UNIVERZITET U SARAJEVU
ELEKTROTEHNIČKI FAKULTET

Saša Mrdović

Računarske mreže

Univerzitetsko izdanje

Sarajevo, 2020. godina

Autor: Saša Mrdović
Naziv djela: Računarske mreže
Broj izdanja: I
Izdavač: Elektrotehnički fakultet Univerziteta u Sarajevu

Recenzenti:
Red. prof. dr Samim Konjicija, Elektrotehnički fakultet, Univerzitet u Sarajevu
Vanr. prof. dr Sabina Baraković, Fakultet za saobraćaj i komunikacije, Univerzitet u Sarajevu

DTP: Saša Mrdović

Godina izdanja i godina štampanja: 2020.

CIP - Katalogizacija u publikaciji
Nacionalna i univerzitetska biblioteka
Bosne i Hercegovine, Sarajevo

004.7(075.8)

MRDOVIĆ, Saša
Računarske mreže / Saša Mrdović. - 1. izd. - Sarajevo : Elektrotehnički fakultet, 2020. - XIII, 158 str. : graf. prikazi ; 25 cm

Bibliografija: str. [155]-158.

ISBN 978-9958-629-83-9

COBISS.BH-ID 38769926

Odlukom Senata Univerziteta u Sarajevu broj 01-9-66/20 od 29.4.2020. godine ova publikacija je dobila univerzitetsku saglasnost.

Mimici,
DS.

Predgovor

Ova knjiga napisana je da posluži kao udžbenik na kursu "Računarske mreže", na prvoj godini master studija na Odsjeku za računarstvo i informatiku, Elektrotehničkog fakulteta, Univerziteta u Sarajevu.

Ono što bi se poznavalac literature iz oblasti računarskih mreža odmah zapitao je zašto još jedna knjiga iz ove oblasti. Pošto bih isto upitao i ja navodim razloge koji su mene ponukali da je napišem i zbog čega sam je smatrao potrebnom.

Postoje zaista izvrsni univerzitetski udžbenici iz ove oblasti koji se koriste u cijelom svijetu. Dva koja bih izdvojio i toplo preporučio su "Computer Networking: A Top-Down Approach" autora Kurose i Ross [27] i "Computer Networks: A Systems Approach" autora Peterson i Davie [29]. Ove knjige su napisane na engleskom jeziku. Knjige iz ove oblasti, izvorno napisane, na bosanskom jeziku su rijetke. Relativno su rijetke i one napisane na nekom od nama razumljivih jezika. Trebamo njegovati pisanje na našem jeziku u ovo vrijeme globalizacije, jer je bogatstvo svijeta u njegovim različitostima. Smatram da je jedna obaveza univerzitetskih profesora, da uvode terminologiju novih pojmova iz svoje oblasti nalazeći adekvatne prevode izvornih engleskih riječi. U skladu sa ovom težnjom u knjizi su prevedeni svi pojmovi za koje sam smatrao da je to moguće bez gubitka značenja, uz navođenje originalnog naziva na engleskom jeziku, prilikom prvog pominjanja.

Kurs za koji je knjiga namijenjena je na master studiju i drugi je kurs iz oblasti računarskih mreža. Dio gradiva koji se obrađuje u standardnim udžbenicima, poput gore pomenutih, je već izučavan i nije dio ovog kursa. To je otvorilo prostor i stvorilo potrebu da se temeljitije obrade neke dodatne teme vezane za savremene, u vrijeme pisanja, računarske mreže. Većina knjiga koje se koriste kao udžbenici pokrivaju tradicionalne računarske mreže. To su mreže koje su se pojavile sredinom 20. vijeka i do kraja tog vijeka omogućile uvezivanje cijele zemaljske kugle u globalnu mrežu Internet. U posljednjih desetak godina došlo je do većih promjena u kojim se sva komunikacija prebacila na računarske mreže. Ove mreže su preuzele ulogu telefonskih mreža, televizijskih mreža, pošte i svih drugih sistema komuniciranja. Društvene mreže kao

prilično savremeni fenomen promijenile su način komuniciranja i razmjene podataka. Sve ove promjene uticale su na ulogu računarskih mreža i dovele do potrebe za novim pristupima i protokolima. Ti novi pristupi su, uglavnom, samo djelomično obrađeni u udžbenicima, ali su najčešće opisani u posebnim knjigama. Ni jedno ni drugo nisu u potpunosti adekvatni za korištenje na univerzitetskom kursu kao osnovna literatura. U ovoj knjizi su nove teme obrađene kao zasebna poglavlja o obimu primjerenom za nastavnu jedinku na univerzitetu.

Većina udžbenika ima tematski, a nerijetko i enciklopedijski pristup, iznošenju gradiva u kom se navode bitne činjenice i podaci. Ponekad se od detalja pomalo izgubi fokus i principi kao da ostanu nedovoljno jasno izrečeni. Knjiga iz 2018. "Computer Networking Problems and Solutions" autora White i Banks [54] ima nešto drugačiji, ali vrlo koristan pristup objašnjavanju principa. Ta me je knjiga inspirisala da i ja pokušam nešto slično. Kroz godine predavanja sam shvatio da studenti najbolje prihvate novo gradivo ako shvate koji problem se rješava i koja je ideja iza, dobrog, rješenja. U ovoj knjizi je naglasak na tome koji problem rješava određeni protokol i zašto je to bitno za komuniciranja. Čitaoci će ocijeniti koliko sam uspio u ovoj nakani.

