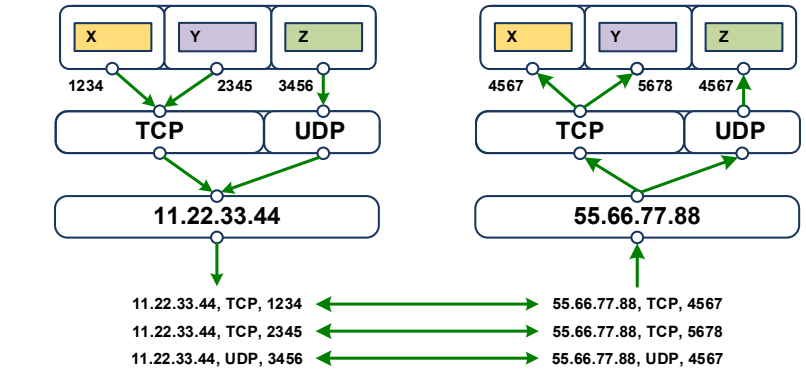
- ***Well-known Ports*** – u opsegu vrednosti od 0 do 1.023, koji su unapred dodeljeni za određene serverske aplikacije.
- ***Registered Ports*** – u opsegu vrednosti od 1.024 do 49.151, koji mogu da se koriste i za serverske i za klijentske aplikacije.
- ***Private (Dynamic) Ports*** – u opsegu vrednosti od 49.152 do 65.535, koji mogu da se koriste samo za klijentske aplikacije.

IANA organizacija sprovodi registraciju portova za serverske aplikacije i to iz opsega *Well-known Ports* i *Registered Ports.* Nasuprot tome instance klijentskih aplikacija po potrebi biraju portove iz opsega *Registered Ports* i *Private (Dynamic) Ports*, uz jedini uslov da brojevi portova za određeni protokol transportnog nivoa budu jedinstveni na nivou uređaja. Takođe, jedna vrsta aplikacije na strani klijenta može da ima više instanci i svaka od njih mora da ima jedinstvene portove kako bi se međusobno razlikovali (npr. različiti tabovi veb pretraživača).

Navedenim mehanizmom portovi jedinstveno identifikuju instance aplikacija unutar protokola transportnog nivoa na jednom uređaju, dok je uređaj na mreži jedinstveno identifikovan preko IP adrese. Kombinovanjem ovih identifikacija, odnosno IP adrese, porta i identifikacije primenjenog transportnog protokola, dobija se jedinstvena identifikacija aplikacije na mreži, što se naziva **soket** (*socket*).

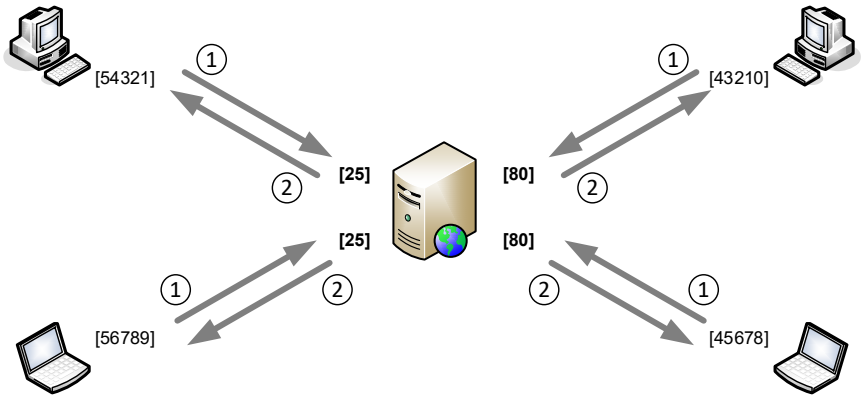
U međusobnoj komunikaciji aplikacija, soket se može tretirati kao „ulaz u aplikaciju“. Soketi za serverske aplikacije su unapred poznati, dok klijenti koji im pristupaju samostalno biraju lokalno raspoložive portove, čime dobijaju svoj soket kojim iniciraju komunikaciju sa serverom (Slika 11.4). Kao što se IP adrese klijenta i servera, zajedno sa identifikacijom transportnog protokola, prenose u zaglavlju IP paketa, tako se i brojevi portova klijenta i servera prenose u zaglavlju transportnog protokola kao izvorišni port (*source port*) i odredišni port (*destination port*). Odredišni port služi da se podaci predaju odgovarajućoj aplikaciji, dok je izvorišni port neophodan za identifikaciju aplikacije na drugoj strani. U inicijalnom zahtevu koji pristiže od klijenta, izvorišni port sadrži vrednost koju je izabrao klijent, na osnovu čega server može da pošalje odgovor u suprotnom smeru.

[12] U ovom kontekstu „server“ je aplikacija, a ne uređaj.



Slika 11.4. Komunikacija na nivou soketa

Na ovaj način, putem soketa, klijent može da pokrene više komunikacija prema jednom ili većem broju klijenata. Takođe, na jednom uređaju može da se koristi više različitih serverskih aplikacija, koje mogu da opslužuju više klijenata (Slika 11.5). U slučaju komunikacije dve aplikacije koje primarno imaju serversku namenu, jedna aplikacija mora da inicira komunikaciju kao klijent, biranjem klijentskog porta i tako pristupi drugom serveru.



Slika 11.5. Različite serverske aplikacija ne jednom uređaju nezavisno opslužuju više klijenata

Osim enkapsulacije i multipleksiranja, koje su osnovne funkcije svakog sloja, pa time i transportnog, ovo je prilika da se obezbede i dodatne funkcije koje niži slojevi ne pružaju, a mnoge aplikacije zahtevaju, kao što su sledeće:

- **Uspostavljanje veze** (*Connection-oriented*) – Obe strane u komunikaciji moraju da se saglase za razmenu podataka, tako što se komunikaciona sesija eksplicitno uspostavlja, održava i na kraju raskida.
- **Održavanje redosleda podataka** (*Ordered data reconstruction*) – Segmenti mogu putovati po različitim putanjama i pristići u izmenjenom redosledu, ali prijemna strana rekonstruiše njihov originalni redosled.
- **Pouzdan prenos** (*Reliable transmission*) – Garantuje se pouzdan prenos svih aplikativnih podataka u celini, tako što se izgubljeni ili oštećeni segmenti detektuju i ponovo šalju.
- **Kontrola toka** (*Flow control*) – Dinamičko prilagođavanje protoka podataka u zavisnosti od mogućnosti i trenutnog opterećenja mreže.

Navedene dodatne funkcije su korisne za mnoge aplikacije, ali njihova realizacija opterećuje strane u komunikaciji i usporava prenos podataka, pa su za pojedine potrebe one otežavajuće i nepoželjne. Iz ovog razloga realizovana su dva protokola transportnog nivoa:

- **UDP** (***User Datagram Protocol***) – transportni protokol koji realizuje samo osnovne funkcije
- **TCP** (***Transmission Control Protocol***) – transportni protokol koji realizuje i navedene dodatne funkcije.

Većina aplikacija ima potrebe za dodatnim funkcijama i koristi TCP protokol, pa je uobičajeni naziv za celu familiju Internet protokola **TCP/IP**.

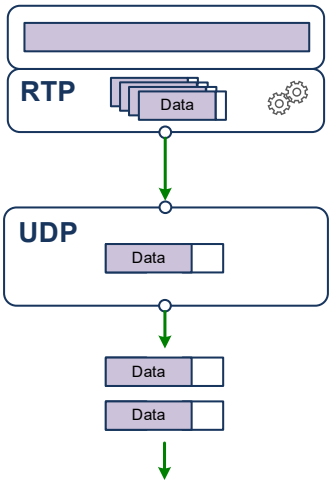
11.2 UDP protokol

UDP protokol obezbeđuje samo osnovne funkcije, nezavisnim prenosom celokupnih poruka koja se dobijaju od aplikacije (*datagram*) u *message-stream* režimu, bez potrebe za prethodnim uspostavljanjem komunikacije (***connectionless***) [23]. Identifikacija UDP protokola koja se koristi u zaglavlju IP paketa ima vrednost 17.

UDP protokol obezbeđuje jednostavnu i brzu komunikaciju, ali ne i pouzdan prenos podataka u smislu da se ne garantuje prenos paketa. Takođe, ne može se detektovati da je neki paket izgubljen, ali se može detektovati oštećenje primljenih paketa. Priroda aplikacija koje koriste UDP protokol je takva da im je primarna potreba jednostavnost i brzina, dok mogu da tolerišu povremeni gubitak paketa. Dve su opšte grupe ovakvih aplikacija:

- Jednostavne aplikacije specijalizovane namene koje prenose manju količinu podataka, često u jednom smeru, periodično ili na zahtev, koji se u slučaju gubitka podataka može ponoviti. Primer ovakvih aplikacija su:
  - RIP ruting protokol, koji za prenos podataka koristi UDP port 520
  - SNMP (*Simple Network Management Protocol*), protokol za očitavanja (periodično ili na zahtev) operativnih parametara na uređajima, koji koristi port 161
  - *Syslog* protokol za prenos sistemskih logova sa uređaja na određeni server, koji koristi UDP port 514.
- Aplikacije za prenos video i audio sadržaja u realnom vremenu (*real-time applications*), kao što su video konferencije, IP telefonija, IP televizija (IPTV) i slično. Za ove aplikacije se gubitak pojedinih paketa može tolerisati, ali je bitan kontinuitet prenosa podataka, i to uz malo kašnjenje. Kod interaktivnih dvosmernih komunikacija (npr. IP telefonija), da bi se održao prihvatljiv kvalitet servisa, potrebno da se podaci između dve strane prenose za ne više od 200ms (*one-way delay*). Takođe je bitno da ovo kašnjenje tokom vremena bude ujednačeno, bez velikih varijacija (npr. do +/- 30ms), što je parametar koji se naziva **džiter** (***jitter***). Aplikacije koje primenjuju jednosmernu komunikaciju, kao što je IPTV, mogu koristiti brodkast ili multikast prenos.

  Budući da se radi o složenim aplikacijama posebne namene, koje zahtevaju prenos velike količine podataka u kontinuitetu, one se obično oslanjaju na korišćenje posebnog protokola, koji se pozicionira kao međusloj između aplikacije i UDP protokola, tzv. ***Real-Time Transport Protocol*** (**RTP**). Na ovaj način se obezbeđuju posebne funkcionalnosti koje nisu podržane od strane UDP protokola, kao što su serijalizacija bajtova na segmente, baferovanje, kontrola džitera i drugih parametara.

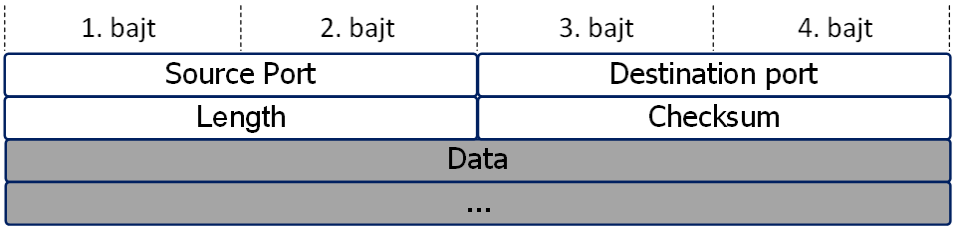


Slika 11.6. Real-Time Protocol, za obezbeđivanje dodatnih specifičnih funkcija za prenos audio i video sadržaja u realnom vremenu

Format UDP zaglavlja je takođe jednostavan, uz zauzeće svega 8 bajta (Slika 11.7). Osim izvorišnog i odredišnog porta, postoje još samo dva polja:

- ***Length*** – Odnosi se na dužinu celog UDP paketa, uključujući i zaglavlje, a izraženo je u bajtovima. Polje je dužine 2 bajta, a najveća količina aplikativnih podataka koji se mogu preneti je 65.507 bajtova, budući da je potrebno sačuvati 8 bajtova za UDP zaglavlje, kao i 20 bajtova za zaglavlje IP paketa, koje za svoju dužinu dozvoljava maksimalno 65.535 bajtova. Ipak, zbog ograničenja veličine paketa na drugom nivou (MTU), koje je obično manje od 1500 bajtova, radi izbegavanja fragmentiranja na IP nivou, veličina UDP paketa je u realnosti znatno manja.
- ***Checksum*** – Odnosi se na kontrolu greške na nivou bita, primenom metode „prvog komplementa“ (invertovane binarne cifre) sume reči od 16 bita nad sledećim podacima:
  - UDP zaglavlje, podrazumevajući sve nule u polju *Checksum*
  - Aplikativni podaci koji se prenose u UDP paketu (polje *Data*)
  - „Pseudo zaglavlje“ (*Pseudo Header*), koje se sastoji od IP adrese izvorišta i odredišta (iako je to deo IP zaglavlja, a ne UDP poruke), identifikacije protokola (u ovom slučaju broj 17) i dužine UDP paketa (polje *Length*). Pseudo zaglavlje se privremeno formira i na strani izvorišta pri generisanju *Checksum* polja i na odredišnoj strani pri proveri da li je došlo do bitske greške, ali se ono ne prenosi unutar UDP paketa. Primetimo da se provera grešaka nad IP adresama već sprovodi na nivou IP paketa, ali je ovakvo računanje ipak zadržano i na nivou UDP paketa, iako suštinski nije neophodno.

  Zbog opisanog načina generisanja *Checksum* vrednosti, njenim binarnim sabiranjem sa odgovarajućim podacima koji su podeljeni u reči od po 16 bajta, daje vrednost sa svim binarnim jedinicama. Ovo se sprovodi na prijemnoj strani, a u slučaj da se dobija bar jedan bit koji ima vrednost nula, smatra se da je došlo do greške, i ceo paket se odbacuje.



Slika 11.7. Format UDP zaglavlja

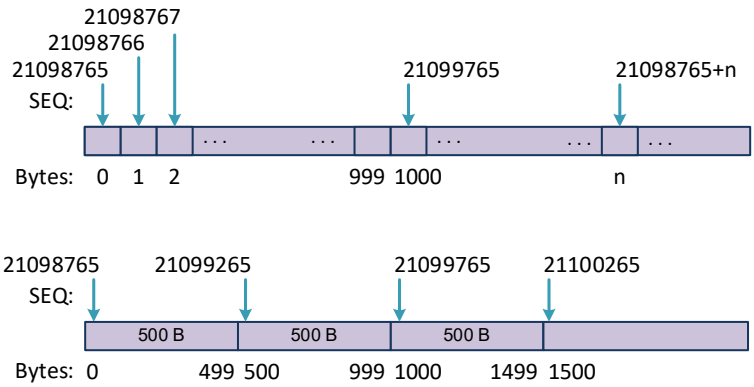
11.3 TCP protokol

Većina aplikacija od mreže očekuje pouzdan prenos podataka, što zajedno sa ostalim dodatnim funkcijama transportnog sloja obezbeđuje TCP protokol (*Transmission Control Protocol*). Inicijalna verzija TCP (*Transmission Control Protocol*) protokola je objavljena 1980. godine [24], koja je više puta modifikovana i unapređivana [25], ali ipak uz zadržavanje svih osnovnih koncepata. Identifikacija TCP protokola koja se koristi u zaglavlju IP paketa ima vrednost 6.

TCP je *byte-stream* protokol, što znači da se aplikativni podaci tretiraju kao niz bajtova, koje aplikacija u manjim celinama sukcesivno prosleđuje TCP protokolu. Nezavisno od toga, TCP pristigle bajtove deli na još manje delove, koji se nazivaju **segmenti**, gde se svaki segment prenosi u jednoj TCP poruci, koja se enkapsulira u IP paket. Maksimalna veličina segmenta (*Maximum Segment Size - MSS*) se određuje prilikom uspostavljanja veze i kasnije se ne može menjati, a podrazumevana vrednost na većini operativnih sistema iznosi 536 bajtova [26]. Po prijemu u odredištu, segmenti se objedinjuju u originalnom redosledu, rekonstruišući poslati niz bajtova, koji se predaju prijemnoj aplikaciji.

Zbog obezbeđivanja pouzdanog prenosa, segmenti moraju da se jedinstveno identifikuju i to na način koji će omogućiti očuvanje njihovog redosleda. To se postiže kroz atribut koji se naziva ***Sequence Number***, a koji se pridružuje svakom bajtu i odnosi se na njegovu relativnu poziciju u nizu bajtova aplikativnih podataka. Relativna pozicije se ne računa od nultog ili prvog bajta, već od određene inicijalne vrednosti koja se slučajno uspostavlja i razmenjuje prilikom uspostavljanja veze. Na ovaj način, prvi bajt u nizu podataka ima relativnu poziciju koja odgovara inicijalnoj uspostavljenoj vrednosti za *Sequence Number*, a koja se uvećava za po jedan za svaki naredni bajt. Identifikacija segmenta odgovara *Sequence Number* vrednosti njegovog početnog bajta (Slika 11.8). Razlog zašto se uvodi slučajno izabrana početna vrednost za *Sequence Number* je da se izbegne eventualna zabuna oko tumačenja podataka dve vremenski bliske komunikacione sesije.

Prilikom slanja segmenta, odgovarajuća *Sequence Number* vrednost se navodi u posebnom polju TCP zaglavlja. Ovim mehanizmom prijemna strana može da odredi poziciju svakog segmenta, čak iako oni ne pristižu u originalnom redosledu, čime se obezbeđuje uspešno objedinjavanje podataka za predaju prijemnoj aplikaciji.

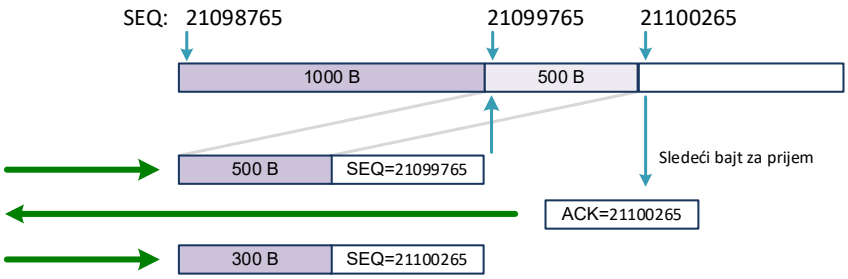


Slika 11.8. Sequence Number *za proizvoljni bajt u nizu i odgovarajuće segmente, pri inicijalnoj vrednosti od 21098765*

Sequence Number vrednosti se takođe koriste za obezbeđivanje pouzdanog prenosa podataka uvođenjem eksplicitne potvrde primljenih segmenata, kroz atribut koji se naziva ***Acknowledge Number***. Koristeći istu relativnu numeraciju bajtova, uvećanu za inicijalnu vrednost *Sequence Number*, vrednost *Acknowledge Number* atributa predstavlja poziciju narednog bajta podataka koji se očekuje na prijemnoj strani, pod uslovom da su svi prethodni bajtovi u potpunosti primljeni.