Iz tog razloga knjiga počinje opisom tradicionalnih načina komuniciranja i pitanjima koja su morala biti riješena za njih. Principi rješavanja ovih pitanja nisu se promijenili ni u tradicionalnim računarskim mrežama, a nisu ni u savremenim, kao što vrlo vjerovatno neće ni u nekim budućim mrežama.

Čitavo drugo poglavlje posvećeno je tradicionalnim računarskim mrežama. To je najobimnije poglavlje, jer pokriva tradicionalno gradivo računarskih mreža. Sa druge strane je vrlo kratko za obim problematike koju obrađuje. Ideja je bila da se ukaže zašto su tradicionalne računarske mreže tako uspješne. Predstavljeni su dominantni protokoli na svim mrežnim slojevima i pokazano kako je jednostavnost uvijek izašla kao pobjednik. Kraj tog poglavlja objašnjava razloge za velike promjene koje se dešavaju.

Treće poglavlje opisuje razloge za prve drastične izmjene HTTP i TLS od vremena njihovog nastanka. Predstavljeno je kako su nova rješenja omogućila uštede vremena uz povećanu sigurnost i zašto je to bilo potrebno i korisno.

IoT (*Internet of Things*) odavno nije samo popularna riječ među onim koji prate tehnologiju, već svakodnevnica, kako u poslovnom tako i u privatnom okruženju. Ove nove, ili bolje reći sadašnje, mreže više ne uključuju samo moćne uređaje za obradu podataka, poput računara i mobilnih telefona, već sve uređaje koji i na koji način obrađuju podatke. Kakve promjene to donosi u umrežavanje i šta su nova pitanja na koja treba odgovoriti je tema četvrtog poglavlja.

Tendencija centralizacije obrade podataka koja se ogleda u manjem broju ogromnih društvenih mreža i davalaca *cloud* usluga rezultirala je izgradnjom podatkovnih centara sa ogromnom procesnom moći. Toliku procesnu moć je potrebno uvezati međusobno i sa ostatkom svijeta. Način uvezivanja i zahtjevi su definitivno drugačiji od onih kod tradicionalnih računarskih mreža. Ovi zahtjevi i pristupi njihovom rješavanju su obrađeni u poglavlju pet.

Posljednje poglavlje bavi se najvećom promjenom u pristupu rješavanju pitanja umrežavanja od njegovog nastanka. Softverski definisano umrežavanje donosi ideje koje su u svijetu računarstva omogućile otvaranje i ubrzan razvoj hardvera, operativnih sistema i softvera. Ovo otvaranje je dobro došlo i potrebno je za dalji razvoj umrežavanja. U sklopu svakog poglavlja su referencirani standardi i knjige koje dobro pokrivaju i proširuju znanja iz oblasti poglavlja.

Želio bih se zahvaliti recenzentima knjige profesorima Samimu Konjiciji i Sabini Baraković za korisne savjete koji su učinili da konačna verzija ove knjige bude bolja od one koju su oni prvobitno dobili. Roditeljima hvala što su imali strpljenja za milione mojih pitanja kad sam bio dijete. To mi je omogućilo da danas i ja mogu ponekom odgovoriti na poneko pitanje sa ovom knjigom. I naravno najveća zahvala je mojoj LAMS porodici bez koje sve ovo ne bi bilo moguće niti bi imalo smisla. Volim vas najviše na svijetu.

Sarajevo, maj 2020. *Saša Mrdović*

Sadržaj

1

Tradicionalne ne-digitalne komunikacije

Ljudi su vrlo davno počeli da komuniciraju. Sposobnost dobre i efikasne komunikacije doprinijela je razvoju ljudske vrste i društva. Tokom ovog dugog perioda razvoja komunikacija susreli smo se sa različitim otvorenim pitanjima koja smo uspješno riješili. Tako su izgrađene osnove na kojim su zasnovani svi vidovi komunikacija od prvih do najsavremenijih. Principi komunikacija su ostali isti, a mijenjala se tehnologija. Pojavljivala su se i nova pitanja sa novim načinima komuniciranja, ali su uvijek rješavana primjenom dobro poznatih principa. To nas uči da će i buduće komunikacije biti zasnovane na istim principima. Razumijevanje ovih principa pomaže stvaranju trajnog znanja koje će omogućiti upotrebu najsavremenijih i razvoj novih načina komuniciranja.

Računarske mreže su, u raznim oblicima i izvedbama, suštinski preuzele ulogu omogućavača svih oblika komunikacije. Od nekadašnjeg prenosa isključivo podataka, danas se koriste za prenos govora, muzike, slika, video snimaka, očitanja temperature, izdavanje komandi, ... Moguće je napraviti različita predviđanja o budućnosti računarskih mreža. Jedno koje je sigurno je da će se u narednom periodu ubrzano razvijati i mijenjati. Da bi se pratio, i podržavao taj razvoj, potrebno je razumjeti šta su osnovna pitanja komunikacija i principe koji se koriste da se odgovori na njih. Onda je samo pitanje kako se neki od principa i rješenja zasnovanih na njemu izvode u aktuelnoj tehnologiji.

U ovom poglavlju napravljen je pregled tradicionalnih načina ljudskog komuniciranja. Cilj poglavlja je da ukaže na pomenuta pitanja na koja je potrebno odgovoriti da bi se ostvarila uspješna komunikacija. U svakom od navedenih načina komuniciranja pitanja su riješena na različit, ali principijelno sličan način. To potvrđuje tezu da odavno znamo odgovore na pitanja kako omogućiti komunikaciju, samo treba da ih primijenimo na savremenu tehnologiju.

1.1 Verbalna komunikacija

Ljudi uvijek komuniciraju na daljinu.

Osim kad pričaju sami sa sobom. Pričanje sa samim sobom, ma koliko čudno i uobičajeno bilo, je dobar početak razmišljanja o komunikaciji. Ako se pretpostavi da komunikacija služi za razmjenu ideja i razmišljanja, onda da bi smo komunicirali, moramo svoje misli dostaviti drugoj strani u komunikaciji. Samo ako smo ta druga strana mi sami nema potrebe za slanje misli. One su u svom izvornom obliku direktno dostupne objema stranama u komunikaciji. U svim drugim slučajevima moramo drugoj strani nekako omogućiti da vidi sadržaj naših misli (onih koje želimo, ali o tome kasnije). To omogućavanje je suština ljudske komunikacije. Ostvarivanje te komunikacije je daleko od jednostavnog, ma kako to nama, uglavnom, lako išlo. Postoji veći broj pitanja koja se trebaju riješiti. Ono što je ovdje bitno je da su ta pitanja gotovo ista kao pitanja koja treba riješiti u savremenim digitalnim komunikacijama i računarskim mrežama koje ih omogućavaju.

Kada bi telepatija bila moguća (ili bar jednostavna i dostupna svima, za one koji u nju vjeruju) onda bi komunikacija bila mnogo jednostavnija. Naše misli bi u izvornom obliku dolazile do onog sa kim komuniciramo i on bi ih odmah razumio. Kad bi smo to pogledali malo dublje sa stanovišta medicine i fizike, došli bi do teških pitanja o tome kakav je to izvorni oblik misli i kako one u tom izvornom obliku mogu putovati do primaoca. Kako ova diskusija ne bi doprinijela razumijevanju računarskih mreža, odnosno imala bi sasvim suprotan efekat, vratićemo se na uobičajenije načine komunikacije.

Krenimo od najjedostavnijeg slučaja verbalne komunikacije između dvije osobe koje govore isti jezik. Da bi smo svoje misli iskazali kao govor moramo ih pretvoriti u glasove koji čine riječi koje onda čine rečenice. Rečenice imaju svoju strukturu i pravila formiranja, koja se i u gramatici naziva sintaksa, kako bi onom ko ih čuje mogle prenijeti sadržaj, odnosno značenje, što se naziva semantikom jezika. To je prvi korak, pretvaranje misli iz oblika u kom se nalaze u našoj memoriji/mozgu u format koji je omogućava drugima da ih razumiju. U terminologiji računarskih nauka, pa i računarskih mreža, ovakav proces, kad se odvija na računarima, se naziva ***marshalling***[1]. Izvorna rečenica ili više njih, koje se prenose od pošiljaoca do primaoca se, u računarskim mrežama, naziva **poruka**.

Sad kad su te naše rečenice lijepo formirane, one su još uvijek u nama i potrebno je da nekako dođu do onoga kome su namijenjene. Pošto smo rekli da se radi o verbalnoj komunikaciji, potrebno je da, korištenjem našeg glasovnog aparata, ove rečenice izgovorimo. Izgovaranjem se naše misli uobličene u rečenice pretvaraju u zvuk. Taj zvuk su vibracije medija, uglavnom vazduha, koji se nalazi između nas i slušaoca. Te vibracije dospijevaju do uha slušaoca i tamo se pretvaraju u zvukove koji grade rečenice, koje slušalac razumije. To

[1] U nedostatku adekvatnog prevoda ovaj termin se koristi u svom izvornom obliku na engleskom.

je drugi neophodni korak, utiskivanje rečenica, formata podataka razumljivog drugima, u mediji između pošiljaoca i primaoca. Ovo utiskivanje mijenja fizičke karakteristike medija i prostire se kroz njega, Ovaj proces je analogan procesu koji se u telekomunikacijama naziva **modulacija**.

Ma koliko govornik i slušalac bili blizu, i ma kako tiho šaputali, navedena dva koraka moraju se provesti da bi se ostvarila komunikacija. Slično bi bilo i sa svim drugim oblicima ljudskih komunikacija, poput pisanih, znakovnih, a pogotovo onih potpomognutih tehnologijom. Zato ljudi uvijek komuniciraju na daljinu.

Slušaoci otkrivaju postojanje poruke tako što čuju zvuk, ljudskog glasa. Početak poruke je trenutak kad tišinu zamjeni govor, a kraj je kad opet nastupi, dovoljno duga, tišina. Pitanje otkrivanja da je poruka poslana, utvrđivanja njenog početka i kraja, u računarskim se mrežama naziva **uokvirivanje** (*framing*).