Sequence Number i *Acknowledge Number* sadrže uparene vrednosti, koje se odnose na isti niz bajtova i njihove relativne pozicije, ali posmatrano na suprotnim stranama komunikacije (Slika 11.9):

- *Sequence Number* vrednost se određuje pri slanju i prenosi u zaglavlju zajedno sa podacima na koje se odnosi
- *Acknowledge Number* vrednost se određuje pri prijemu i prenosi se u suprotnom smeru zahtevajući naredne podatke, čime se potvrđuje da su svi prethodni podaci primljeni.

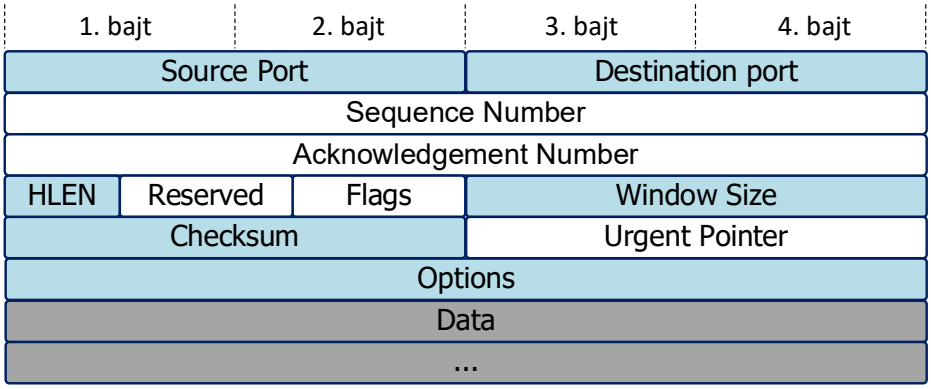


Slika 11.9. Acknowledge Number *za potvrdu prijema svih prethodnih bajtova*

TCP zaglavlje sadrži i polje sa određenim kontrolnim flegovima, koji se po potrebi koriste da odrede značenje poruke ili validnost pojedinih polja. Značenje pojedinih flegova će se prikazati nešto kasnije u ovom poglavlju, kroz detaljan opis dodatnih funkcija koje sprovodi TCP protokol.

Format TCP zaglavlja prikazuje Slika 11.10. Osim polja koja sadrže izvorišni i odredišni broj porta (*Source port*, *Destination port*), kao i *Checksum* polja, koje ima isti smisao kao i kod UDP protokola, TCP zaglavlje sadrži i dodatna polja, kao što su *Sequence Number*, *Acknowledge*

Number, polje sa flegovima (*Flags*), kao i polje *Windows Size*, čija će značenja takođe biti objašnjena kroz opis funkcija koje obezbeđuje TCP protokol.



Slika 11.10. Format TCP zaglavlja

Uspostavljanje veze

Da bi se podaci prenosili preko TCP protokola, neophodno je uspostaviti vezu kroz eksplicitni dogovor obe strane, što predstavlja tzv. *connection oriented* režim komunikacije. U okviru jedne komunikacione sesije prenos podataka se sprovodi u oba smera, koji se tretiraju nezavisno, pa je i vezu potrebno uspostaviti u oba smera.

Uspostavljanje veze u jednom smeru podrazumeva da se generiše i šalje inicijalna *Sequence Number* vrednost, za koju se od druge strane dobija potvrda u vidu odgovarajuće *Acknowledge Number* vrednosti. Ovaj proces se sprovodi u dva koraka, odnosno razmenom dve poruke (Slika 11.11):

- **1. Korak**

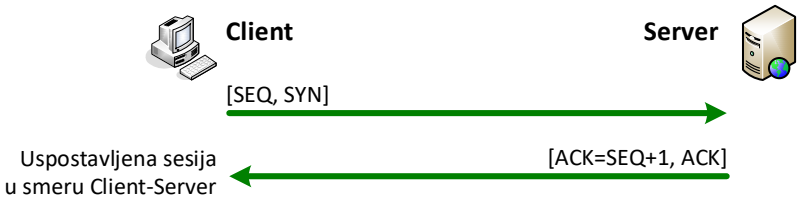
  Klijent šalje inicijalni zahtev za uspostavljanje veze prema serveru o obliku poruke čije zaglavlje sadrži sledeće atribute:

  - Slučajno izabrana *Sequence Number* vrednost, koja će da se koristi kao referentna početna vrednost za niz bajtova koji treba da se šalju.
  - **SYN** fleg (*Synchronization*), koji označava zahtev za uspostavljanje veze, odnosno zahtev za sinhronizaciju inicijalne *Sequence Number* vrednosti.

- **2. Korak**

  Kao odgovor na primljeni zahtev, server šalje poruku čije zaglavlje sadrži sledeće atribute:

  - *Acknowledge Number* vrednost koja je za broj jedan uvećana u odnosu na primljenu *Sequence Number* vrednost (ako se uz inicijalnu poruku nisu preneli aplikativni podaci), čime se potvrđuje prijem zahteva.
  - **ACK** fleg (*Acknowledge*), koji označava da *Acknowledge Number* polje sadrži validnu vrednost.



Slika 11.11. Uspostavljanje veze u jednom smeru

Proces uspostavlja veza u drugom smeru od servera prema klijentu je isti, uz nezavisnu i novu *Sequence Number* vrednost, ali je uobičajeno da se veze u oba smera uspostavljaju u tri objedinjena koraka, tzv. ***three-way handshake***, tako što se zahtev servera za uspostavljanje veze prema klijentu spaja sa odgovorom servera na inicijalnih zahtev klijenta (Slika 11.12):

- **1. Korak**

  Klijent šalje inicijalnih zahtev za uspostavljanje veze prema serveru, koji sadrži inicijalnu *Sequence Number* vrednost i postavljen SYN fleg

- **2. Korak**

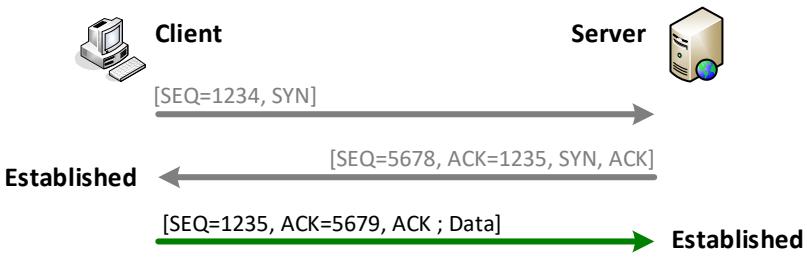
  Server prihvata zahtev od klijenta, i šalje mu novu poruku koja sadrži *Acknowledge Number* vrednost uvećanu za 1 u odnosu na primljenu *Sequence Number* vrednost, kao i ACK fleg.

  U istoj poruci server šalje zahtev prema klijentu tako što postavlja *Sequence Number* vrednost koja je različita i nezavisna od primljene vrednosti, kao i SYN fleg.

- **3. Korak**

  Klijent prihvata zahtev od servera, i šalje mu novu poruku koja sadrži *Acknowledge Number* vrednost uvećanu za 1 u odnosu na primljenu *Sequence Number* vrednost, kao i ACK fleg.

  U istoj poruci klijent može da šalje i podatke prema serveru, i to od pozicije koja odgovara prethodno primljenoj *Acknowledge Number* vrednosti, a koja se postavlja u *Sequence Number* polje.



*Slika 11.12. Uspostavljanje veze u oba smera (*three-way handshake*)*

Napomenimo da uz zahtev za uspostavljanje veze u bilo kom smeru mogu da se istovremeno prenose i početni aplikativni podaci. U tom slučaju *Acknowledge Number* vrednost u odgovoru se ne uvećava za 1, već za dužinu podataka izraženu u bajtovima. Ovi podaci se neće predati aplikaciji dok se ne završi ceo proces uspostavljanja veze u oba smera, kao bi se aplikacija zaštitila od malicioznih pokušaja „lažnog predstavljanja“ postavljanjem pogrešne izvorišne IP adrese (*spoofed*) i podmetanjem određenog sadržaja.

Zbog *connection oriented* režima prenosa podataka, koncept soketa se kod TCP protokola može dodatno interpretirati i kao „ulaz u aplikaciju samo za uparenu stranu“. U tom kontekstu se soket posmatra kao uređena petorka sastavljena od izvorišne IP adrese i TCP porta, odredišne IP adrese i TCP porta i identifikacije TCP protokola, koji ima vrednost 6.

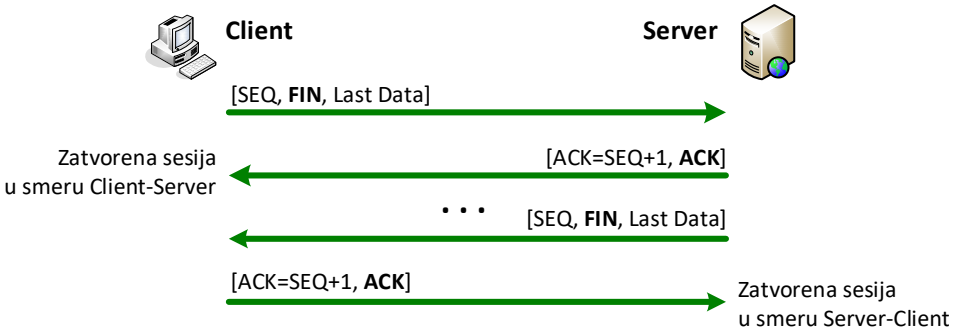
Kada se sprovede prenos svih aplikativnih podataka, vezu je potrebno raskinuti, što je takođe proces međusobnog dogovora koji se sprovodi u dva koraka i to nezavisno u oba smera (Slika 11.13):

- **1. Korak**

  Sa slanjem poslednjih aplikativnih podataka postavlja se **FIN** fleg (***Finish***) koji označava zahtev za prekid veze.

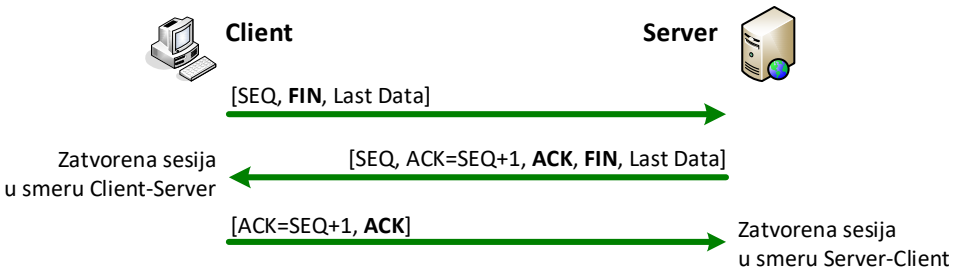
- **2. Korak**

  Potvrđuje se prijem prethodno dobijenih podataka preko odgovarajuće *Acknowledge Number* vrednosti i postavljenim ACK flegom.



Slika 11.13. Raskidanje veze nezavisno u oba smera

Za razliku od procesa uspostavljanja veze koji uvek inicira klijent, raskidanje veze u jednom smeru može da inicira bilo koja strana. Druga strana može da nastavi sa slanjem preostalih podataka još neko vreme, ali i da zahteva iniciranje veze u suprotnom smeru. To se sprovodi postavljanjem FIN flega u drugom koraku, uz potvrdu prethodnog zahteva za raskidanjem veze i eventualno slanje preostalih podataka (Slika 11.14).



Slika 11.14. Raskidanje veze u oba smera u tri koraka

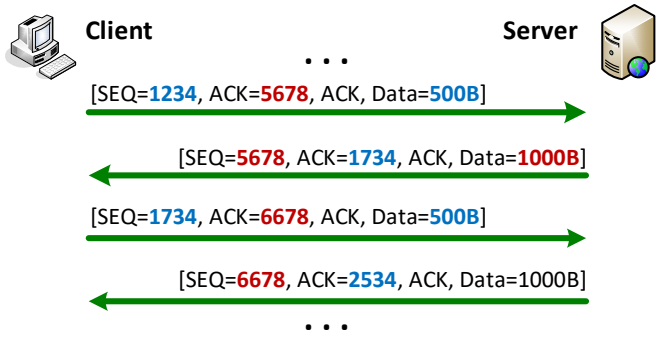
Veza se može raskinuti i na „grublji“ način, bez potrebe za slanjem potvrde od druge strane. Ovo se postiže slanjem **RST** flega (*Reset*), kada se veza, ili proces uspostavljanja veze, momentalno prekida. Tipičan slučaj korišćenja RST flega je da se naznači da server odbija zahtev za uspostavljanje veze od klijenta, a razlog za to može da bude da ne postoji port koji je naznačen u TCP poruci. Ovo ponašanje je opciono i često se ne primenjuje, kada server

jednostavno ignoriše zahtev klijenta bez slanja bilo kakve poruke u odgovoru, čime se izbegava dodatno opterećenje servera u slučaju DoS/DDoS napada.

Pouzdan prenos

Za mnoge aplikacije pouzdan prenos podataka (*Reliable transmission*) je i najvažnija funkcija koju sprovodi TCP protokol. Treba naglasiti da pouzdan prenos, u potpunom smislu, ne znači da će podaci sigurno biti isporučeni, već da će isporučilac dobiti potvrdu da su poslati podaci primljeni na drugoj strani. Za ovu namenu se koristi *Sequence Number* i *Acknowledge Number* mehanizam i to odvojeno u oba smera.

Nakon uspostavljanja veze i sinhronizacije inicijalne *Sequence Number* vrednosti, za poslate podatke se očekuje *Acknowledge Number* koji garantuje da su uspešno primljeni svi bajtovi do naznačene relativne pozicije. Na primeru koji prikazuje Slika 11.15, za prenos 500 bajtova uz *Sequence Number* vrednost od 1234 dobija se poruka koja sadrži *Acknowledge Number* vrednost od 1734 (dobijeno kao 1234+500), što potvrđuje da je prijemna strana dobila bez greške svih 500 bajtova (kao i sve bajtove do tada), i da je spremna da nadalje prima bajtove od relativne pozicije 1734. Isti, ali nezavisan proces se sprovodi u drugom smeru, sa različitim *Sequence Number* i *Acknowledge Number* vrednostima.



Slika 11.15. Pouzdan prenos ostvaren preko potvrde prijema podataka

TCP je naravno nemoćan ako se javi greška pri prenosu na putu do odredišta. U slučaju da je greška prisutna duže vreme (npr. prekid linka na putanji do druge strane), TCP će da obavesti aplikaciju da je konekcija u prekidu (*connection error*). Sa druge strane, nisu retke ni pojave da se pojedini paketi unište pri prenosu (npr. usled lošeg kvaliteta veze, opterećenja linka ili rutera, grešaka u konfigurisanju rutiranja itd.). Bilo da se uništi paket koji prenosi aplikativne podatke ili paket koji prenosi potvrdu u suprotnom smeru, smatraće se da ovi podaci nisu isporučeni i poslaće se ponovo (*retransmission*).

Izostanak paketa potvrde se može detektovati jedino čekanjem određeni vremenski period (*timeout*), što TCP sprovodi za svaki poslati segment. Ovo vreme treba da bude što manje, kako bi se što pre ponovo poslali podaci, ali ipak dovoljno veliko da se pruži šansa odredištu da odgovori. Problem je što TCP treba da se prilagodi komunikaciji i sa bliskim odredištima, za koje se vreme prenosa izražava u milisekundama, kao i sa udaljenim odredištima, čiji je odziv ponekad i više od sekunde. Takođe, na vreme potrebno za dobijanje potvrde utiče i trenutno opterećenje i mreže i odredišnog uređaja, koje može da značajno varira tokom komunikacije.

TCP ovo rešava tako što prati vreme koje protekne od slanja svakog segmenta do trenutka prijema potvrde, odnosno vreme komunikacije u oba smera (*Round Trip Time* – RTT) i na osnovu toga procenjuje očekivano vreme dolaska potvrde, koje se uveća za određeni faktor tolerancije. Ovo se sprovodi metodom predikcije bazirane na svim prethodnim vrednostima, ali sa većim težinskim faktorom za skorašnje vrednosti u odnosnu na vrednosti koje su se desile ranije, tzv. *Exponential Moving Averages* (EMA), po sledećoj formuli:

$$SRTT_{n+1} = \alpha * SRTT_n + (1 - \alpha)\, RTT_n$$

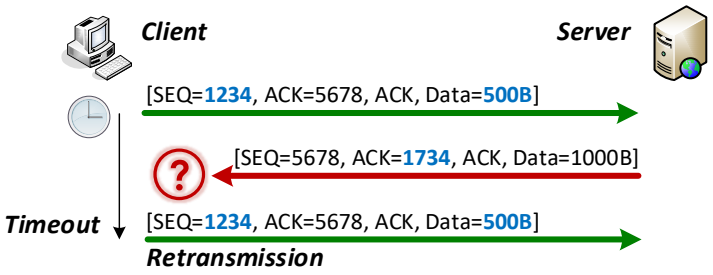
gde je *n* posmatrani trenutak, RTT_n poslednja izmerena vrednost za RTT, $SRTT_n$ poslednja procena za posmatrani trenutak (*Smoothed Round Trip Time*), dok je α težinski faktor u intervalu od 0 do 1, na osnovu čega se izračunava $SRTT_{n+1}$ kao procena naredne vrednosti. Formula je rekurzivnog karaktera, jer zavisi samo od prethodne procenjene vrednosti i nove izmerene vrednosti RTT. Ako se formula razvije za više poslednjih intervala, primećuje se da svaka prethodna procenjena vrednost utiče na novu procenu i to kao eksponencijalna vrednost gde je osnova faktora α. Ako je faktor α bliži vrednosti nula, tada je uticaj prethodno procenjenih vrednosti manji, pa se procena brže prilagođava poslednjoj stvarno izmerenoj vrednosti. Nasuprot tome, ako je faktor α bliži vrednosti 1, tada poslednja izmerena vrednost ima mali uticaj i procenjene vrednosti će sporije da prate trend promena (*smoothed*).

Interval čekanja na potvrdu poslatog segmenta RTO (*retransmission timeout*) se postavlja da bude srazmerno veći od procenjene vrednosti i to za faktor β:

$$RTO_{n+1} = \beta * SRTT_{n+1}$$

Faktor β tipično ima vrednost 2.

Ako za vreme *RTO* ne stigne potvrda o uspešnom prijemu segmenta, segment se smatra neisporučenim i poslaće se ponovo kroz proces retransmisije (*retransmission*).