Vrlo često imamo više od jednog sagovornika ili se u prostoriji nalazi više od jedne osobe, a mi želimo saopštiti svoje misli tačno jednoj. U tom slučaju potrebno je na neki način naglasiti ko je namijenjeni prijemnik onoga što govorimo. To uglavnom radimo tako što na početku spomenemo ime željenog slušaoca. Termin koji se koristi za ovaj proces u telekomunikacijama je **adresiranje**.

U prethodnom primjeru bio je jedan primalac, što se u terminologiji telekomunikacija naziva ***unicast***[2]. Ponekad želimo nešto prenijeti svima, koji nas čuju. U tom slučaju ili ne navodimo specifično ime ili na neki način naglasimo da je poruka namijenjena svima. Ovakvo slanje se u terminologiji telekomunikacija naziva ***broadcast***[3]. Postoje i posebne situacija u kojima je poruka namijenjena samo dijelu onih koji je mogu čuti. Tada na neki način naglasimo na koga se poruka odnosi, adresiramo je. Recimo na početku se može reći: "Studenti predmeta Računarske mreže! ...". Slanje grupi slušalaca naziva se ***multicast***[4].

Moguće je da slušalac iz bilo kog razloga ne dobije sve glasove koje čine rečenicu. Razlozi za ovo mogu biti razni. Možda je slušalac previše daleko da bi do njega pouzdano stigli svi glasovi sa glasnoćom sa kojom su izgovoreni. Možda na putu između govornika i slušaoca postoje smetnje, poput drugih razgovora ili zvukova u istoj prostoriji. To su **greške** u komunikaciji, koje se na neki način moraju otkloniti, da bi komunikacija bila uspješna. Ako slušalac nije dobio kompletnu poruku moguće su dvije situacije. Jedna je da i pored nedostajućih glasova slušalac može rekonstruisati izvornu rečenicu. Ljudski jezik je redundantan, odnosno nosi više podataka nego što je neophodno da bi se poruke prenijele [58]. To je u slučaju komunikacija korisna osobina. Recimo

[2] U nedostatku adekvatnog prevoda ovaj termin se koristi u svom izvornom obliku na engleskom.

[3] U nedostatku adekvatnog prevoda ovaj termin se koristi u svom izvornom obliku na engleskom.

[4] U nedostatku adekvatnog prevoda ovaj termin se koristi u svom izvornom obliku na engleskom.

da slušalac čuje: "Ovo je vlo vžno da dbro čjete". Uz manji ili veći napor slušalac može zaključiti koja je bila izvorna, poslana, rečenica. Ovakav način ispravljanja grešaka u prenosu podataka naziva se ispravljanje grešaka unaprijed (**FEC** - *Forward Error Correction*[5]). Mogućnost ispravljanja grešaka na prijemniku zavisi od toga koliko glasova nije došlo do prijemnika ili su došli izmijenjeni. Ako je ovaj broj veći od onog za koji redundantnost jezika omogućava ispravke, poruka neće moći biti rekonstruisana.

To je druga situacija, u kojoj ono što slušalac dobije nije razumljivo. Slušalac može takvu poruku zanemariti (odbaciti). Ako slušalac smatra da je poruka važna i da je želi primiti, ili ako su se govornik i slušalac dogovorili, uspostavili **protokol**, da su sve poruke važne i mora se znati da su ispravo primljene, onda mora postojati način da govornik sazna da slušalac jeste ili nije ispravno primio poruku. Ljudi će uobičajeno u govoru, ako nisu nešto čuli reći nešto poput: "Molim vas ponovite? Nisam vas čuo.". Ovo se u telekomunikacijama naziva **negativna potvrda**. Na osnovu nje će pošiljalac znati da treba ponovo poslati poryuku. Ponekad govornik postavi pitanje slušaocu na koje očekuje odgovor. Ako slušalac samo zanemari nerazumljivu poruku, onda će govornik, nakon nekog vremena, zaključiti da pitanje nije stiglo do slušaoca i postaviće ga ponovo. Vrijeme koje pošiljalac čeka na odgovor ili potvrdu prijema, u računarskim mrežama, naziva se ***timeout***[6]. Nakon isteka vremena čekanja govornik će ponoviti svoje pitanje. Postoji i mogućnost da govornik želi, ako mu je to važno, da mu slušalac potvrdi da je poruku uredno primio i razumio. Obično u tom slučaju na kraju poruke kaže nešto poput: " ... Jeste li me razumjeli?". Potvrdan odgovor slušaoca se naziva **pozitivna potvrda** ili samo **potvrda**.

U razgovoru dvije osobe mogu se javiti još neke poteškoće koje mogu dovesti do grešaka, ali koje nisu izazvane smetnjama. Govornik može govoriti brže nego što je slušalac u stanju da prihvati. Razlozi za ograničenje brzine kojom slušalac može prihvatati rečenice mogu biti različiti. Slušalac može biti neko kome je jezik kojim se govori strani jezik te mu je potrebno više vremena da ga procesira, može istovremeno obavljati još nešto i ne biti potpuno koncentrisan na slušanje ili može zapisivati riječi govornika što je uglavnom sporije od govorenja. U svim ovim slučajevima, za uspješnu komunikaciju neophodno je da govornik ne govori brže nego što to slušalac može prihvatiti. Slušalac može biti taj koji će upozoriti govornika da uspori ili govornik, poput učiteljica kada diktiraju tekst učenicima, može na osnovu iskustva znati kojom brzinom da govori. Ovaj proces usklađivanja brzina slanja i prijema naziva se **kontrola toka**.