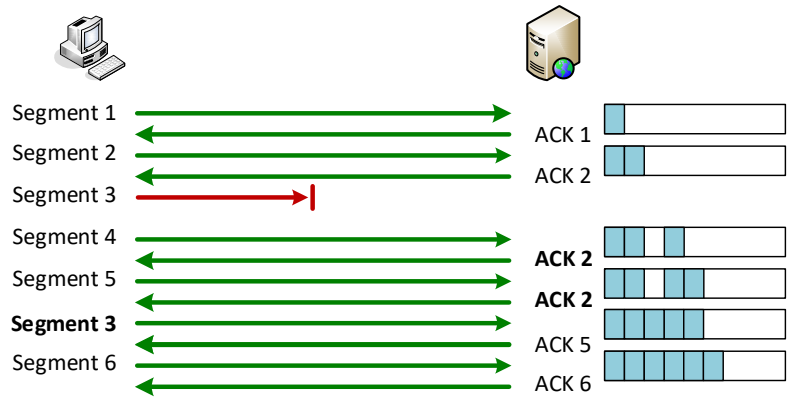
Slika 11.16. Retransmisija segmenta za koji nije dobijena potvrda tokom određenog perioda (RTO)

Napomenimo da izostanak potvrde prijema tokom RTO perioda ne znači nužno da segment nije isporučen, već razlog može da bude i da je poruka potvrde uništena na putu u suprotnom smeru. Takođe, poruka sa potvrdom može da bude samo usporena i da pristigne nakon isteka RTO perioda i retransmisije segmenta. U tom slučaju ponovo poslati segment neće napraviti problem, jer će on biti prepoznat kao duplikat i odbaciće se na prijemnoj strani. Ipak, naknadno pristigla potvrda je problematična za merenje RTT i računanje narednog RTO intervala, jer se ne može jednoznačno odrediti da li se ona odnosi na prvobitno poslati segment ili retransmitovani segment. Stoga se za retransmitovane segmente ne sprovodi merenje RTT i ažuriranje SRTT, odnosno RTO intervala, što je poznato kao Karnov algoritam [27][28].

Posmatrajmo sada kako na prijemnoj strani funkcioniše mehanizam pouzdanog prenosa odnosno oporavka ako neki segment izostane. Najpre istaknimo da pošiljalac ne čeka potvrdu za svaki poslati segment da bi poslao naredni segment. Drugim rečima, pošiljalac sukcesivno šalje više segmenata, a potvrde za njih naknadno pristižu, što reguliše funkcija kontrole toka. U slučaju gubitka jednog segmenta, naredni segmenti mogu da pristignu, a preko *Sequence Number* vrednosti rekonstruiše se originalna pozicija tih segmenata. Segment koji nije pristigao do tada stvara rupu u originalnom nizu bajtova (Slika 11.17).

Podsetimo se da se *Acknowledge Number* vrednost odnosi na relativnu poziciju bajta koji se očekuje da se primi u kontinuitetu, što podrazumeva da su primljeni svi prethodni bajtovi. Takođe, pravilo je da se potvrda, odnosno *Acknowledge Number* vrednost, šalje samo kada se primi paket sa podacima, što može da bude bilo koji segment. U ovom primeru segment broj 3 je uništen na putu, ali je pristigao naredni segment broj 4. To će da izazove prazninu u primljenim podacima i slanje potvrde sa *Acknowledge Number* vrednosti koja se odnosi na očekivane podatke iz izgubljenog segmenta broj 3. Ova *Acknowledge Number* vrednost je već poslata kada je pristigao segment broj 2 (ACK 2). Isto se dešava i nakon prijema segmenta broj 5 – podaci su prihvaćeni, ali se potvrđuje segment broj 2, uz očekivanje podataka iz segmenta broj 3. Ova pojava se naziva „višestruke potvrde“ (*duplicate acknowledge*).

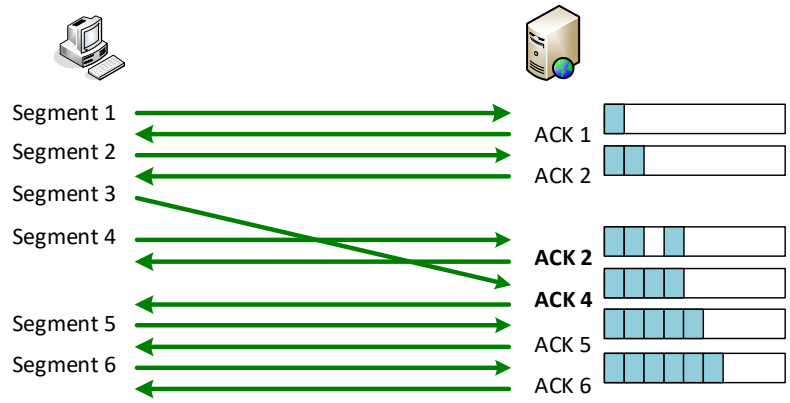
Nakon retransmisije segmenta broj 3 na prijemnoj strani se popunjava praznina u nizu primljenih bajtova, nakon čega se šalje *Acknowledge Number* čija vrednost odgovara poziciji narednog bajta u nizu, što odgovara potvrdi poslednjih podataka prenetih u segmentu broj 6 (ACK 6).



Slika 11.17. Prijem narednih segmenata i slanje potvrde u slučaju gubitka jednog segmenta

Održavanje redosleda podataka

Prethodni primer ujedno demonstrira i funkciju održavanja redosleda podataka (*Ordered data reconstruction*). Promena redosleda ne nastaje samo pri gubitku pojedinih segmenata, već može nastati i kao posledica putovanja segmenata po različitim putanjama. U svakom slučaju, prijemna strana rekonstruiše njihov originalni redosled koristeći *Sequence Number* mehanizam (Slika 11.18).



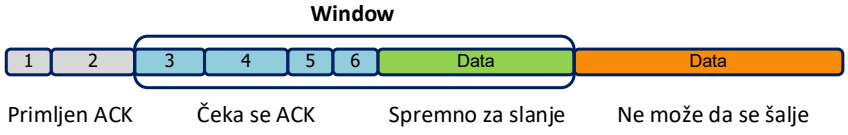
Slika 11.18. Rekonstrukcija poretka pristiglih segmenata

Kontrola toka

Protokoli na transportnom nivou direktno komuniciraju između dva krajnja uređaja bez međutačaka na putu u mreži, pa se može reći da su oni bliži aplikativnom sloju nego mrežnoj infrastrukturi, koja za njih ostaje nepoznata apstrakcija. Sa druge strane TCP protokol treba da obezbedi prenos velike količine kontinuiranih podataka na što efikasniji način, što zavisi od mrežnih performansi, kao što su protok i kašnjenje paketa. Stoga je neophodno da TCP protokol dinamički prilagodi intenzitet prenosa podataka u zavisnosti od trenutnog opterećenja mreže i mogućnosti prijemne strane da prihvati podatke, što se postiže kroz funkciju kontrole toka (*Flow control*).

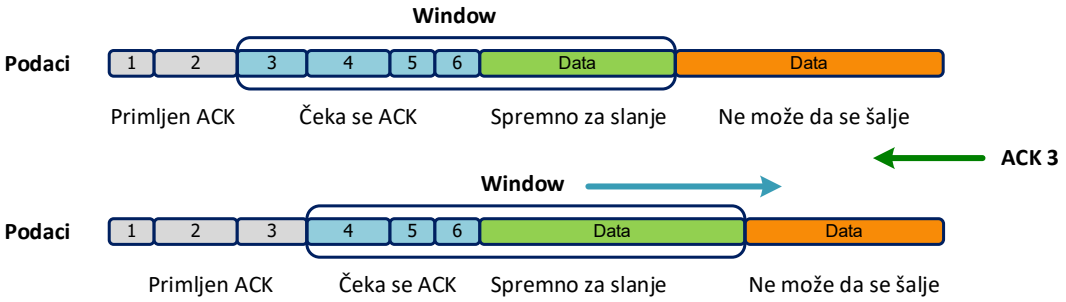
Podaci se od aplikacije dobijaju kao niz bajtova, koje je potrebno preneti određenim intenzitetom. Ako je intenzitet slanja suviše mali, komunikacija je neefikasna. Sa druge strane, preveliki intenzitet slanja može izazvati gubitak paketa, retransmisiju segmenata, što takođe dovodi do neefikasnog prenosa. TCP stoga treba da se dinamički prilagodi kapacitetu mreže, uključujući i mogućnosti prijemne strane, tako što se odrediti optimalni intenzitet slanja podataka.

Kontrola toka se postiže kroz mehanizam prozora (*window*), koji se sprovodi na strani koja šalje podatke, ali pod inicijalnim uslovima koje određuje strana koja treba da prima podatke. Prozor u ovom kontekstu predstavlja opseg bajtova u nizu aplikativnih podataka koji se šalju (Slika 11.19). U njemu se nalaze bajtovi koji su poslati, ali se još uvek čeka na njihovu potvrdu, kao i bajtovi koji još uvek nisu poslati, što se očekuje u narednim trenucima. Na ovaj način se omogućava slanje veće količine bajtova u kontinuitetu, podeljeno na sukcesivne segmente za koje naknadno pristižu potvrde. U nizu aplikativnih podataka ispred prozora se nalaze bajtovi koji su poslati i potvrđeni, dok se posle prozora nalaze podaci koji ne mogu da se šalju, čak iako su poslati svi podaci iz prozora, sve dok se prozor ne pomeri do njih.



Slika 11.19. Mehanizam prozora nad nizom bajtova aplikativnih podataka

Kada za najranije podatke u prozoru pristigunu potvrde o prijemu, oni se uklanjau iz prozora tako što se prozor pomera ka novim podacima (Slika 11.20), zbog čega se ovaj mehanizam naziva i pomerajući prozor (*sliding window*).

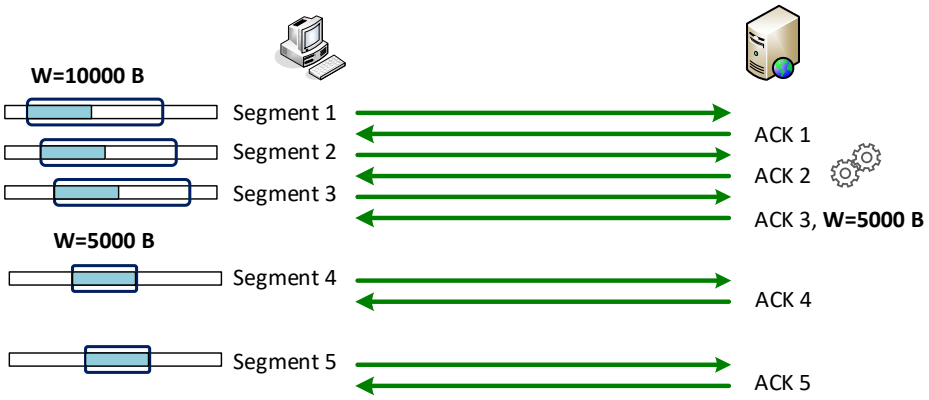


Slika 11.20. Pomeranje prozora nakon pristizanja potvrda za poslate podatke

Veličina prozora određuje intenzitet slanja podataka. Ako je prozor suviše velik, podaci će brzo da se šalju, što može izazvati zagušenje, gubitak paketa i pad efikasnosti. Sa druge strane ako je prozor mali, on će brzo da se popuni, čekajući prijem potvrda, nakon čega će da se pomere i ponovo brzo popuni uz novo čekanje, što takođe dovodi do pada efikasnosti.

Veličinu prozora definiše prijemna strana prilikom uspostavljanja veze prenošenjem odgovarajuće vrednosti u polju *Window Size* u TCP zaglavlju. Ovo polje je veličine 16 bajtova, što se kasnije pokazalo nedovoljnim za velike brzine prenosa, koje su kasnije postale uobičajene, a koje zahtevaju veće veličine prozora. Stoga je naknadno uveden parametar *TCP Window Scale*, koji se prenosi kao opciono polje u TCP zaglavlju i predstavlja multiplikativni faktor koji se primenjuje kako bi se povećala *Window Size* vrednost [29]. Tačnije, *TCP Window Scale* predstavlja eksponent broja 2, u opsegu vrednosti od 0 do 14, a koji se primenjuje za binarno šiftovanje vrednosti *Window Size*. Time se postiže da se veličina prozora može maksimalno povećati za 2^{14} puta, odnosno da se postigne maksimalna veličina prozora od 2^{30} bajtova. Na ovaj način pošiljalac u zahtevu za uspostavljanjem veze može opciono da pošalje i *TCP Window Scale* vrednost, dok u odgovoru primalac uvek šalje odgovarajuću *Window Size* vrednost. Veličina prozora koja se uspostavlja na strani pošiljaoca predstavlja srazmerno uvećanu *Window Size* vrednost, ako se koristi *TCP Window Scale*.

Za razliku od fiksno uspostavljene veličine segmenata, *Window Size* vrednost se može menjati tokom komunikacione sesije. Ako primalac detektuje preveliki intenzitet prijemnih podataka, on može zahtevati od pošiljaoca da smanji veličinu prozora, čime se efektivno postiže manji intenzitet slanja podataka (Slika 11.21). Zbog navedene osobine koristi se i termin dinamički prozor (*dynamic window*).



Slika 11.21. Dinamički prozor – promena veličine prozora tokom komunikacione sesije

Kontrola zagušenja

Dinamički prozor ipak nije dovoljno fleksibilan metod za agilno prilagođavanje protoka slanja podataka prema stvarnom ograničenju koje nameće mreža, uzimajući u obzir i trenutna opterećenja. TCP protokol je naknadno uveo dodatne mehanizme još fleksibilnijeg dinamičkog uspostavljanja efektivne veličine prozora [30], što je kasnije objedinio pod naziv kontrola zagušenja (*congestion control*) [31], a koja implementira mehanizme koji se opisuju u nastavku.

Slow Start

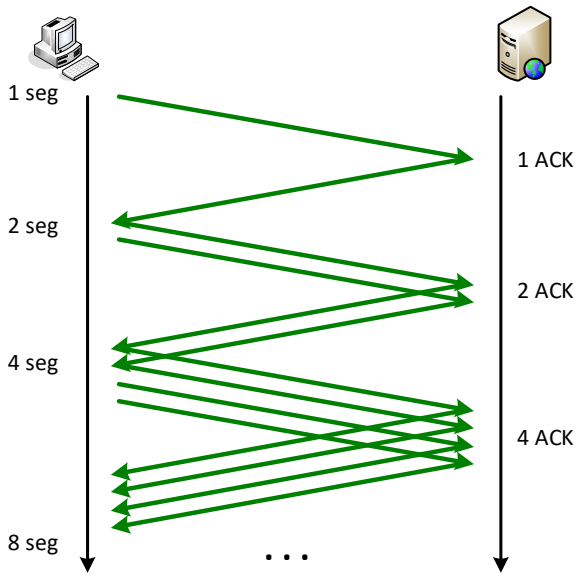
Prilikom uspostavljanja veze prijemna strana, prema svojim performansama, definiše inicijalnu veličinu prozora, koja zapravo predstavlja i najveću moguću vrednost, tzv. *Advertised Window* (AW). Da bi se brzina slanja prilagodila stvarnim performansama mreže, ali i trenutnom opterećenju, stvarna veličina prozora, tzv. *Congestion Window* (CW), se postepeno povećava do maksimalne vrednosti (AW), na sledeći način:

- Početna vrednost za CW odgovara maksimalnoj veličini jednog segmenta (MSS).
- Vrednost CW se povećava za MSS vrednost pri dolasku svake naredne potvrde (ACK).
- CW se ne povećava kada dostigne maksimalnu vrednost (AW).
- U slučaju gubitka paketa, što se detektuje izostankom potvrde za odgovarajući *timeout* period, CW se resetuje na inicijalnu vrednost za jedan segment, nakon čega se ponovo uspostavlja proces postepenog povećanja.

Primenom ove metode postiže se režim slanja paketa koji ilustruje Slika 11.22, gde je zbog jednostavnosti veličina CW izražena u segmentima, tačnije MSS vrednosti.

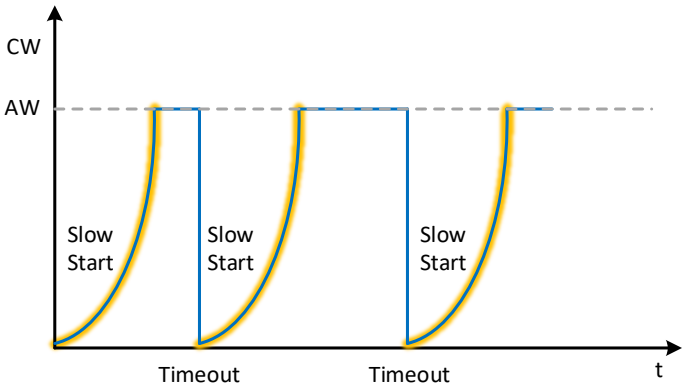
- Inicijalna vrednost CW je 1, pa se šalje samo jedan segment, za koji se čeka potvrda.
- Za poslati segment se dobija potvrda, vrednost CW se povećava na 2, pa se šalju 2 segmenta.
- Za poslate segmente se dobijaju dve potvrde, vrednost CW se povećava sa 2 na 4, pa se šalju 4 segmenta.
- U svakom narednom koraku pristižu potvrde za poslate segmente, čime se vrednost CW duplira, što predstavlja eksponencijalni rast. U realnosti, po prijemnu bliskih segmenata u kratkom vremenu, primalac može jednom porukom da potvrdi više segmenata. U proseku

se tipično istovremeno potvrđuje oko dva segmenta, pa je porast veličine CW nešto sporiji, ali i dalje eksponencijalan.



*Slika 11.22. Slow Start – povećanje stvarne veličine prozora (*Congestion Window*) na osnovu broja primljenih potvrda*

Iako se metoda zove *Slow Start*, vrednost CW raste eksponencijalno, ali ipak treba određeno vreme da se dostigne maksimalna vrednost od AW. Problem je što gubitak makar jednog paketa uzrokuje resetovanje na inicijalnu najmanju vrednosti i ponovni rast do maksimalne vrednosti (Slika 11.23).

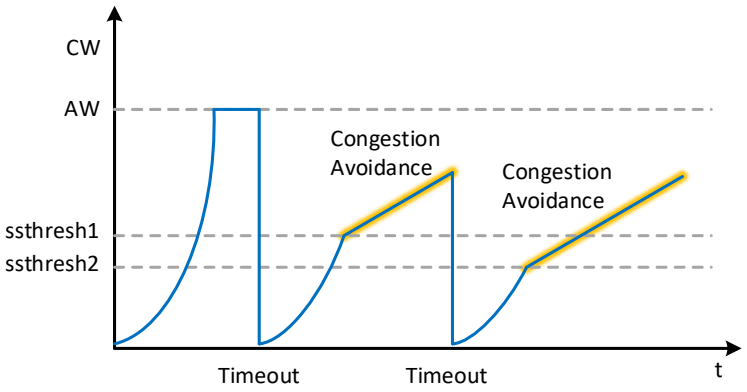


*Slika 11.23. Slow Start – postepeno povećanje stvarne veličine prozora (*Congestion Window*) i naglo smanjenje u slučaju gubitka paketa*

Congestion Avoidance

Gubitak paketa je indikacija prebrzog slanja podataka, što se usporava smanjenjem CW na minimalnu vrednost i ponovnim sprovođenjem *Slow Start* procesa. Kako bi se smanjila verovatnoća ponovnog gubitka paketa i brzo dostizanje maksimalne vrednosti, eksponencijalno povećanje CW će preći u linearno, nakon što CW dostigne određeni nivo, tzv. *Slow Start Threshold Size* (*ssthresh*). Ova vrednost se određuje kao polovina vrednosti CW u trenutku gubitka paketa (*ssthresh = CW/2*). Linearno povećanje CW vrednosti sporije će

povećavati intenzitet slanja podataka, čime se nastoji izbegavanje zagušenja, što se naziva *Congestion Avoidance* (Slika 11.24).



Slika 11.24. Congestion Avoidance – *linearno povećanje stvarne veličine prozora (*Congestion Window*) nakon dostizanja određenog nivoa (ssthresh)*

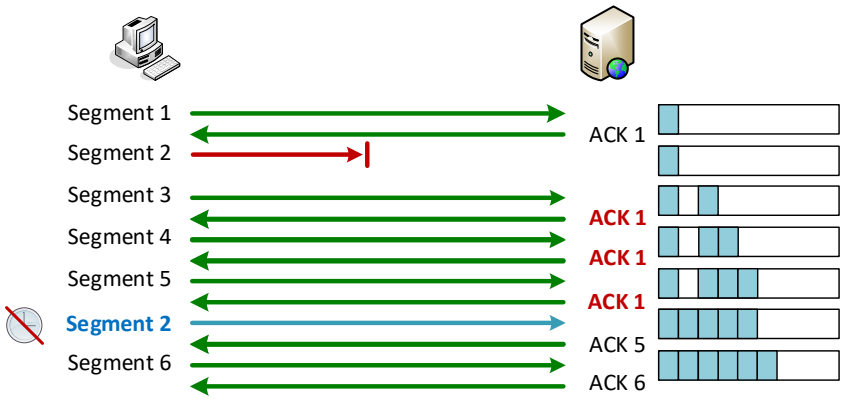
U slučaju ponovnog gubitka paketa i pre nego što se dostigne maksimalna vrednost, nova *ssthresh* vrednost će biti još manja i ranije će da se uđe u *Congestion Avoidance* proces linearnog povećanja.

Fast Retransmit

Gubitak paketa se generalno detektuje putem *timeout* mehanizma. Ipak, sporadičan gubitak samo jednog paketa, uz pristizanje narednih paketa, izazvaće ponavljanje potvrda za podatke iz poslednjeg segmenta primljenog u kontinuitetu, tzv. duple ili višestruke potvrde (*duplicate acknowledgment*), obično i pre isteka *timeout* perioda (Slika 11.25).

Jedna ili dve višestruke potvrde mogu da znače i da je došlo do promene redosleda paketa. Ipak, ako se primi i treća višestruka potvrda smatraće se da je paket zaista izgubljen, ali da su pristigli ostali paketi. Tada će se ponovo poslati odgovarajući segment iako nije istekao *timeout* interval.

Opisana metoda ubrzava nadoknadu izgubljenog segmenta, što se naziva *Fast Retransmit*.

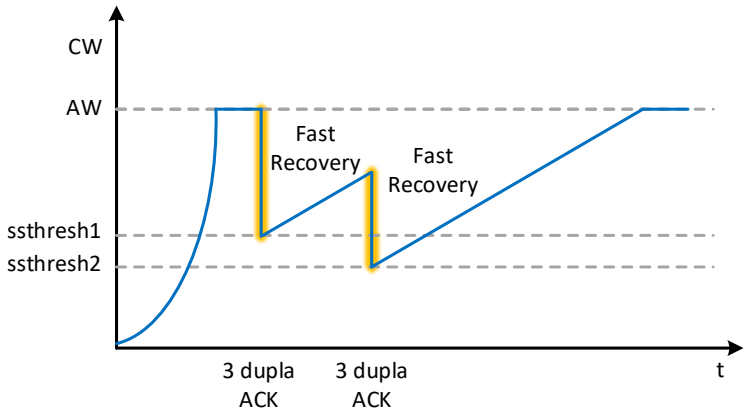


Slika 11.25. Fast Retransmit – *ponovno slanje segmenta nakon prijema treće ponovljene potvrde*

Fast Recovery

Nastanak *Fast Retransmit* situacije znači da je jedan segment izgubljen, ali da su najmanje naredna tri segmenta ispravno pristigla u odredište. Drugim rečima, veza nije u prekidu, a ni potpuno zagušena, pa je neopravdano drastično smanjenje CW na najmanju vrednost i ulazak u *Slow Start* proces.

Stoga se uspostavlja tzv. brzi oporavak, odnosno *Fast Recovery* princip, kada se CW vrednost smanjuje na polovinu od trenutne vrednosti (*ssthresh*), izbegava *Slow Start* faza i odmah ulazi u *Congestion Avoidance* proces linearnog povećanja CW vrednosti (Slika 11.26).

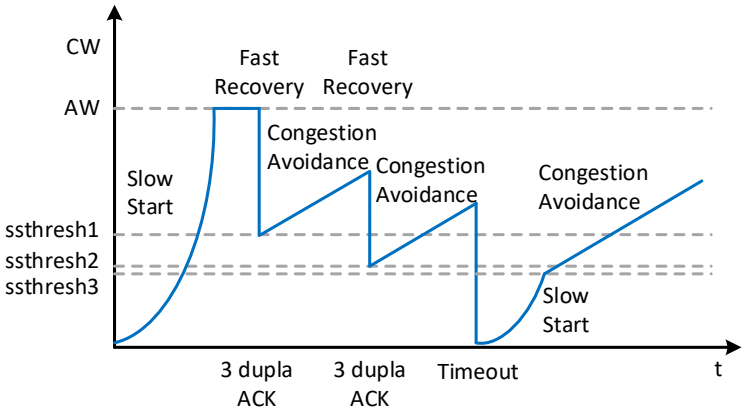


Slika 11.26. Fast Recovery – *izbegavanje* Slow Start *faze i ulazak u* Fast Recovery *fazu nakon* Fast Retransmit *procesa*

Na ovaj način se veličina efektivnog prozora smanjuje na polovinu, slanje podataka se usporava, a zatim postepeno povećava, kao bi se izbeglo eventualno zagušenje.

U idealnoj situaciji, prozor će dostići maksimalnu veličinu što će ostvariti veliki protok podataka. Ako mreža i prijemna strana mogu da podnesu ovakav intenzitet saobraćaja, ostvariće se velika protočnost uz isporuku svih paketa.

U realnosti, pogotovo na većim rastojanjima na Internetu, kada na pojedinim delovima puta dolazi do agregacije velike količine saobraćaja, a time i do potencijalnog zagušenja, gubitak paketa nije retka pojava. Sumarno posmatrano, TCP se prilagođava na ovakve uslove primenom kontrole zagušenja, koja podrazumeva sledeća četiri mehanizma: *Slow Start, Congestion Avoidance, Fast Retransmit* i *Fast Recovery.* U zavisnosti od uslova pod kojima dolazi do gubitka pojedinih, ali ne svih, paketa, veličina efektivnog prozora (CW) može da se značajno menja, kao što ilustruje Slika 11.27. Navedeni mehanizmi nisu idealni, ali su se ipak održali kao podrazumevani mehanizmi u implementacijama TCP protokola na svim standardnim operativnim sistemima.

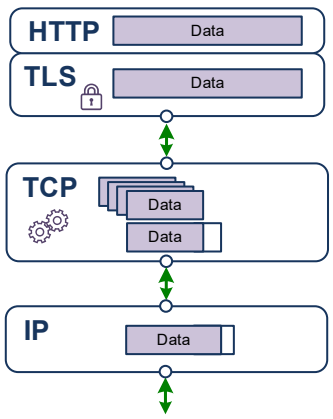


Slika 11.27. Kontrola zagušenja

11.4 QUIC protokol

TCP i UDP protokoli, zajedno sa IP protokolom, predstavljaju osnovne mehanizme komunikacije na Internetu. Iako su davno nastali u akademskim krugovima i eksperimentalnom projektu ARPANET mreže, njihova jednostavnost, otvorenost i efikasnost zaslužna je za prihvatanja od strane korisnika, što je dovelo do nastanka Internet mreže. Eksplozivni rast Interneta je potvrda uspeha ovih protokola, ali kada je njihovo korišćenje postalo masovno, svaki dalji razvoj je bio veoma otežan. To je osnovni razlog što su UDP i TCP protokoli, uz manja unapređenja, suštinski ostali neizmenjeni već više od 50 godina.

Za to vreme su se znatno promenile performanse mreže, način korišćenja mrežnih servisa, potrebe korisnika, ali i same aplikacije. Primena veb tehnologije je sa klasičnih veb sajtova proširena na mnoge poslovne aplikacije, koje su zahtevale dodatne funkcionalnosti, kao što je ostvarivanje sigurnosti, što se postiže proverom identiteta učesnika u komunikaciji i šifrovanjem saobraćaja. Ovi i drugi nedostajući elementi TCP/IP familije protokola su dopunjavani dodatnim međuslojevima, kao što je TLS protokol (*Transport Layer Security*), a koji se postavljaju između aplikacije i transportnog sloja (Slika 11.28).



Slika 11.28. TLS međusloj između aplikacije i transportnog sloja

Promene se odnose i na vrstu podataka koji se prenose, gde po volumenu najveći deo predstavlja video saobraćaj, koji se isporučuje preko HTTP protokola od strane pružalaca sadržaja koji imaju svoje servere širom sveta (npr. Google youtube) ili specijalizovanih mreža

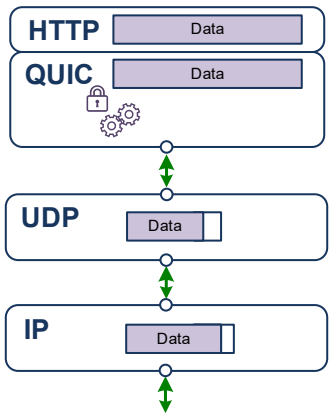
za distribuciju sadržaja (*Content Delivery Networks* - CDN). U ovim slučajevima mehanizmi uspostavljanja veze, kontrole toka i zagušenja koje primenjuje TCP protokol su postali nedovoljno neefikasni, što je dovelo do potrebe za novim rešenjima.

Industrija je često pokretač daljeg razvoja tehnologije, a u ovom slučaju kompanija *Google* je ponudila rešenje u vidu novog transportnog protokola pod nazivom QUIC, izvedeno iz *Quick UDP Internet Connections*, a koji je kasnije standardizovan od strane IETF [32].

Pri koncipiranju QUIC protokola namera je bila da se modernizuju funkcionalnosti TCP protokola i objedine sa funkcijama TSL sigurnosnog protokola, ali i dalje bazirano na postojećim servisima iz TCP/IP mrežne arhitekture. Izabrano je da se QUIC implementira unutar aplikacije (npr. veb pretraživač), a da se oslanja na transportne usluge UDP protokola (Slika 11.29). Osnovna namena je da služi za prenos veb saobraćaja putem HTTP protokola, ali ne postoje suštinska ograničenja da se koristi i za druge potrebe.

Osnovne karakteristike QUIC protokola su sledeće:

- Brzo uspostavljanje veze, što se objedinjeno sprovodi i na transportnom nivou i na nivou sigurnosnih funkcija (npr. dogovaranje parametara i razmena ključeva za šifrovanje).
- Efikasniji i fleksibilniji mehanizmi kontrole toka i zagušenja doprinose boljem protoku podataka koji se prenose putem HTTP protokola.
- Jedna uspostavljena veza se deli između većeg broja tokova za prenos podatka (*stream*), ali svaki tok sa nezavisnom kontrolom toka i zagušenja.
- Zadržavanje uspostavljene veze i komunikacija, čak i pri promeni IP adrese, npr. pri prelasku mobilnog korisnika sa *wireless* mreže na mrežu mobilne telefonije.



Slika 11.29. QUIC protokol

11.5 Praćenje stanja na transportnom nivou

Tipično korišćenje računara danas podrazumeva istovremeni rad mnogih aplikacija i procesa, što je posebno izraženo na računarima individualnih korisnika. Svi popularni operativni sistemi omogućavaju uvid u osnovne podatke o stanju konekcija na transportnom nivou. Tipična komanda ove namene je *netstat*, koja se može zadati sa različitim opcijama, od kojih opcije „*netstat -na*“ daju detaljno stanje TCP konekcija i otvorenih UDP portova, kao što prikazuje Slika 11.30.

```
C:\Users\user>netstat -na
Active Connections
  Proto  Local Address          Foreign Address        State
  TCP    147.91.15.32:49509     74.125.133.188:5228    ESTABLISHED
  TCP    147.91.15.32:49709     13.92.229.58:443       ESTABLISHED
  TCP    147.91.15.32:53532     23.6.112.112:80        TIME_WAIT
  TCP    147.91.15.32:53544     92.123.16.209:80       TIME_WAIT
  TCP    147.91.15.32:53695     62.67.193.75:443       TIME_WAIT
  TCP    147.91.15.32:53701     104.17.58.239:443      ESTABLISHED
  TCP    147.91.15.32:53712     64.68.120.41:443       ESTABLISHED
  TCP    147.91.15.32:53715     162.247.242.20:443     ESTABLISHED
  TCP    147.91.15.32:53722     13.57.4.17:443         ESTABLISHED
  TCP    147.91.15.32:53723     172.217.23.206:443     ESTABLISHED
  TCP    147.91.15.32:53725     23.7.203.117:80        TIME_WAIT
   ...
```

Slika 11.30. Rezultat izvršavanja komande „netstat -na“

Lokalni računar 147.91.15.32 je kao klijent, korišćenjem porta iz dinamičkog opsega preko 49152, ostvario mnoge TCP konekcije prema različitim udaljenim računarima, pretežno na serverskim portovima 80 i 443, koji se odnose na veb saobraćaj. Stanje TCP konekcija je ili uspostavljeno (ESTABLISHED) ili u stanju TIME_WAIT, koje označava da je konekcija završena uz slanje FIN-ACK flegova, ali se korišćeni port zadržava još neko vreme, kako bi se osiguralo da će ovaj poslednji paket pristići do odredišta. Ovo vreme keširanja završenih konekcija je dosta konzervativno i obično iznosi 4 minuta.

Opcija „*netstat -s*“ kao statističke podatke prikazuje stanja različitih brojača, kako na IP nivou, tako i za TCP i UDP protokole, što na primeru *Windows* klijenta demonstrira Slika 11.31.

```
C:\Users\user>netstat -na
…
TCP Statistics for IPv4

  Active Opens                        = 3735
  Passive Opens                       = 29
  Failed Connection Attempts          = 90
  Reset Connections                   = 870
  Current Connections                 = 49
  Segments Received                   = 2086436
  Segments Sent                       = 1188827
  Segments Retransmitted              = 895
…
UDP Statistics for IPv4

  Datagrams Received    = 495911
  No Ports              = 1452
  Receive Errors        = 5790
  Datagrams Sent        = 153999
…
```

Slika 11.31. Rezultat izvršavanja komande „netstat -s“

Prethodna komanda je korisna za utvrđivanje stanja i mogućih problema pri uspostavljanju određenih konekcija. U slučaju da se konekcija ne može uspostaviti, korisno je proveriti da li serverska strana ima otvoren odgovarajući TCP ili UDP port koji se očekuje. To se može postići komandom *nmap*, koja je na raspolaganju na *Linux* operativnim sistemima.

Komanda *nmap* poseduje mnogobrojne opcije, a uobičajeno izvršavanje pokreće veliki broj paralelnih konekcija na različite ili čak sve moguće portove. Za uspostavljene TCP konekcije prikazuje se broj porta, na koji način se dobija lista otvorenih portova na kojima server čeka za uspostavljanje komunikacije.

```
[root@user~]# nmap 147.91.111.222
Interesting ports on abc.bg.ac.rs (147.91.111.222):
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
80/tcp   open  http
443/tcp  open  https
514/tcp  open  shell
3306/tcp open  mysql
8009/tcp open  ajp13
...
```

Slika 11.32. Rezultat izvršavanja komande „nmap“

12. Aplikativni sloj

Podela mrežnih funkcija na slojeve omogućava da se implementiraju različiti protokoli različitih namena, a sve u cilju da se omogući razmena podataka između aplikacija, koje se nalaze na vrhu ove mrežne arhitekture – na aplikativnom sloju.

Komunikacija između većine aplikacija se sprovodi u klijent-server režimu. Serverske aplikacije su dostupne na unapred poznatim portovima, prema kojima klijentske aplikacije iniciraju komunikaciju koristeći dinamički izabrane portove iz raspoloživog opsega.

U nastavku poglavlja se najpre ukratko opisuju primeri popularnih mrežnih servisa i pripadajućih protokola koji ih realizuju na aplikativnom nivou, a zatim se u više detalja objašnjava rad aplikativnog servisa DNS koji je posebno bitan za rad ostalih aplikacija.

12.1 Veb servis

Jedan od najpopularnijih mrežnih servisa odnosi se na prikaz različitih vrsta sadržaja, tzv. veb servis (*web*). Nastanak ovog servisa se pripisuje *sir Timothy John Berners-Lee*, koji je kao istraživač iz organizacije CERN, 1989. godine predložio koncept funkcionisanja servisa za prikaz informacija, tada nazvan „*Information management system*“. Sistem se bazira na HTTP protokolu (*HyperText Transfer Protocol*) [33] koji omogućava prenos tekstualnih poruka koje su formatirane korišćenjem posebnog jezika za označavanje (tagovanje), tzv. HTML (*HyperText Markup Language*) [34].

Formatiranje teksta korišćenjem posebnih tagova se i ranije koristio za različite namene, a baziran na SGLM (*Standard Generalized Markup Language*) [35]. Ipak, *Timothy John Berners-Lee* je našao novu primenu u deljenju sadržaja preko mreže, tako što je realizovao serversku aplikaciju (*web server*) i odgovarajuću klijentsku aplikaciju, koja je predstavljala prvi veb pretraživač (*web browser*). Istovremeno je nastao naziv *World Wide Web* i još popularnija skraćenica „*www”*. Nešto kasnije je omogućen prenos i binarnih podataka kao posebnih objekata, što se najpre odnosilo na slike, a zatim i audio i video zapise, kao i bilo koje druge binarne podatke.

Za prenos sadržaja je bitna funkcija pouzdanosti, pa se HTTP aplikativni protokol bazira na TCP transportnom protokolu, za šta je izabran port broj 80 za serversku komponentu. Kasnije potrebe su nametnule korišćenje TLS podsloja za ostvarivanje sigurnosnih funkcija, što se označava kao *Secure HTTP*, odnosno HTTPS, sa pridruženim TCP portom 443.

Na klijentskoj strani veb pretraživači iniciraju konekciju sa veb serverom na portu 80 ili 443, i tom prilikom veza može da se sprovodi u dva moda:

- ***Non-persistent mode***, neperzistentne (kratkotrajne) konekcije – svaki klik na link u veb pretraživaču inicira zahtev za uspostavljanje nove konekcija prema veb severu, a nakon prenosa zahtevanih podataka, konekcija se raskida.
- ***Persistent mode***, perzistentne (dugotrajne) konekcije – konekcija sa veb serverom se uspostavlja na prvi zahtev i održava se aktivnom i za naredne zahteve, budući da je uobičajeno preuzimanje više sadržaja (stranica) sa jednog veb servera. Čak iako se ne koristi, konekcija se održava još neko vreme (*timeout*), nakon čega se zatvara.