Kada je u jednoj prostoriji ili u nekom prostoru unutar dometa ljudskog glasa više govornika za očekivati je da će međusobno stvarati smetnje ako

[5] Zbog svoje preovladavajuće upotrebe ovaj termin se, u nastavku teksta, koristi u svom izvornom obliku na engleskom ili kao skraćenica.

[6] U nedostatku adekvatnog prevoda ovaj termin se koristi u svom izvornom obliku na engleskom.

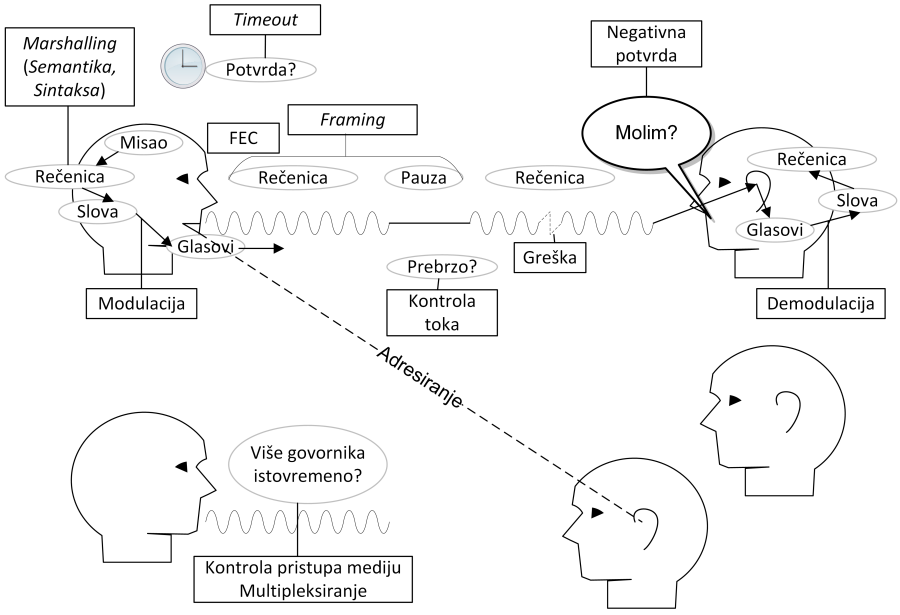
govore istovremeno. Ljudi ovo pitanje rješavaju na više načina. Prvi pristup, koji pada na pamet, je govoriti glasnije. Ipak ispostavlja se da to nije najbolje rješenje, pogotovo ako i drugi govornici koriste isti pristup. Rezultat može biti da zapravo niko nikoga ne čuje. Češće korišten, i bolji pristup, je da potencijalni govornik čeka trenutak tišine i onda počinje da govori. Drugi potencijalni govornici, koji nisu počeli govoriti u tom trenutku tišine, sačekaće naredni i onda početi govoriti. Naravno moguće je i da dva, ili više, govornika počnu pričati u istom trenutku tišine. Tada obično jedan odustane, a drugi kreme iz početka ili svi ljubazno ponude drugima da kažu ono što žele. Ovo ukazuje na treći mogući pristup ovom pitanju. Za dovoljno velike, i uređene, grupe ljudi, obično postoji neko ko uređuje razgovor i daje mogućnost govora u jednom trenutku samo jednom govorniku. Pitanje ko u nekom trenutku može slati podatke po dijeljenom mediju se u računarskim mrežama naziva **kontrola pristupa mediju**.

Potrebno je spomenuti i mogućnost da u prostoriji ima više govornika koji govore istovremeno, ali da ih slušaoci ipak mogu čuti i razumjeti. Ovako nešto se često dešava na javnim mjestima ili zabavama gdje se obično vodi više različitih razgovora, istovremeno. Sa jedne strane, to je moguće jer su obično sagovornici dovoljno blizu tako da je glas onoga sa kim razgovaraju glasniji od drugih glasova. Ljudi su u stanju da, do neke mjere, iz više pomiješanih glasova isfiltriraju onaj koji ih zanima. Ovo se posebno odnosi na izdvajanje govora na jednom jeziku, ako su glasovi koji se miješaju na više jezika. omogućavanje da se po istom mediju šalje više tokova podataka koji su na neki način pomiješani, a da primaoci mogu izdvojiti tokove koji su za njih namijenjeni, u terminologiji računarskih mreža naziva se **multipleksiranje**.

Osnovni elementi verbalne komunikacije, pomenuti u prethodnom tekstu. prikazani su na slici 1.1. Sa slike je očigledno koliko je ova komunikacija daleko od jednostavne, a mi je ipak obavljamo sa lakoćom.

Očigledno da su ljudi, da bi omogućili verbalnu komunikaciju, odgovorili na većinu pitanja koja su aktuelna u savremenim računarskim mrežama. Doduše, ova komunikacija se razvijala kroz desetine hiljada godina, pa je imala vremena da sazri i bude usavršena. Danas vodimo razgovore bez razmišljanja o pitanima koja su ovdje postavljena. I dobro je da je tako. To je i cilj dobrih komunikacija, da su neprimjetne i u pozadini omogućavaju razmjenu ideja.

Međutim domet ljudskog glasa, ma kako glasan bio, ne prelazi stotine metara. Održiva i efikasna verbalna komunikacija, bez pomoći tehnologije, odvija se uglavnom unutar nekoliko metara. Razvoj ljudskog društva nametnuo je potrebu za komunikaciju na veće udaljenosti. Neki od načina prevazilaženja ovog problema bili su zvučna ili vizualna signalizacija. Primjeri zvučne signalizacije, korištene u prošlosti, su bubnjevi i topovi. Primjeri vizualne signalizacije su dimni signali, ogledala za odbijanje sunčevih zraka i zastavice. Ono što je slično za sve ove načine prenosa podataka je da ne prenose misli u obliku rečenica, jer je način prenosa nepogodan za to, već u nekom izmijenjenom obliku, pogodnom za način prenosa. Da bi komunikacija bila uspješna potrebno je da i pošiljalac i primalac znaju protumačiti značenje poslanog



Slika 1.1: Elementi verbalne komunikacije

signala. Pošiljalac misli pretvara u signale, po nekom dogovorenom pravilu. Ovaj proces se naziva **kodiranje**. Primalac primljene signala pretvara nazad u misli koje predstavljaju. Ovaj proces se naziva **dekodiranje**. Ideja da se radi prilagođenje izvornog oblika podataka u oblik pogodniji za slanje po korištenom mediju je od ključne važnosti za razvoj telekomunikacija. To je bilo pitanje koje se nije pojavilo u verbalnoj komunikaciji, a od presudnog je značaja, pa se ovdje pominje. Svi savremeni digitalni sistemi komunikacije na daljinu, pa i računarske mreže, oslanjaju se u svom radu na kodiranje. O tome će biti više riječi u narednom poglavlju.

Ipak, pisana komunikacija je dugo vremena bila dominantan način rješavanja pitanja komunikacije na daljinu. Analiza ovog načina komuniciranja kroz principe i pitanja pomenute kod verbalne komunikacije napravljena je u nastavku. Nova pitanja, koja postavlja novi oblik komuniciranja, i odgovori na njih takođe su analizirani.

1.2 Pisana komunikacija

Pisana komunikacija ima mnogo oblika i kroz istoriju se mijenjala, od prvih crtanja po zidovima pećina, preko urezivanja u stijene, do pisanja po papiru i savremenih, ne-elektronskih, poštanskih komunikacija. Pod pisanim komunikacijama se ovdje podrazumijeva oblik zapisivanja informacija koji ostavlja

vizuelno vidljiv trag. Ono što je zajedničko za sve ove oblike pisane komunikacije je da koriste simbole. Simboli su izražavali ideje, pojmove, događaje ili, u savremenim alfabetima, glasove. Kroz simbole je bilo moguće ispričati priču ili iznijeti neku ideju, ukratko prenijeti misli primaocu. Bitna karakteristika pisane komunikacije je što ostavlja trag komunikacije i nakon što je prenos podataka završen, za razliku od verbalne komunikacije. To ukazuje na bitnu osobinu komunikacija, da pošiljalac i primalac ne moraju biti udaljeni samo u prostoru već mogu biti udaljeni i/ili u vremenu.

Savremeno pisanje, zasnovano na alfabetima, kodira glasove, koji tvore riječi i rečenice sa slovima, jednim ili više znakova. Pošiljalac i primalac dijele alfabet i na jednoznačan način pretvaraju glasove u znakove alfabeta i nazad u glasove.

Način pisanja i tumačenja napisanog definisani su gramatikom jezika koji se koristi, odnosno njegovom sintaksom i semantikom. Kod verbalne komunikacije dio značenja prenosi se i kroz naglasak, izraz lica ili pokret tijela. Kod pisane komunikacije svo značenje dolazi iz onoga što je napisano, jer drugih načina prenošenja značenja, uglavnom, nema.

Utiskivanje rečenica u mediji koji se koristi za prenos, modulacija, je ovdje proces ostavljanja vidljivog traga, u obliku korištenih simbola, na mediji koji se koristi za prenos. Danas je to pisanje slova na papiru. Ovaj proces zahtjeva više mentalnog i fizičkog napora nego govor. Pored toga i traje duže.

Slanje pisanog teksta na daljinu otvorilo je neka pitanja koja ne postoje kod verbalne komunikacije. Jedno pitanje je **kašnjenje**. Vrijeme potrebno da pisana poruka stigne od pošiljaoca do primaoca je duže nego nego kod verbalne komunikacije. Na ovo se dodaje vrijeme potrebno da se poruka napiše i pročita da se dobije ukupno kašnjenje. Koliko je kašnjenje bitno zavisi od toga koliko je bitno da poruka brzo dođe do primaoca, Ono što je očigledno da se dopisivanjem, putem pisama, ne može voditi interaktivan razgovor kao kod verbalnog komuniciranja.