Način kako veb server tretira klijente koji mu pristupaju određuje dva moda rada:

- ***Stateless*** – veb server ne pamti klijente i njihove aktivnosti,

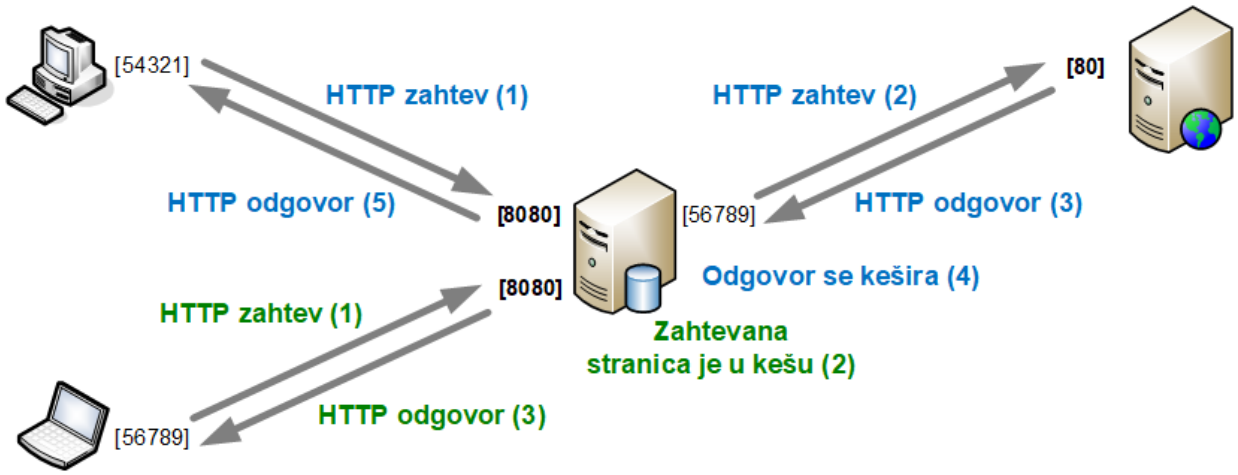
- ***Stateful*** – veb server pamti klijente (IP adrese, username itd.) i njihove aktivnosti unutar veb servera u obliku tzv. kolačića (*cookies*), što pri ponovnoj poseti klijenta omogućava njegovo prepoznavanje i prilagođavanje sadržaja (npr. prepoznavanje ulogovanog korisnika, prikazivanje prethodno uneti podataka itd.).

12.2 Proksi servis

Uspeh veb servisa je bio ogroman, što je dovelo do pojave velikog broja veb sajtova, sa bogatim i raznovrsnim sadržajima, koji su preplavili Internet i u mnogim slučajevima doveli do zagušenja eksternih linkova prema Internetu. Način korišćenja veb servisa u mrežama pojedinih organizacija je takav da više korisnika često pristupa istom sadržaju na udaljenom veb sajtu (npr. poslovne aplikacije, vesti, popularni sajtovi, sajtovi za preuzimanje softvera itd.).

Navedene pojave su iskorišćene za realizaciju posredničkog servisa za pristup udaljenim sajtovima uz lokalno keširanje prethodno preuzetih sadržaja, tzv. proksi servis (*proxy service*).

Proksi server zahteve klijenata prihvata na TCP portu 8080 (Slika 12.1, korak 1). Ako zahtevani sadržaj trenutno ne postoji u lokalnom kešu, tada proksi server u ulozi klijenta pokreće novu konekciju prema udaljenom veb serveru (korak 2). Nakon dobijanja odgovora, pristigli sadržaja najpre kešira (korak 4), a zatim se isti prosleđuje klijentu koji je to inicijalno zahtevao (korak 5).



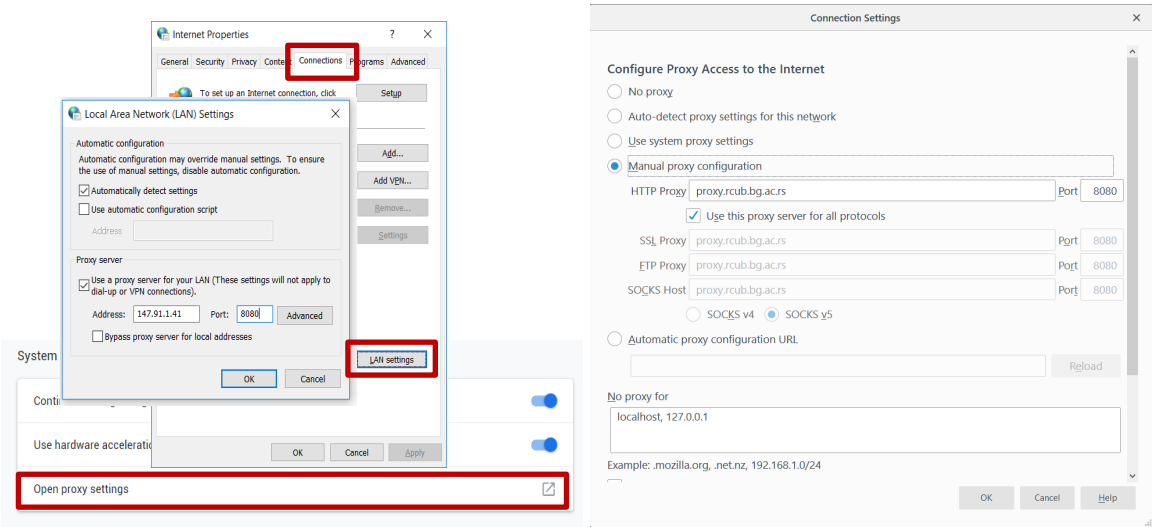
Slika 12.1. Proksi server

Prednost koju ostvaruje proksi server se postiže pri naknadnim zahtevima za isti sadržaj, kada se klijentu vraća keširani sadržaj, bez potrebe da se on ponovo preuzima od udaljenog veb servera. Na ovaj način se racionalno obrađuju zahtevi uz smanjenje opterećenja spoljašnjeg linka i brži odziv prema korisnicima. Keširani sadržaj se čuva određeno vreme i postoji opasnost da nije u potpunosti ažuran u slučaju izmena originalnih stranica.

Dodatne prednosti koje ostvaruje proksi servis se ogledaju u povećanju privatnosti korisnika, koji za spoljašnje veb sajtove ostaju sakriveni iza proksi servera, kao i mogućnost kontrole pristupa određenim sadržajima. Treba istaći pozitivnu stranu ove kontrole, koja se ostvaruje radi zaštite klijenata, a ne cenzure, kao npr. filtriranje neprimerenog sadržaja za decu, što se često sprovodi u školskim mrežama.

Korišćenje proksi servera na korisničkoj strani se podešava kao opcija u veb pretraživaču (Slika 12.2). Osim postavljanja IP adrese ili naziva proksi servera, kao i broja porta, postoji mogućnost zadavanja sajtova za koje se neće koristiti proksi.

Nakon sprovedenog podešavanja, korisnik uobičajeno koristi pretraživač navodeći nazive željenih sajtova, dok pretraživač u pozadini komunicira direktno sa proksi serverom, koji za njega obezbeđuje sadržaj.



Slika 12.2. Primer podašavanja proksi server na pretraživačima Google Chrome i Mozilla Firefox

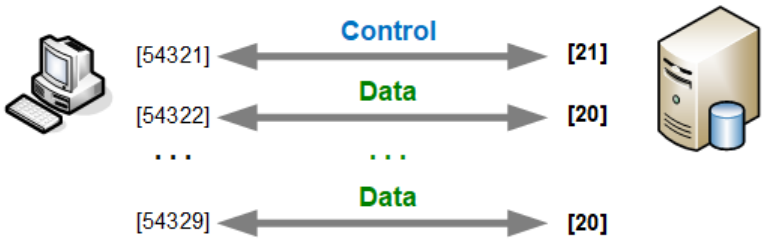
12.3 FTP – Protokol za razmenu datoteka

U ranom periodu korišćenja TCP/IP protokola i Interneta, a pre pojave veb servisa, jedan od osnovnih načina razmene podataka se odnosio na razmenu celih datoteka, što sprovodi protokol za razmenu datoteka, poznatiji kao FTP [36].

Pouzdan prenos je tom prilikom bitan zahtev, posebno za prenos većih datoteka, tako da se koristi TCP protokol, i to preko dve paralelne konekcije:

- Kontrolna konekcija na TCP portu broj 21, koja služi za zadavanje komandi.
- Konekcija za prenos podataka na TCP portu broj 20, koja se otvora po potrebi za prenos pojedinačnih datoteka.

Kao klijentska aplikacija, FTP se pokreće na strani korisnika iz komandne linije i koristi se u tekstualnom modu. Tom prilikom se uspostavlja kontrolna konekcija sa biranim FTP serverom, često uz zadavanje korisničkog imena i lozinke. Korisnik zadaje određene komande, koje se do servera prenose kao 7-btni ASCII tekst. Omogućeno je da korisnik lista sadržaj dostupnih direktorijuma na FTP serveru, tražeći željenu datoteku. Server pamti stanje aktivnosti svakog korisnika posebno, kao što je putanja trenutno aktivnog direktorijuma, što predstavlja tzv. *stateful* režim rada. Korisnik ima mogućnost da preuzme izabranu datoteku komandom GET, kao i da postavi svoju datoteka na server komandom PUT, čime se postiže prenos datoteka u oba smera. Tom prilikom se za prenos svake datoteke otvara nova konekcija preko TCP porta broj 20, koja se zatvara nakon uspešnog prenosa.



Slika 12.3. Preuzimanje i postavljanje datoteka putem FTP protokola

Iako je ranije FTP bio jedan od bazičnih mrežnih servisa, danas se interaktivno preuzimanje i postavljanje datoteka sprovodi putem veb servisa, odnosno preko HTTP i HTTPS protokola.

Istaknimo da postoji i servis prenosa datoteka koji se bazira na UDP protokolu po portu broj 69, a koji se naziva **TFTP** (***Trivial File Transfer Protocol***). Samo ime ukazuje da je njegova realizacija jednostavnija, a korišćenje UDP protokola čak ne garantuje ni pouzdan prenos. Ipak, TFTP protokol i danas nalazi primenu za jednostavnije potrebe, kao što je preuzimanje ili postavljanje konfiguracionih datoteka na rutere ili svičeve.

12.4 Servis elektronske pošte

Servis elektronske pošte, tzv. imejl (*email*) je jedan od prvih korisničkih servisa, koji je vrlo brzo masovno prihvaćen, a gotovo neizmenjen i podjednako popularan zadržao se i do današnjih dana. Elektronska pošta se razmenjuje između korisnika koji koriste klijentske računare, a posredstvom posebnih servera koji komuniciraju u pozadini.

Ne treba posebno objašnjavati da se adrese elektronske pošte sastoje iz korisničkog imena i naziva domena[13] kome pripada adresa. Za ovaj domen je vezan određeni imejl server, gde korisnici poseduju svoje „elektronske sandučiće“.

Servis elektronske pošte ostvaruje se preko TCP protokola i porta broj 25, pod nazivom SMTP (*Simple Mail Transfer Protocol*) [37][38]. Elektronska pošta se sastoji od tekstualnog sadržaja, koristeći 7-bitsko ASCII kodovanje i posebnog tekstualnog formata pod nazivom *Internet Message Format* [39]. Ova tekstualna poruka se sastoji iz zaglavlja poruke, koje navodi IP adrese pošiljaoca i primaoca, IP adrese servera i druge tehničke detalje bitne za komunikaciju, i tela poruke, koja predstavlja sadržaj koji se prenosi. Originalno je bilo predviđeno prenošenje samo tekstualnih sadržaja, a da bi se podržali i bilo koji drugi binarni formati unutar teksta ili kao prilog (*attachment*), uveden je prošireni format, poznat kao MIME (*Multipurpose Internet Mail Extensions*) [40]. Na ovaj način binarni sadržaj se tretira kao niz bita, koji se pretvaraju u 7-bitske ASCII karaktere i prenose sa tekstualnim delom poruke putem originalnog SMTP protokola. Kao i kod HTTP protokola, za ostvarivanje sigurne komunikacije koristi se TLS podsloj, za šta su rezervisani dodatni TCP portovi 465 i 587.

Prilikom slanja elektronske pošte putem SMTP servisa, cilj je da se poruka dostavi u poštansko sanduče primaoca, što se sprovodi u dva koraka (Slika 12.4):

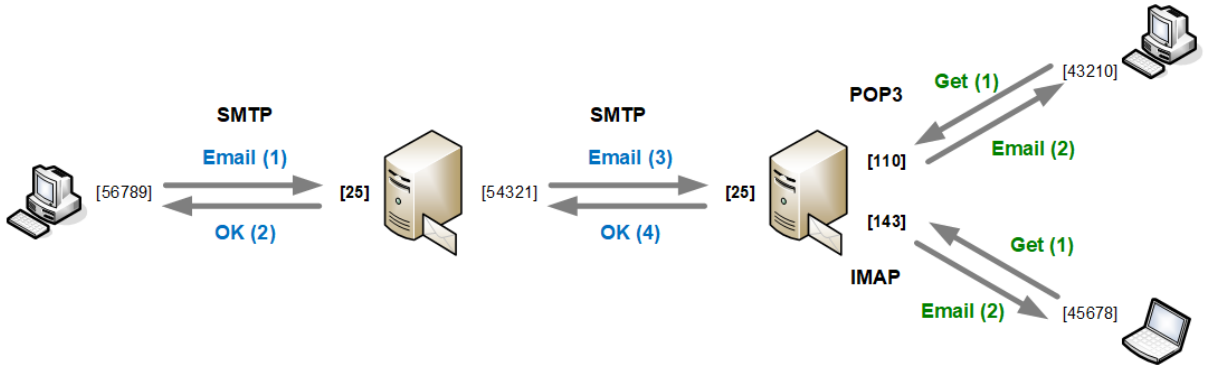
- Klijentska aplikacija pošiljaoca otvara TCP konekciju na port 25 prema matičnom imejl serveru, kojem prosleđuje poštu.

[13] Koncept domena se detaljno objašnjava u potpoglavlju DNS (*Domain Name System*)

- Matični server prihvata poštu, a na osnovu imejl adrese primaoca pronalazi odredišni server, sa kojim uspostavlja novu TCP konekciju na port 25, ali ovog puta u ulozi klijenta, i isporučuje poštu koja se smešta u poštansko sanduče krajnjeg korisnika.

Da bi korisnik pročitao primljenu poštu, on mora da pristupi svom serveru i očita sadržaj svog poštanskog sandučeta. Ovo se sprovodi kroz servis preuzimanja pošte, za šta se mogu koristiti dodatni protokoli:

- POP3 (*Post Office Protocol version 3*) – Prvobitni protokol za preuzimanje elektronske pošte sa servera, za šta se koristi TCP port 110, dok se uz korišćenje TLS podsloja koristi TCP port 995 [41].
- IMAP (*Internet Message Access Protocol*) – Napredniji servis kojim se sprovodi sinhronizacija sadržaja elektronske pošte na serveru i na više uređaja, za šta se koristi TCP port 143, dok se uz korišćenje TLS podsloja koristi TCP port 993 [42].



Slika 12.4. Proces slanja i preuzimanja elektronske pošte

Pristup imejl serveru uz slanje, preuzimanje i upravljanje elektronskom poštom je moguće ostvariti i putem veb servisa, koristeći posebne aplikacije na samom serveru, tzv. veb-mejl (*web mail*). Ovo je uobičajeni način korišćenja koji obezbeđuju provajderi servisa elektronske pošte (*gmail*, *yahoo*, *Microsoft 365* itd.), ali i alternativa za pristup sa bilo kog uređaja, bez potrebe da se instalira i podešava posebna klijentska aplikacija.

12.5 Udaljeni pristup uređajima

Posebna pogodnost koju nude računarske komunikacije je mogućnost udaljenog pristupa uređajima, korišćenjem lokalnog monitora i tastature kao da su fizički povezani na udaljeni uređaj. Ovaj način pristupa je još značajniji pri korišćenju virtuelnih servera i računarstva u oblaku (*cloud computing*, *cloud services*), novom trendu „rada od kuće“ i stvaranju virtuelnih poslovnih infrastruktura. I pored ovih novih slučajeva korišćenja, udaljeni pristup uređajima je jedan od najstarijih korisničkih servisa, inicijalno realizovan kroz servis ***telnet*** [43].

Telnet servis omogućava pristup komandnom tekstualnom modu, tzv. CLI (*Command Line Interface*), korišćenjem TCP prota broj 23. Tradicionalno se primenjuje na mnogim serverima, posebno pod Linux operativnim sistemima, ali i ruterima i svičevima. Tom prilikom je kroz tekstualni dijalog potrebno dostaviti korisničko ime i lozinku za pristup udaljenom sistemu.

Kao jedan od bazičnih servisa, korišćenjem telnet protokola moguće je uspostaviti TCP konekciju na bilo koji port, sprovodeći *tree-way handshake* razmenu poruka. Nakon uspostavljanja komunikacije, moguće je unositi i tekstualne poruke koje će da se prenesu na

drugu stranu, ali je malo verovatno da će one biti od koristi. Ipak, ova osobina je korisna ne da bi se simulirala komunikacija drugih servisa, već da se utvrdi da li postoji mogućnost ostvarivanja komunikacije ili postoji neki problem sa korišćenjem servisa.

Osnovni nedostatak telnet protokola je što se sav sadržaj prenosi u otvorenom tekstu (*plain taxt*), bez korišćenja šifrovanja podataka. Ovo predstavlja posebnu ranjivost, posebno pri prenosu korisničkih naziva i lozinki, jer se pristupom mrežnoj infrastrukturi u određenoj tački podaci mogu kopirati i njihov sadržaj kompromitovati. Ovo je osnovni razlog što se telnet protokol danas retko koristi, a njegovo mesto je zauzeo noviji protokol pod nazivom **SSH** (***secure shell***) [44].

Već i ime ukazuje da je SSH siguran protokol za pristup udaljenim sistemima, koristeći TCP port broj 22. Celokupni saobraćaj koji se odvija između obe strane je šifrovan, što se postiže generisanjem privremenog zajedničkog ključa za šifrovanje koji se koristi tokom trajanja cele TCP konekcije. Posebno je bitno što se i korisničko ime i lozinka prenose šifrovano.

I telnet i SSH servisi omogućavaju pristup uređajima samo u tekstualnom modu, dok je danas uobičajen način korišćenja kroz grafički mod, posebno kod *Windows* operativnih sistema. Udaljeni pristup grafički orjentisanim sistemima se sprovodi putem **RDP** protokola (***Remote Desktop Protocol***), koji originalno predstavlja protokol za *Windows* operativne sisteme, koristeći TCP port 3389, ako i UDP port 3389 za pojedine namene.

RDP klijent predstavlja sastavnu komponentu savremenih *Window* operativnih sistema, a postoje i klijentske implementacije za ostale operativne sisteme, kao što su *Linux, macOS, iOS, Android* i sl. Ostali operativni sistemi podržavaju varijantu protokola pod nazivom **freeRDP**.

13. DNS – sistem imenovanja

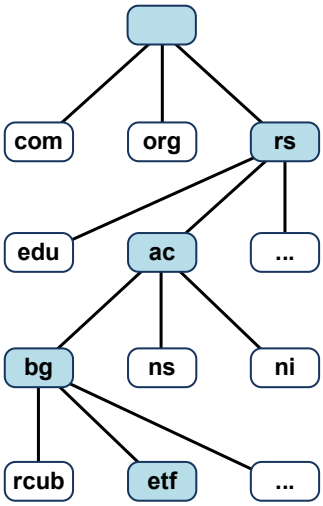
U prethodnim poglavljima detaljno je opisan koncept komunikacije na internetu korišćenjem IP adresa, koje su jednostavne za mašinsku obradu, ali veoma nepraktične za upotrebu od strane korisnika. Umesto IP adresa, za korisnike je jednostavniji oblik identifikacije uređaja putem simboličkih naziva koji se navode u korisničkim aplikacijama, kao što su nazivi veb sajtova ili imejl adrese. Stoga je neophodan mehanizam koji će za zadate nazive da vraća njima uparene IP adrese.

Potreba za imenovanjem računara i mapiranjem naziva u IP adrese se javila u ranim danima povezivanja putem TCP/IP protokola, u tadašnjoj ARPANET mreži iz koje je nastao Internet. Nazivi i IP adrese su se inicijalno definisale na jednom mestu, na računaru u Stenford institutu (*Stanford Research Institute*), i to u tekstualnoj datoteci pod nazivom „HOSTS.TXT“, koji su svi ostali računari u mreži preuzimali na dnevnom nivou [45]. Ovaj jednostavan centralizovani pristup nije bio skalabilan, pa nije moglo da isprati nagli razvoj mreže i veliki porast broja računara. Bilo je neophodno razviti skalabilni sistem i distribuiranu infrastrukturu koja funkcioniše u pozadini mreže i aplikacija, a obezbeđuje uparivanje naziva i IP adresa, sistem koji se naziva **DNS** (***Domain Name System***) [46][47].

U tehnološkom smislu, DNS predstavlja protokol aplikativnog sloja, koji primarno radi preko UDP protokola po portu 53, a u posebnim slučajevima se koristi i TCP, takođe po portu 53. DNS sistem čini veliki broj DNS servera na kojima su definisani nazivi uređaja i njima pridružene IP adrese, a koji su raspoređeni po celom Internetu. Osnovni zadatak DNS servera je da za upite o nazivima uređaja vraća njima uparene IP adrese. Upite generiše sistemska klijentska DNS aplikacija na uređajima, tzv. DNS *resolver*, i to na zahtev i za potrebe ostalih korisničkih aplikacija na istom uređaju. Osnovni problem koji se tom prilikom javlja je da se za određeni naziv na celom Internetu pronađe upravo onaj DNS server gde je taj naziv definisan, kako bi se preuzela njemu pridružena IP adresa.

13.1 Organizacija i definicija domena

Simbolički nazivi se definišu hijerarhijski i interpretiraju se u strukturi stabla. Čvorovi stabla su simbolička imena koja se nazivaju domeni. Za određeni domen, čvorovi izvedeni na nižem nivou („deca“) se nazivaju poddomeni. Krajnji čvorovi u stablu (tzv. „listovi“) predstavljaju nazive uređaja unutar pripadajućeg domena. Puni naziv bilo kog čvora, odnosno domena ili krajnjeg uređaja, predstavlja putanja do korena stabla, koja redom sadrži sve nazive čvorova razdvojene znakom tačke (***Fully Qualified Domain Name*** – **FQDN**). Koren hijerarhije je neimenovan, odnosno predstavljen je praznim tekstom, tako se puni naziv završava znakom tačke. U uobičajenoj upotrebi, poslednji znak tačke se implicitno podrazumeva i može da se izostavi (Slika 13.1).



Slika 13.1. Hijerarhija domena na primeru punog naziva „etf.bg.ac.rs.“

Simbolički naziv jednog čvora u hijerarhiji (domena ili krajnjeg uređaja) sastoji se od maksimalno 63 ASCII karaktera, koje čine slova, brojevi i znakovi „-“ (minus) i „_“ (donja crta). Pri tome se ne pravi razlika između velikih i malih slova. Ukupna dužina punog naziva je ograničena na 255. Dosta kasnije, omogućena je i upotreba unikod karaktera (*unicode*), čime je moguće koristiti različita pisma, uključujući i ćirilicu [48]. Ova praksa ipak nije zaživela u većoj meri.

Domeni najvišeg nivoa nazivaju se „top-level domeni“, skraćeno TLD (*Top-Level Domain Names*), i dele se u dve kategorije [49]:

- *Country-code TLD* (ccTLD) – domeni dodeljeni međunarodno priznatim državama, čiji nazivi čine dva slova oznake države prema standardu ISO 3166 [50].
- *Generic TLD* – generički domeni, koji su inicijalno bili namenjeni za Sjedinjene Američke Države, dodatno u odnosu na postojeći ccTLD pod nazivom „US“ i to sledeći domeni:
  - EDU – za obrazovne organizacije
  - COM – za komercijalne organizacije
  - NET – za provajdere
  - GOV – za vladine organizacije
  - MIL – za vojne organizacije
  - ORG – za ostale organizacije
  - INT – za međunarodne organizacije.

Vremenom se lista generičkih TLD proširila sa novim nazivima za dodatne namene (TRAVEL, JOBS, MUSEUM, AERO itd.), ali su se i liberalizovala pravila korišćenja, čime su se domeni COM, NET, ORG i INT otvorili i za sve organizacije i pojedince, nevezano od teritorije registracije ili boravka. Ovi top-level domeni su ujedno i najpopularniji.

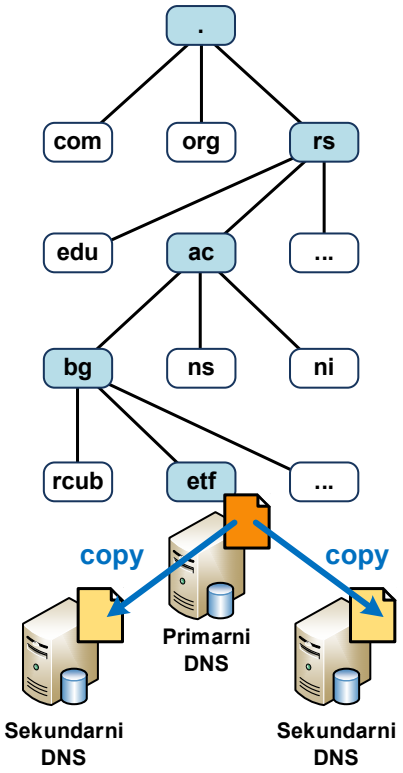
Svaki domen, sa pripadajućim nazivima uređaja ili nazivima drugih poddomena, mora biti negde fizički definisan, a logički vezan za roditeljski domen. To se postiže konceptom podele na tzv. **zone**, koje sadrže odgovarajuće definicije i koje se parcijalno odnose na mali deo celokupne DNS hijerarhije, obično samo jedan čvor, eventualno i sa svojim poddomenima. U

organizacionom smislu, zona obično pripada jednom entitetu koji je vlasnik domena (npr. univerzitet, kompanija, organizacija, projekat ili pojedinac). U tehničkom smislu zona predstavlja kolekciju informacija, tzv. zapisa resursa (***Resource Record*** – **RR**), a koje se nalaze na serverima koji realizuju DNS servis, tzv. **DNS serveri**, često označavani kao NS (***Name Server***).

Tradicionalan način predstavljanja zapisa u zoni je u vidu tekstualne datoteke (***zone file***), koja se održava u posebnoj sintaksi, ali postoje i aplikacije koje omogućavaju jednostavnije definisanje zapisa i upravljanje domenima. Zone su definisane samo na jednom DNS serveru, koji se naziva **primarni DNS server** za tu zonu (*primary DNS server*). Dodatno se uvode i tzv. **sekundarni DNS serveri** (*secondary DNS server*), na koje se kopiraju zone kroz proces koji se naziva **transfer zone**. Kada se jednom konfiguriše na primarnom i sekundarnom DNS serveru, transfer zona se sprovodi automatski. Na taj način i primarni i sekundarni serveri sadrže iste podatke i podjednako su nadležni i ravnopravni za razrešavanje imena za domene čije zone sadrže. U DNS terminologiji kaže se da su primarni i sekundarni DNS server **autoritativni serveri** za određeni domen.

Snažna preporuka je da postoji najmanje jedan sekundarni DNS server, i to na mreži koja je što više razdvojena od mreže gde se nalazi primarni DNS server. Na taj način se povećava pouzdanost rada DNS sistema, ali i rasterećuje primarni DNS server.

Uloga primarnog i sekundarnog DNS servera se uvek posmatra u kontekstu određenog domena. Jedan DNS server može da bude primaran za određene domene, a za druge domene da bude sekundaran.



Slika 13.2. Transfer zona sa primarnog na sekundarne DNS servere

Novi domen se može u potpunosti definisati unutar zone roditeljskog domena, što se obično radi za poddomene iste organizacije (npr. domen za katedru na fakultetu ili sektor u kompaniji).

U slučaju otvaranja novog domena unutar samostalne zone, proces definisanja obuhvate sledeće neophodne korake:

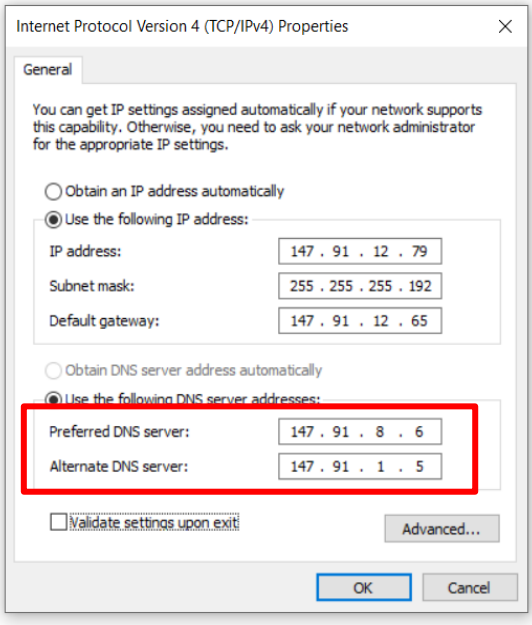
- Konfigurisanje primarnog DNS servera za domen.
- Definisanje zapisa novog domena na primarnom DNS serveru.
- Konfigurisanje i uparivanje sa jednim ili više sekundarnih DNS servera.
- Prijavljivanje kod roditeljskog domena radi integracije u hijerarhiju domena, što se postiže izmenom zapisa na primarnom DNS serveru roditeljske zone, kako bi se definisao naziv domena i postavila delegacije zone na primarni DNS server za domen.

Hijerarhija domena i organizacija DNS servera potpuno su nezavisne od fizičke strukture mreže i primenjenih IP adresa, što za posledicu ima sledeće.

- Primarni i sekundarni DNS server za određeni domen ne moraju da pripadaju mreži na kojoj su uređaji iz tog domena, već mogu da se nalazi bilo gde na Internetu.
- Uređaji koji imaju nazive iz jednog domena mogu da se nalaze na različitim fizičkim i IP mrežama, budući da se nazivu uređaja može pridružiti bilo koja IP adresa.
- Jedna fizička ili IP mreža može sadržati uređaje čiji nazivi pripadaju različitim domenima.
- Jedan uređaj, tj. njegova IP adresa, može da ima više naziva iz istog ili različitog domena.

13.2 Razrešavanje naziva

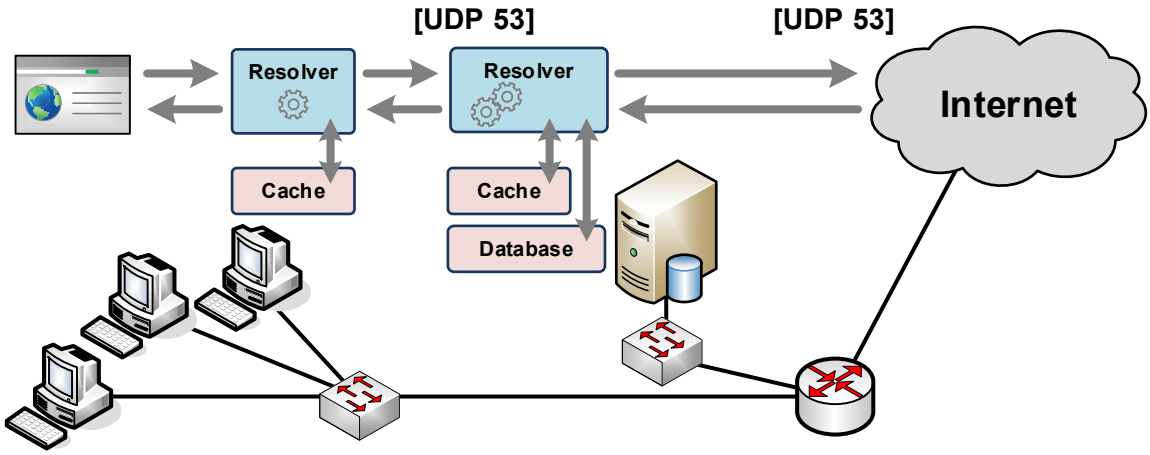
Osnovni zadatak DNS servisa je razrešavanje naziva krajnjih uređaja, odnosno nalaženje njima pridruženih IP adresa. Potrebu za ovom uslugom imaju praktično svi učesnici na Internetu, tako da sastavni deo konfiguracije mrežnih parametara na svim operativnim sistemima podrazumeva i „povezivanje“ na DNS sistem, navođenjem IP adrese podrazumevanog DNS servera koji će da razrešava nazive u IP adrese, tzv. **preferirani DNS server** (*preferred DNS server*). Uobičajeno je da se navede i **alternativni DNS server** (*alternate DNS server*) kao rezerva u slučaju otkaza preferiranog (Slika 13.3).



Slika 13.3. Podešavanje preferiranog i alternativnog DNS servere

Česta je zabluda poistovećivanje preferiranih i alternativnih DNS servera, sa pojmovima primarni i sekundarni DNS serveri. Razlika je očigledna kada se sagleda uloga i rad preferiranih i alternativnih DNS servera pri procesu razrešavanja naziva, što se sprovodi na sledeći način (Slika 13.4):

- Korisnička aplikacija za zadati naziv upit za nalaženje IP adrese postavlja DNS klijentu, koji je deo sistemskog softvera pri operativnom sistemu, tzv. *resolver*.
- DNS klijent najpre traži odgovor na upit u lokalnom kešu, a ako ga ne nađe, onda šalje DNS upit preferiranom DNS serveru koji je definisan za taj računar. Ovaj upit se šalje na UDP port 53.
- Preferirani DNS server za primljeni upit najpre traži odgovor u svojim zonama ako je autoritativan za naziv, zatim u lokalnom kešu, a ako ga ne nađe, onda pronalazi autoritativni server za zadati domen, kome šalje DNS upit takođe putem UDP protokola na port 53 (detaljan opis ovog postupka sledi u nastavku).
- Po dobijanju konačnog odgovora, preferirani DNS server ga prosleđuje DNS klijentu koristeći UDP komunikaciju u suprotnom smeru, uz keširanje rezultata u lokalnom kešu, kao bi se ubrzali odgovori za naredne upite.
- DNS klijent na lokalnom računaru dobijenu IP adresu prosleđuje aplikaciji, uz keširanje rezultata u svom lokalnom kešu.
- Aplikacija nastavlja komunikaciju sa udaljenim uređajem korišćenjem njegove IP adrese.



Slika 13.4. Razrešavanje imena od strane preferiranog DNS servera za potrebe ostalih računara

Za proces razrešavanja naziva na Internetu ključno je nalaženje autoritativnog (primarnog ili sekundarnog) DNS servera za traženi naziv, što se odvija u više iteracija, koje demonstrira Slika 13.5 na primeru upita za naziv www.etf.bg.ac.rs.

Najpre istaknimo da postoji ukupno 13 fiksnih i dobro poznatih globalnih autoritativnih DNS servera za sve top-level domene. IP adrese ovih servera su predefinisane za sve DNS servere, uključujući i posmatrani preferirani DNS server koji treba da razreši upit za posmatrani naziv, što se sprovodi kroz sledeće korake:

- Ako u lokalnom kešu preferiranog DNS servera ne postoje prethodno zapamćeni autoritativni DNS serveri ni za jedan od poddomena unutar naziva “www.etf.bg.ac.rs”, šalje se DNS upit na bilo koji od 13 *root* DNS servera za razrešavanje naziva.
- *Root* DNS server ne može da razreši ceo naziv “www.etf.bg.ac.rs”, ali prepoznaje da je u njegovoj top-level zoni definisano domen „rs“ i kao parcijalni odgovor vraća IP adrese autoritativnih DNS servera kojima je konfigurisana zona za domen „rs“.
- Preferirani DNS server bira jedan od dobijenih autoritativnih DNS servera za „rs“ domen i ponavlja upit, za koji na isti način kao parcijalni odgovor dobija listu IP adresa autoritativnih severa za domen „ac.rs“.
- Postupak se iterativno ponavlja na niže, sve dok se ne dođe do poslednjeg poddomena, u ovom slučaju „etf.bg.ac.rs“, čiji autoritativni serveri imaju definiciju naziva „www“ sa odgovarajućom IP adresom, koja se vraća kao konačan odgovor.

Navedeni DNS upiti za koje se u više iteracija vraćaju parcijalni odgovori nazivaju se **iterativni upiti**. DNS upiti koje klijent postavlja preferiranom DNS serveru se u potpunosti razrešavaju, koristeći u pozadini sekvencu iterativnih upita. Ovi upiti se nazivaju **rekurzivni upiti**.