Drugo pitanje, pisane komunikacije, je slanje preko **posrednika**. Engleski termin koji se često koristi u računarskim mrežama je ***proxy***. Ako pošiljalac ne bi koristio posrednika, već sam išao do prijemnika, ne bi bilo potrebe da svoje misli zapisuje, već bi ih mogao direktno usmeno saopštiti prijemniku. Uloga posrednika je da poruku dostavi od pošiljaoca do primaoca. Da bi to ostvario on ne mora, za razliku od primaoca poruke, razumjeti sadržaj poruke. Posredniku je bitno samo da zna kome je poruka namijenjena. To njegov posao čini lakšim. Naravno postavlja se pitanje treba li, ili smije li, posrednik znati sadržaj poruke. Ovo je pitanje sigurnosti koje će biti obrađeno na kraju ovog potpoglavlje.

Posrednik može biti jedan, kao kurir koji prenosi poruke između dva kralja, ali ih može biti mnogo više, kao kod savremenih poštanskih komunikacija pismima. Za pošiljaoca je posrednik, jedan ili više, komunikacioni sistem kom predaje poruku za primaoca. Za primaoca je posrednik, jedan ili više, komunikacioni sistem od kog dobiva poruku od pošiljaoca.

Više posrednika olakšava rad komunikacionog sistema koji ima veći broj pošiljalaca i primalaca. Sada svaki posrednik ne mora znati isporučiti od svih pošiljalaca do svih primalaca. Jedan posrednik treba samo znati od kojih pošiljalaca ili drugih posrednika preuzima poruke i kojim posrednicima ili primaocima predaje poruke. Ovom podjelom smanjena je dimenzionalnost problema za svakog od učesnika, posrednika. Svaki od posrednika od pošiljaoca do primaoca obavlja jednostavniji zadatak. Svaki vrši samo, relativno, lokalnu isporuku unutar ograničenog domena pošiljalaca i primalaca. Ne mora rješavati problem globalne isporuke. Ipak, kroz ovaj niz lokalnih rješenja ostvaruje se rješenje pitanja globalne isporuke. Kod savremenih poštanskih sistema, poštari prikupljaju i raznose poštu, razna prevozna sredstva raznih pošta prevoze pisma do odredišnih pošta, tamo opet poštari raznose pisma unutar ograničenog prostora. Ovakav pristup omogućava da se sistem dobro prilagođava povećanju broja onih koji šalju i primaju poštu. Za veći broj korisnika samo se poveća broj poštara i/ili prevoznih sredstava. Nema potrebe da se mijenja sistem rada ili uloge elemenata sistema. Ova, veoma bitna, karakteristika svakog sistema naziva se **skalabinost**. Skalabilnost računarskih mreža omogućila je da Internet poraste do svoje sadašnje veličine bez potrebe za ozbiljnijim promjenama protokola na kojim je zasnovan.

Posrednik može prenositi poruke od jednog ili više pošiljalaca, za jednog ili više primalaca. Posrednik koji prenosi više poruka odjednom je efikasniji komunikacioni sistem od onog koji prenosi samo jednu. Ako kurir prenosi poruke između više pošiljalaca i primalaca u dva zamka jednu po jednu onda on u vremenu potrebnom da ode od jednog do drugog zamka prenese samo jednu poruku. Ako prenosi više poruka onda je količina informacija koju može prenijeti u istom vremenu veća. Ova, bitna, osobina komunikacionog sistema naziva se **propusnost** (***throughput***) i definiše količinu podataka koji može prenijeti u jedinici vremena.

Prenošenje poruka od više pošiljalaca za više primalaca istovremeno znači da je potrebno spriječiti da se poruke promiješaju ili isporuče pogrešnim primaocima. To je pitanje multipleksiranja pomenuto u govornoj komunikaciji. Takođe je, uglavnom, potrebno omogućiti primaocima da znaju od koga su dobili poruku i kako mogu odgovoriti na nju ili potvrditi njen prijem.

Jednostavno odvajanje pisanih poruka može se ostvariti tako što je svaka poruka na posebnom listu papira. To otvara pitanje šta raditi sa porukama koje su duže od jednog lista papira (što su pisma često bila). Očigledno je potrebno na neki način označiti početak i kraj poruke, odnosno uraditi uokvirivanje (*framing*).

Adresiranje rješava pitanje ko je primalac, a ko pošiljalac poruke. Posrednik, odnosno komunikacioni sistem, koristi informacije o adresama primalaca, ili češće zvanim odredišnim adresama, za isporuku poruka. Adresu pošiljaoca, odnosno izvorišnu adresu, komunikacioni sistem može koristiti da izvijesti o uspješnoj isporuci poruke ili problemima sa isporukom ili da vrati poruku pošiljaocu, ako se ne može isporučiti, kao kod preporučenih pošiljaka.