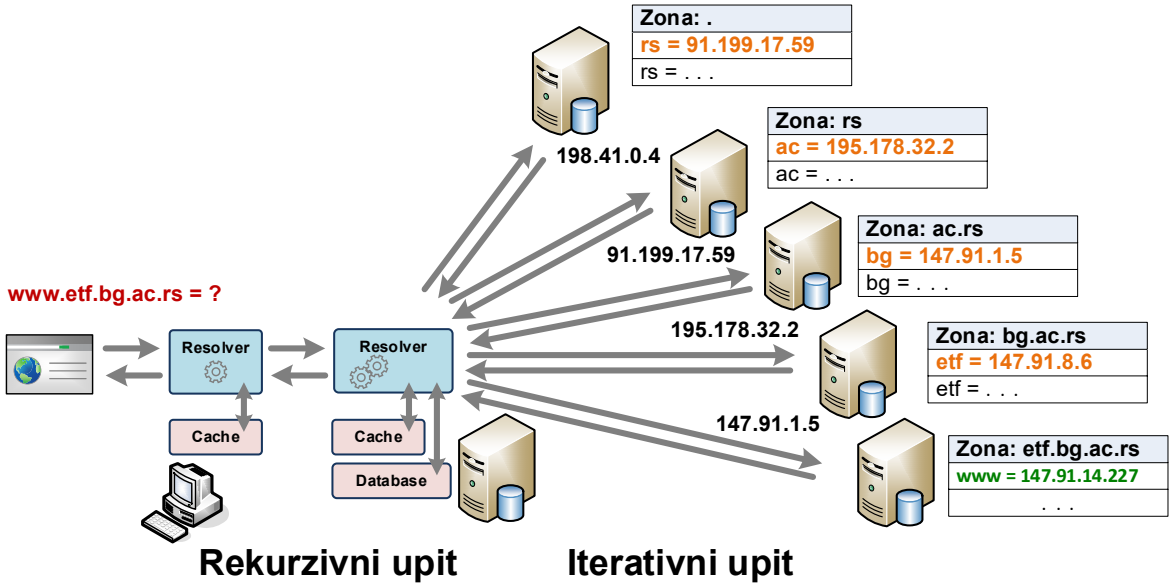
Čitaoci su kroz lično iskustvo sigurno primetili da prvobitni odziv pojedinih veb sajtova može da bude nešto duži, čak nekoliko sekundi, dok su naredni pristupi drugim stranicama istog sajta znatno brži. Tipičan razlog je iterativni postupak razrešavanja imena koji se sprovodi u pozadini. Spor odziv je posebno izražen u slučaju da je neki autoritativni server trenutno nedostupan, obično zbog greške u konfiguraciji, pa se dodatno čeka *timeout* period, dok se na pređe na naredni autoritativni server iz dobijenog odgovora.

Imajući u vidu navedeno, jasno je da keširanje prethodno dobijenih odgovora znatno ubrzava ceo proces. Tom prilikom je bitno vreme „životnog veka“ zapisa u kešu, što se postavlja u definiciji samog zapisa.

Istaknimo i to da preferirani DNS serveri obavljaju zahtevan posao za korisničke računare, često opslužujući veliki broj zahteva klijenata, a istovremeno postavljajući upite drugim DNS serverima. Dobra praksa je da se uloga preferiranog servera kroz dozvoljavanje rekurzivnih upita omogućava samo za ograničeni i kontrolisani skup računara, npr. unutar iste organizacije. Ovo nije obavezno, pa se usled nepažnje greškom mnogi DNS serveri ostavljaju „otvoreni“ za rekurzivne upite bilo kog klijenta na Internetu, što predstavlja dvojaku pretnju:

- DNS server je podložan prevelikom broju zahteva, što može da ga preoptereti i naruši njegovu operativnost, tzv. napad na raspoloživost servisa (*Denial-of-Service – DoS*).
- DNS server može da bude izložen prevelikom broju zahteva koji su „lažno“ poslati koristeći tuđu IP adresu kao izvorišnu (*spoofed*), na koju se šalje veliki broj neželjenih odgovora. Ova vrsta napada se zove pajačavajući DNS napad (*DNS amplification attack*), u kom slučaju je DNS server zloupotrebljen da učestvuje u distribuiranom napadu na raspoloživost servisa (*Distributed Denial-of-Service – DDoS*).

Postoje i izuzeci kada se dozvoljavaju rekurzivni upiti za sve korisnike (*Global Public Recursive DNS Resolver Service*), što se često koristi od strane organizacija koje nemaju svoje lokalne DNS servere. Popularni serveri ove vrste su Google DNS server na adresi 8.8.8.8 i DNS server fondacije *Quad9* na adresi 9.9.9.9, a koji su posebno zaštićeni od prethodno opisanih malicioznih napada.



Slika 13.5. Iterativno razrešavanje imena od strane u hijerarhiji domena za potrebe preferiranog DNS servera

13.3 Zapisi u DNS zonama

Relevantni DNS podaci za domen se definišu u zoni na primarnom DNS serveru, koja je predstavljena u obliku tekstualne datoteke u ASCII formatu. Sadržaj zone čine tzv. zapisi o resursu (*Resource Record* – RR), koji definišu pojedinačne DNS podatke, kao što je definicija naziva uređaja sa pridruženom IP adresom ili definicija poddomena. Odgovori na DNS upite koje vraćaju DNS serveri čine jedan ili više ovih zapisa.

U opštem slučaju, sintaksa zapisa je sledeća:

Name TimeToLive Class Type Value

gde je:

- **Name** – naziv resursa (npr. naziv domena ili uređaja). Ako se izostavi naziv, podrazumeva se naziv iz prethodnog zapisa. Takođe, ako naziv ima tačku na kraju, podrazumeva se njegovo puno DNS ime (FQDN). Bez tačke, naziv se tretira kao prefiks uz tekući naziv domena (*ORIGIN*).
- **TimeToLive** – opcioni parametar koji predstavlja vreme validnosti podatka prilikom keširanja odgovora u drugim DNS serverima ili klijentima, izražen u sekundama. Ako se izostavi ova vrednost, podrazumeva se vrednost koja važi na nivou cele zone (polje MINIMUM iz SOA zapisa, kako sledi).
- **Class** – „klasa“ resursa, koja u ovom slučaju nosi fiksnu oznaku „IN“, ukazujući na „Internet resurse“.
- **Type** – oznaka tipa resursa, koja ukazuje na šta se odnosi definicija zapisa.
- **Value** – vrednost koja se pridružuje resursu (npr. konkretna IP adresa koja se pridružuje nazivu).

Inicijalna DNS specifikacija definiše 16 tipova zapisa [50], što je kasnije proširivano za nove potrebe. U nastavku se opisuju najvažniji tipovi zapisa, čija je upotreba demonstrirana na

primeru definicije zone za domen „bg.ac.rs“, čije je početak naveden u listingu koji prikazuje Slika 13.6.

```
bg.ac.rs. IN     SOA  ns.rcub.bg.ac.rs. helpdesk.rcub.bg.ac.rs. (
                        2022032502 ; serial
                        10800      ; refresh (3 hours)
                        3600       ; retry (1 hour)
                        604800     ; expire (1 week)
                        86400      ; minimum (1 day)
                  )
                  NS   147.91.1.5           ; autoritativni server za zonu
                  NS   147.91.1.7           ; autoritativni server za zonu
                  NS   ns1.uns.ac.rs.       ; autoritativni server za zonu
                  NS   ban.junis.ni.ac.rs.  ; autoritativni server za zonu
                  NS   ns.unic.kg.ac.rs.    ; autoritativni server za zonu
                  NS   ns.etf.bg.ac.rs.     ; autoritativni server za zonu
                  MX      147.91.79.3       ; predefinisani email server za zonu
                  A       147.91.79.3     ; predefinisani veb server za zonu
etf               IN NS   ns.etf.bg.ac.rs. ; uvodi se novi poddomen
                  IN NS   ns.rcub.bg.ac.rs. ;   sa autoritativnim serveri
ns.etf            IN A    147.91.8.6 ; glue record
rcub.bg.ac.rs.    IN NS   ns.ni.ac.rs.    ; uvodi se novi poddomen
                  IN NS   ns.rcub.bg.ac.rs. ;   sa autoritativnim serveri
ns.rcub           IN A    147.91.1.5      ; glue record - adresa servera
www               IN A    147.91.79.3            ; adresa veb servera
proxy-web         CNAME   www            ; alijas na web server
...
```

Slika 13.6. Primer definicije zone za domen „bg.ac.rs“

SOA zapis (Start of Authority)

Ovaj zapis mora da stoji na početku datoteke koja definiše zonu. Naziv resursa se odnosi na naziv domena za koji se definišu podaci u zoni, dok vrednost sadrži više parametara u sledećem redosledu:

- **MNAME** (*Master Name*) – Naziv primarnog DNS servera za tekući domen, koji se koristi samo kao informativni podatak, bez stvarnog uticaja na konfiguraciju i funkcionisanje DNS sistema.
- **RNAME** (*Responsible Name*) – Imejl adresa administratora domena, takođe kao informativni podatak, uz specifičnost da se umesto uobičajenog znaka “@” koristi tačka (“.”). Razlog tome je što znak “@” ima posebnu namenu u sintaksi zona datoteke, koja ukazuje na naziv domena koji se može staviti kao opšti parametar u zona datoteku („$ORIGIN”).
- **SERIAL** – Serijski broj verzije zone, koji može da bude bilo koji pozitivan broj, gde nova verzija može da ima proizvoljnu vrednost, ali obavezno veću od prethodne. Verzija je jako bitan podatak, budući da sekundarni DNS serveri sa primarnog servera prvo čitaju verziju, pa jedino ako je ona veća (novija) od tekuće koju poseduju, tada sekundarni inicira transfer zone sa primarnog. Česta je greška da se pri izmenama zone ne ažurira serijski broj, pa se novi ili izmenjeni zapisi u zoni ne propagiraju na sekundarni DNS server, dovodeći podatke u nekonzistentno stanje.

  Dobra je praksa da se serijski broj bude u formatu: „yyyymmddnn“, ukazujući na trenutni datum, sa poslednje dve cifre koje se odnose na redni broj promene u tekućem danu.
- **REFRESH** – Period vremena, izražen u sekundama, koji ukazuje posle kog vremena će sekundarni DNS server proveravati da li ima promena u zoni na primarnom DNS serveru, odnosno da li je SERIAL vrednost povećana. U zavisnosti od frekventnosti promena, preporuka je da ova vrednost bude u intervalu od 1.200 sekundi (20 minuita) do 86.400 sekundi (24 časa).

- **RETRY** – Period vremena, izražen u sekundama, koji ukazuje posle kog vremena će sekundarni DNS server ponovo proveravati da li ima promena u zoni na primarnom DNS serveru, ukoliko prethodna provera nije uspela, odnosno ako se primarni DNS server nije odazivao. Preporuka je da ova vrednost ne bude veća do 7.200 sekundi (2h).
- **EXPIRE** – Period vremena, izražen u sekundama, koji ukazuje koliko dugo će sekundarni DNS server čuvati podatke iz zone u slučaju da je komunikacija sa primarnim DNS serverom konstantno u prekidu. Preporuka je da ova vrednost bude u intervalu od 1.209.600 sekundi (14 dana) do 2.678.400 sekundi (31 dan).
- **MINIMUM** – Predefinisani period vremena (tajmer) za sve zapise iz zone, izražen u sekundama, koji ukazuje koliko dugo se zapisi iz zone čuvaju prilikom keširanja odgovora u drugim DNS serverima ili klijentima. Preporuka je da ova vrednost bude u intervalu od 86.400 sekundi (1 dan) do 432.000 sekundi (31 dan).

Alternativno, navedeni periodi se mogu zadati i u formatu koji ukazuje na ostale vremenske jedinice, veće od sekunde, što ilustruju sledeći primeri:

- 30M – 30 minuta
- 3h – 3 sata
- 1d – 1 dan
- 3W2d – 3 nedelje i 2 dana

U listingu koji prikazuje Slika 13.6, karakter „;“ se odnosni na početak teksta komentara, što se ignoriše.

NS zapis (Name Server)

Zapisima ovog tipa se definišu autoritativni DNS serveri za tekući domen i pripadajuće poddomene.

Zona se odnosi na određeni domen i počinje njegovim imenom koje se navodi u SOA zapisu. Kod narednih zapisa koji se odnose na taj isti domen, ime domena može da se izostavi. Uobičajeno je da se nakon SOA zapisa navedu NS zapisi koji definišu adrese primarnog i svakog sekundarnog DNS servera tekućeg domena. Tom prilikom se ne navodi koji je primarni, a koji je sekundarni DNS server, budući da oni ravnopravno razrešavaju imena, pa ni njihov redosled nije bitan.

Poddomeni se unutar tekućeg domena definišu navođenjem naziva poddomena na početku NS zapisa, a zatim i adrese primarnog i svih sekundarnih servera. Naziv poddomena se može navesti na dva načina:

- Naziv samo labele poddomena, bez znaka tačke na kraju i bez roditeljskog domena (Slika 13.6, primer poddomena „etf“).
- Pun naziv poddomena (FQDN), koji mora da se završava za znakom tačke (Slika 13.6, primer poddomena „rcub.bg.ac.rs.“).

Definicija poddomena preko NS zapisa u zoni roditeljskog domena je od ključnog značaja, imajući u vidi iterativni postupak razrešavanja imena, koji se sprovodi od vrha hijerarhije domena naniže do pripadajućih poddomena, za koje je neophodno znati IP adrese autoritativnih DNS servera. Šta više, za tekući domen koji se definiše u posmatranoj zoni, osim

navođenja NS zapisa neposredno ispod SOA zapisa još važnije je da se on definiše kao poddomen u zoni njegovog roditeljskog domena, da bi se oni vratili kao parcijalni odgovor u iterativnom postupku razrešavanja imena. Bez definicije na roditeljskom nivou preko NS zapisa, domen bi ostao odsečen od celokupne DNS hijerarhije.

Autoritativni DNS server za domen ili poddomen se u NS zapisu osim u obliku IP adrese može navesti i kao naziv u FQDN formi. Ovaj pristup je zgodan za administratore domena, zato što istovremeno ukazuje u čijoj je nadležnosti DNS server. Ipak, to je manje efikasno u procesu razrešavanja imena, budući da se u iterativnom odgovoru vraća naziv servera, koji takođe treba razrešiti. Ako on nije u lokalnom kešu DNS servera koji sprovodi upit, tada je potrebno pokrenutu novi „ugnježđeni“ niz iterativnih upita za njegovo razrešenje. Ovo dodatno čekanje će da se odrazi na odziv aplikacije koja je pokrenula inicijalni rekurzivni DNS upit od svog preferiranog DNS servera.

Drugi mogući problem pri korišćenju naziva DNS servera u NS zapisima se odnosi na slučaj kada se za definiciju poddomena koristi naziv DNS servera koji pripada tom istom poddomenu. Proces razrešavanja naziva iz poddomena tada uzlazi u petlju – za dalje razrešavanje naziva iz tog poddomena potrebno je pristupiti autoritativnom DNS serveru, a ako je on dat preko naziva koji pripada tom domenu, onda se ni njegov naziv ne može razrešiti u IP adresu. Ako se u iterativnom upitu koristi baš ovako definisan server, tada će se u razrešavanju čekati određeni period (više sekundi), nakon čega će se pokušati sa drugim DNS serverom. Krajnji rezultat je da aplikativni servis koji je pokrenuo korisnik obično radi ispravno, osim u pojedinim slučajevima kada se pokuša da se koristi navedeni DNS server, pa je proces inicijalnog razrešavanja primetno usporen. Zbog navedene manifestacije, ova neregularnost često ostaje neprimetna za administratore domena.

Drastičan problem nastaje ako su svi DNS serveri iz posmatranog domena navedeni preko naziva koji pripada tom domenu. Tada su svi oni nedostupni, jer se njihove IP adrese ne mogu razrešiti, a time ni druge IP adrese iz tog domena.

Kako bi se izbegli prethodno opisani problemi i ubrzalo razrešavanje imena ukoliko se u NS zapisu koristi puni naziv DNS servera, za ove servere je potrebno eksplicitno dodeliti i IP adrese u posmatranoj zoni, iako je originalno mesto za njenu dodelu zona drugog domena. U konfiguraciji zone koju prikazuje Slika 13.6, to je slučaj sa serverima „ns.etf.bg.ac.rs.“ i „ns.rcub.bg.ac.rs“, kojima su u roditeljskom domenu „bg.ac.rs“ pridružene i IP adrese 147.91.1.5 i 147.91.8.6. Ovakva vrsta dodatnog zapisa se žargonski zove ***glue record***.

MX zapis (Mail Exchange)

Domeni predstavljaju i sastavni deo imejl adresa, pa ih je potrebno povezati sa IP adresom odgovarajućeg imejl servera, što se postiže korišćenjem MX zapisa.

Slično kao i NS zapisi, MX zapis se pridružuje tekućem domenu na koji se odnosi zona, navođenjem ispod SAO zapisa. U konfiguraciji zone koju prikazuje Slika 13.6. za „bg.ac.rs“ domen je preko MX zapisa pridružena IP adresa 147.91.79.3.

A zapis (Host Address)

Prethodni zapisi su se odnosili na tekući domen ili poddomene radi njihovog ispravnog funkcionisanja, dok je primarna uloga DNS sistema razrešavanje naziva krajnjih uređaja u IP adrese. Za ovu namenu se koristi zapis čiji je tip označen sa „A“ (*Host Address*). U konfiguraciji zone koju prikazuje Slika 13.6. za domen „bg.ac.rs“ uveden je naziv „www“ za veb sajt, kome

je dodeljena IP adresa 147.91.79.3. Dodatno je naziv za ceo domen „bg.ac.rs“ uparen sa istom ovom IP adresom navođenjem *A* zapisa nakon SOA zapisa, čime je omogućeno da se veb sajt može pozivati i sa „bg.ac.rs“, a ne samo preko naziva „www.bg.ac.rs“.

CNAME zapis (Canonical Name)

Pojedini uređaji imaju višestruku namenu, za koje je zgodno da imaju različite nazive. Iako se to može postići dodavanjem novih „A“ zapisa, omogućeno je i definisanje novih naziva kao alijasa u odnosu na bazični naziv, za šta se koristi CNAME zapis (*Canonical Name*).

U prethodnom primeru na ovaj način je uvedeno nov naziv „proxy-web“, kako alijas za naziv „www“, a oba naziva će da upućuju na istu IP adresu 147.91.79.3.

Korišćenje alijasa omogućava da se IP adresa definiše samo na jednom mestu, što otklanja potencijalnu nekonzistentnost u slučaju njene promene.

PTR zapis (Pointer)

DNS sistem omogućava da se nazivi razrešavaju u IP adrese, ali je ponekad koristan i obrnuti proces pomoću kojeg se od IP adrese dobija njoj pridruženo ime. Ovo je posebno korisno pri analizi i dijagnostici mrežnih servisa, logova i događaja, jer se na osnovu naziva može prepoznati namena i vlasnik određenog uređaja. Za ove potrebe naravno da nije moguće pretraživati celokupnu DNS hijerarhiju, ali je moguće iskoristiti princip rada DNS servisa i uvesti novu hijerarhiju koja odgovara simboličkoj interpretaciji IP adresa, tzv. **inverzni DNS sistem**.

Koren inverznog DNS sistema je domen pod nazivom „in-addr.arpa“, a IP adresa se u *dotted decimal* formatu tretira kao naziv poddomen, gde su bajtovi IP adrese navedeni kao dekadni brojevi u obrnutom poretku – bajtovi najmanje težine se navode prvi, što odgovara redosledu navođenja poddomena. Na primer, adresa 147.91.1.5 se interpretira kao FQDN naziv „5.1.91.147. in-addr.arpa“. U DNS interpretaciji ovaj naziv predstavlja oznaku resursa „5“ u domenu „1.91.147. in-addr.arpa“.

Stoga se inverzni DNS definiše u odnosu na raspoloživi IP adresni prostor. Za mrežu 147.91.0.0/16 najpre je potrebno definisati inverzni domen „91.147.in-addr.arpa“, a u okviru njega ukupno 256 poddomena koji odgovaraju svim vrednostima za treći bajt, od 0 do 255. Za svaki od ovih poddomena se definiše zona u okvir koje se uvode nazivi koji predstavlja četvrti bajt (Slika 13.7).

Za ovako definisanu IP adresu naziv se uvodi preko zapisa tip PTR (*Pointer*).

```
$ORIGIN 147.in-addr.arpa.
1.91     IN      SOA  ns.rcub.bg.ac.rs. helpdesk.rcub.bg.ac.rs. (
                                2022032502 ; serial
                                10800      ; refresh (3 hours)
                                3600       ; retry (1 hour)
                                604800     ; expire (1 week)
                                86400      ; minimum (1 day)
                           )
                 NS   147.91.1.5           ; autoritativni server za zonu
                 NS   147.91.1.7           ; autoritativni server za zonu
$ORIGIN 1.91.147.in-addr.arpa.
5        IN     PTR   ns.rcub.bg.ac.rs.
7        IN     PTR   gaea.rcub.bg.ac.rs.
18       IN     PTR   webhost.rcub.bg.ac.rs.
...
```

Slika 13.7. Primer definicije zone za inverzni domen „1.91.147. in-addr.arpa.“

Na navedenom primeru istaknimo upotrebu posebne ključne reči „$ORIGIN“ koja uvodi naziv podrazumevanog domena na koga se odnose svi dalji nazivi, ako se ne završavaju sa znakom tačke. U ovom slučaju podrazumevani domen za dalje definicije je „147.in-addr.arpa.“, pa je naziv tekućeg domena definisan sa „1.91“ (bez znaka tačke na kraju). Pre definisanja PTR zapisa bilo je potrebno promeniti predefinisani domen kako bi se odnosio na inverzni domen „1.147.in-addr.arpa.“.

Kako ovaj predefinisani domen na bi uticao na nazive koji se navode uz PTR zapis, oni moraju da se navode u FQDN notaciji, odnosno sa znakom tačke na kraju.

Alati za analizu DNS sadržaja

Iako DNS funkcioniše kao i svaki drugi protokol na aplikativnom nivou, zbog važnosti za rad drugih aplikacija, on se može smatrati servisom koji je sastavni deo mrežne infrastrukture. Sistem je distribuiran i redundantan, a za administriranje određenog domena od posebne je važnosti saradnja sa organizacijama koje su nadležne za uparene sekundarne DNS servere i primarni DNS server roditeljskog domena. Ove veze se uspostavljaju pri inicijalnom kreiranju domena i obično se ne menjaju, a često se ne proverava njihovo funkcionisanje. Vremenom mogu da se jave određeni problemi, koji zbog visoke redundantnosti sistema često ostaju prikriveni. Tipične greške ove vrste su sledeće:

- U roditeljskom domenu je definisan nevalidan DNS server za određeni domen. Ovaj server će da se javlja u odgovoru na iterativne upite, ali ne može da razreši nazive u posmatranom domenu, što će izazvati čekanje na *timeout* period, nakon čega će se upiti tražiti od drugih autoritativnih servera, koji vraćaju odgovor. Rezultat je da je u pojedinim slučajevima odgovor značajno usporen, odnosno kada se upit traži od nevalidnog servera.
- U roditeljskom domenu nije definisan validan DNS server koji je autoritativan za određeni domen. Ovaj server se nikada neće javljati u odgovoru na iterativne upite na roditeljskom nivou, što ga praktično čini odsečenim od DNS sistema i funkcionalno neupotrebljivim, što smanjuje redundantnost rada.
- Sekundarni DNS server nije usklađen sa primarnim serverom. Rezultat može da bude da sekundarni server ima zastarele podatke ili čak da izgubi sve podatke iz zone (nakon EXPIRE perioda).

Tipična administracija domena podrazumeva izmene u zoni na primarnom DNS serveru, što je takođe podležno određenim greškama.

- Nakon izmena podataka ne ažurira se SOA serijski broj. Primarni server tada ima nove podatke, ali sekundarni serveri ih neće preuzeti. To dovodi do nestabilnog funkcionisanja, budući da neki korisnici dobijaju ispravne podatke od primarnog servera, dok drugi mogu da dobiju zastarele podatke od sekundarnih servera.
- Čak iako se nakon izmena podataka ažurira SOA serijski broj, potrebno je da prođe određeno vreme (RERESH period) da sekundarni serveri registruju promenu i preuzmu podatke. Iako nije suštinska greška, tokom ovog perioda će se ispoljiti nekonzistentno funkcionisanja kao u prethodnom slučaju. Stoga je kod izmena podataka na primarnom serveru potrebno restartovati sistem, odnosno eksplicitno pokrenuti proces sinhronizacije svih sekundarnih DNS servera.
- Nazivi koji treba da budu u FQDN obliku ne završavaju se sa znakom tačke. Posledica je da se ovi nazivi tretiraju relativno u odnosu na tekući domen, što nije željeni rezultat.

Imajući u vidu navedene probleme koji mogu da nastanu, od velike je važnosti praćenje rada DNS sistema i to sa aspekta podataka koje on vraća. Za ovu namenu se koriste određeni alati koji sprovode različite upite prema DNS sistemu u celini ili određenim DNS serverima.

Na *Linux* operativnim sistemima popularna je komanda *dig*, koja podržava širok skup opcija, za koje vraća rezultat u formi DNS zapisa i dodatnih informacija. Navođenjem bez parametara, vratiće se opšte informacije o *root* DNS serverima, koje obuhvataju nazive i IP adrese (Slika 13.8).

```
[user]$ dig

; <<>> DiG 9.16.1-Ubuntu <<>>
...
;; ANSWER SECTION:
.                       66256   IN      NS      g.root-servers.net.
.                       66256   IN      NS      j.root-servers.net.
.                       66256   IN      NS      e.root-servers.net.
.                       66256   IN      NS      l.root-servers.net.
.                       66256   IN      NS      d.root-servers.net.
.                       66256   IN      NS      a.root-servers.net.
.                       66256   IN      NS      b.root-servers.net.
.                       66256   IN      NS      i.root-servers.net.
.                       66256   IN      NS      m.root-servers.net.
.                       66256   IN      NS      h.root-servers.net.
.                       66256   IN      NS      c.root-servers.net.
.                       66256   IN      NS      k.root-servers.net.
.                       66256   IN      NS      f.root-servers.net.
;; ADDITIONAL SECTION:
g.root-servers.net.     44272   IN      A       192.112.36.4
j.root-servers.net.     324299  IN      A       192.58.128.30
e.root-servers.net.     397513  IN      A       192.203.230.10
...
```

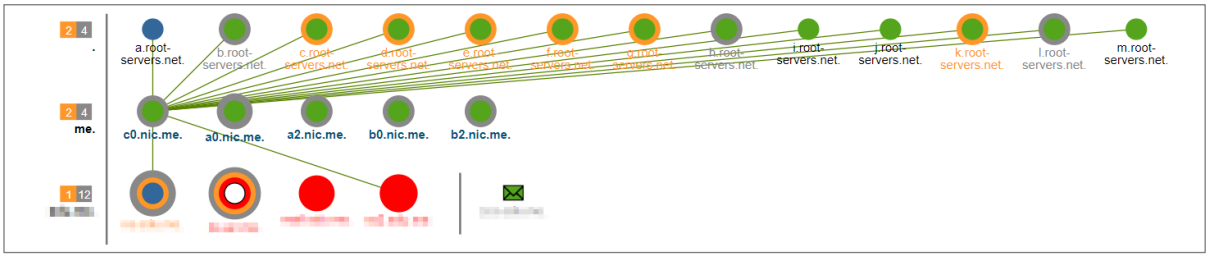
Slika 13.8. Komanda „dig“ za prikaz informacija koje vraća DNS sistem

Navođenjem imena kao parametra, prikazaće se svi relevantni podaci koji odgovaraju navedenom imenu, kao što su IP adresa, eventualni alijasi, autoritativni DNS serveri za ovaj domen i njihove IP adrese (Slika 13.9).

```
[user]$ dig www.etf.bg.ac.rs
;<<>> DiG 9.11.4-P2-RedHat-9.11.4-26.P2.el7_9.10 <<>> www.etf.bg.ac.rs
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 20081
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 4, ADDITIONAL: 6

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.etf.bg.ac.rs.              IN      A

;; ANSWER SECTION:
www.etf.bg.ac.rs.       300     IN      CNAME   proxy-web.etf.bg.ac.rs.
proxy-web.etf.bg.ac.rs. 300     IN      A       147.91.14.227

;; AUTHORITY SECTION:
etf.bg.ac.rs.           300     IN      NS      ns.etf.bg.ac.rs.
etf.bg.ac.rs.           300     IN      NS      ns.rcub.bg.ac.rs.
etf.bg.ac.rs.           300     IN      NS      ns1.nic.rs.
etf.bg.ac.rs.           300     IN      NS      ns2.etf.bg.ac.rs.

;; ADDITIONAL SECTION:
ns.etf.bg.ac.rs.        300     IN      A       147.91.8.6
ns.rcub.bg.ac.rs.       3600    IN      A       147.91.1.5
ns1.nic.rs.             834     IN      A       147.91.8.6
ns2.etf.bg.ac.rs.       300     IN      A       147.91.8.62
ns.rcub.bg.ac.rs.       3600    IN      AAAA    2001:4170:0:1::5
...
```

Slika 13.9. Komanda „dig“ i prikaz rezultata razrešavanja naziva

Analiza rada DNS sistema i uočavanje eventualnih grešaka zahteva sagledavanje rezultata na više servera i domenskih nivoa. Manuelno praćenje ovih tekstualnih informacija je daleko od trivijalnog posla. Bolji pregled stanja DNS sistema pružaju posebni analitički alati, koji su često javno dostupni u onlajn formi. Budući da je DNS sistem dostupan svima, na ovaj način se mogu proveriti podaci o bilo kom domenu. Jedan od takvih alata, koji pruža grafički prikaz veza u DNS hijerarhiji i stanja pojedinih servera, ilustruje Slika 13.10. U navedenom primeru, od tri sekundarna DNS servera, dva su potpuno nedostupna, a treći nije registrovan na roditeljskom nivou, tako da je od četiri autoritativna DNS servera funkcionalan samo primarni server.



Slika 13.10. Onlajn alat za grafički prikaz veza i stanja DNS servera i podataka [51]

Reference

[1] D. W. Davies, „An Historical Study of the Beginnings of Packet Switching“, The Computer *Journal*, Volume 44, Issue 3, 2001, Pages 152-162, https://doi.org/10.1093/comjnl/44.3.152

[2] Cerf V, Kahn R. „A protocol for packet network intercommunication“, IEEE Transactions on communications. 1974 May; 22(5):637-48.

[3] Request for Comments, Internet Engineering Task Force, https://www.ietf.org/standards/rfcs/

[4] „ISO 7498: The Basic Reference Model for Open Systems Interconnection, International Organization for Standardization, May 1983

[5] Carpenter, Brian E. "Is OSI too late?." Computer Networks and ISDN Systems 17.4-5, 1989, 284-286.

[6] Pouzin, Louis. "Ten years of OSI—maturity or infancy?." Computer Networks and ISDN Systems 23.1-3, 1991, 11-14.

[7] Schwartz, M., and N. Abramson. "The Alohanet-surfing for wireless data [History of Communications]." *IEEE Communications Magazine* 47, no. 12, 2009, 21-25.

[8] „IEEE 802.1D-1990: Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges“, IEEE Standards Association, 1990

[9] „ISO/IEC 11801: Information technology — Generic cabling for customer premises specifies general-purpose telecommunication cabling systems“, International Organization for Standardization

[10] „ANSI/TIA-568: Commercial Building Telecommunications Cabling Standard“, American National Standards Institute (ANSI)/Telecommunications Industry Association (TIA)

[11] „IEEE 802.3ad-2000: Link Aggregation Control Protocol (LACP, IEEE Standards Association, 2000

[12] „IEEE Std 802.1w-2001: Rapid Spanning Tree Protocol, IEEE Standards Association, 2001

[13] „IEEE Std 802.1s: Multiple Spanning Tree Protocol“, IEEE Standards Association, 2002

[14] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, https://www.rfc-editor.org/info/rfc791

[15] Eastlake 3rd, D. and J. Abley, "IANA Considerations and IETF Protocol and Documentation Usage for IEEE 802 Parameters", BCP 141, RFC 7042, DOI 10.17487/RFC7042, October 2013, https://www.rfc-editor.org/info/rfc7042

[16] Fuller, V., Li, T., Yu, J., and K. Varadhan, "Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy", RFC 1519, DOI 10.17487/RFC1519, September 1993, https://www.rfc-editor.org/info/rfc1519

[17] Hedrick, C., "Routing Information Protocol", RFC 1058, DOI 10.17487/RFC1058, June 1988, https://www.rfc-editor.org/info/rfc1058

[18] Malkin, G., "RIP Version 2 - Carrying Additional Information", RFC 1723, DOI 10.17487/RFC1723, November 1994, https://www.rfc-editor.org/info/rfc1723

[19] Dijkstra, E. W, "A note on two problems in connexion with graphs", Numerische Mathematik. 1: 269–271. 1959, doi:10.1007/BF01386390.

[20] „ISO/IEC 10589:2002: Information technology — Telecommunications and information exchange between systems — Intermediate System to Intermediate System intra-domain routeing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network service (ISO 8473)“, International Organization for Standardization

[21] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, https://www.rfc-editor.org/info/rfc2328

[22] Plummer, D., "An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware", STD 37, RFC 826, DOI 10.17487/RFC0826, November 1982, https://www.rfc-editor.org/info/rfc826

[23] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, https://www.rfc-editor.org/info/rfc768

[24] Postel, J., "DoD standard Transmission Control Protocol", RFC 761, DOI 10.17487/RFC0761, January 1980, https://www.rfc-editor.org/info/rfc761

[25] Eddy, W., Ed., "Transmission Control Protocol (TCP)", STD 7, RFC 9293, DOI 10.17487/RFC9293, August 2022, https://www.rfc-editor.org/info/rfc9293

[26] Borman, D., "TCP Options and Maximum Segment Size (MSS)", RFC 6691, DOI 10.17487/RFC6691, July 2012, https://www.rfc-editor.org/info/rfc6691

[27] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, https://www.rfc-editor.org/info/rfc1122

[28] Karn, P. and C. Partridge, "Improving Round-Trip Time Estimates in Reliable Transport Protocols", SIGCOMM 87, 1987.

[29] Borman, D., Braden, B., Jacobson, V., and R. Scheffenegger, Ed., "TCP Extensions for High Performance", RFC 7323, DOI 10.17487/RFC7323, September 2014, https://www.rfc-editor.org/info/rfc7323

[30] Stevens, W., "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", RFC 2001, DOI 10.17487/RFC2001, January 1997, https://www.rfc-editor.org/info/rfc2001

[31] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, https://www.rfc-editor.org/info/rfc5681

[32] Iyengar, J., Ed., and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, https://www.rfc-editor.org/info/rfc9000

[33] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, DOI 10.17487/RFC2616, June 1999, https://www.rfc-editor.org/info/rfc2616

[34] Berners-Lee, T. and D. Connolly, "Hypertext Markup Language - 2.0", RFC 1866, DOI 10.17487/RFC1866, November 1995, https://www.rfc-editor.org/info/rfc1866

[35] „ISO 8879: Information processing — Text and office systems — Standard Generalized Markup Language (SGML)“, International Organization for Standardization

[36] Postel, J. and J. Reynolds, "File Transfer Protocol", STD 9, RFC 959, DOI 10.17487/RFC0959, October 1985, https://www.rfc-editor.org/info/rfc959

[37] Postel, J., "Simple Mail Transfer Protocol", STD 10, RFC 821, DOI 10.17487/RFC0821, August 1982, https://www.rfc-editor.org/info/rfc821

[38] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, DOI 10.17487/RFC5321, October 2008, https://www.rfc-editor.org/info/rfc5321

[39] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, https://www.rfc-editor.org/info/rfc5322

[40] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, DOI 10.17487/RFC2045, November 1996, https://www.rfc-editor.org/info/rfc2045

[41] Myers, J. and M. Rose, "Post Office Protocol - Version 3", STD 53, RFC 1939, DOI 10.17487/RFC1939, May 1996, https://www.rfc-editor.org/info/rfc1939

[42] Melnikov, A., Ed., and B. Leiba, Ed., "Internet Message Access Protocol (IMAP) - Version 4rev2", RFC 9051, DOI 10.17487/RFC9051, August 2021, https://www.rfc-editor.org/info/rfc9051

[43] Postel, J. and J. Reynolds, "Telnet Protocol Specification", STD 8, RFC 854, DOI 10.17487/RFC0854, May 1983, https://www.rfc-editor.org/info/rfc854

[44] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, DOI 10.17487/RFC4253, January 2006, https://www.rfc-editor.org/info/rfc4253

[45] Harrenstien, K., Stahl, M., and E. Feinler, "DoD Internet host table specification", RFC 952, DOI 10.17487/RFC0952, October 1985, https://www.rfc-editor.org/info/rfc952

[46] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, https://www.rfc-editor.org/info/rfc1034

[47] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, https://www.rfc-editor.org/info/rfc1035

[48] Sullivan, A., Thaler, D., Klensin, J., and O. Kolkman, "Principles for Unicode Code Point Inclusion in Labels in the DNS", RFC 6912, DOI 10.17487/RFC6912, April 2013, https://www.rfc-editor.org/info/rfc6912

[49] Postel, J., "Domain Name System Structure and Delegation", RFC 1591, DOI 10.17487/RFC1591, March 1994, https://www.rfc-editor.org/info/rfc1591

[50] „ISO 3166-1:2020(E): Codes for the representation of names of countries and their subdivisions — Part 1: Country code“, International Organization for Standardization

[51] NetVizura DNS Checker, http://netvizura.netflow-analyzer.com/dns/

CIP – Каталогизација у публикацији
Народна библиотека Србије, Београд

004.7(075.8)

ГАЈИН, Славко, 1968-
Osnovi računarskih mreža / Slavko Gajin. - Beograd : Akademska misao : Univerzitet, Elektrotehnički fakultet, 2023 (Gornji Milanovac : Grafoprint). – 166 str. : ilustr. ; 29,7 cm

Bibliografija: str. 163-166.

ISBN 978-86-7466-974-7 (AM)

а) Рачунарске мреже

COBISS.SR-ID 118988297