Poštanska usluga, komunikacioni servis, prenosi pisane poruke od velikog broja pošiljalaca, za veliki broj primalaca, sa više posrednika koji prenose poruke po cijelom svijetu. Da bi ovaj sistem radio potrebno je bilo napraviti neku standardizaciju informacija koje prate pisanu poruku i omogućavaju njenu uspješnu isporuku. Svaki posrednik koji učestvuje u prenosu poruke mora biti u mogućnosti prepoznati i protumačiti minimalno adresu primaoca, a ponekad i druge informacije o poruci. Informacija o adresi zato je napisana na definisanoj lokaciji na pismu ili razglednici u definisanom formatu koji važi u cijelom svijetu za sve pošte. Na pismima postoji mjesto za adresu pošiljaoca (zapravo dva moguća) koje omogućava razlikovanje ove adrese od adrese primaoca. Na razglednicama postoji prostor za pisanje poruke, a kod pisama je **sadržaj poruke**, koji može biti više stranica, u koverti. Time se omogućava uokvirivanje, odvajanje različitih poruka. Ovo je slijedeći nivo *marshalling*-a. Poruka se sad sastoji od podataka o poruci koji omogućavaju isporuku i sadržaja poruke. Podaci o poruci se nazivaju **zaglavlje** (***header***). Ovi podaci se ponekad nazivaju meta-podaci, jer nisu direktno podaci koji se prenose već daju informacije o tim podacima.

Poruke, pisma i razglednice, sa zaglavljima na kojima su adrese pošiljalaca i primalaca mogu biti prenošene zajedno, u istoj vreći, a da pošte mogu odvojiti poruke prema prijemnicima. To je način na koji pošte omogućavaju **multipleksiranje**. Bitna stvar kod pisanih komunikacija je da svaka poruka ima svoje zaglavlje na osnovu kog se može dostaviti na odredište. Više poruka koje predstavljaju dopisivanje, konverzaciju, između dva ista učesnika se šalju pojedinačno sa adresama, a krajnje strane ih stavljaju u kontekst konverzacije.

Adresiranje na globalnom nivou, pored toga što mora biti u definisanom formatu, mora omogućiti globalnu jedinstvenost adresa. Kod verbalne komunikacije u jednoj prostoriji, uglavnom je ime željenog sagovornika dovoljno za jedinstveno adresiranje. Ako ima više osoba sa istim imenom, korištenje prezimena ili nadimka može osigurati jedinstvenost adresiranja. Za globalnu jedinstvenost samo imena očigledno nisu dovoljna. Velika većina imena nije jedinstvena. Pitanje globalne jedinstvenosti poštanski sistem rješava se uvođenjem **hijerarhijskog adresiranja**. Adresa se sastoji od naziva države, grada i njegovog poštanskog broja u toj državi, ulice i kućnog broja u tom gradu, te imena i prezimena primaoca. Ovim se pitanje jedinstvenosti opet vraća u lokalne okvire. Bitno je da su imena svih država različita, da se svi gradovi unutra država zovu različito, da su nazivi ulica u jednom gradu jedinstveni, kao i kućni brojevi u ulicama. Sva ova pitanja se rješavaju administrativno, na odgovarajućem nivou, bez potreba da se razmišlja šire od lokalnog sistema. Istina, postoji, vrlo mala, mogućnost da na jednoj adresi, u jednoj zgradi, pogotovo ako je velika, žive dvije osobe sa istim imenom i prezimenom. To se pitanje rješava dodavanjem sprata ili broja stana na adresu ili lokalni poštar na neki način zna za koga je pismo/razglednica. Ta mogućnost je dovoljno mala da nije predstavljala ozbiljna problem u radu svjetskog poštanskog servisa.

Hijerarhijsko adresiranje podržava podjelu odgovornosti i olakšavanje posla svakom od posrednika, poštara ili lokalnih pošta. Sva pisma koja idu izvan neke

države mogu se skupiti na istom mjestu gdje se razvrstavaju po državama u koje idu. Sva pisma iz jedne države u drugu, pristigla između perioda isporuke, mogu se zajedno poslati iz centra pošta izvorišne u centar pošta odredišne države. Za ovu uslugu su bitni samo podaci o državi odredišta. Ostatak adrese, a pogotovo sadržaj poruka su potpuno nebitni. Slična je situacija i sa isporukom po gradovima unutar države, te po poštanskim oblastima u gradovima. Na kraju poštar koji vrši konačnu isporuku u svoju torbu preuzme pisma na kojima gleda samo ime primaoca, te ulicu i broj. Informacije sa viših nivoa hijerarhije adresa, poput države, grada i poštanskog broja, njemu nisu bitne za lokalnu isporuku. Kako je već rečeno, ovaj način rada u kom svaki od učesnika rješava relativno jednostavan lokalni problem, a skup tih rješenja daje globalno rješenje, omogućio je skaliranje poštanskog sistema do globalnih razmjera. Ovo je postavilo dobre osnove za razvoj globalne računarske mreže, Interneta.

Potreba za adresiranjem koje nije lokalno nametnula je obavezu pošiljaocima da znaju pune poštanske adrese primalaca kojim šalju poruke, a ne samo njihova imena. Kade je primalaca veći broj pošiljaocima pamćenje njihovih adresa može predstavljati preveliki napor. To je izrodilo potrebu za uslugom pretvaranja imena po kojim znamo primaoce u njihove poštanske adrese. Tu ulogu su preuzeli lokalni **adresari** pošiljalaca. Ovo pitanje pretvaranje jednih adresa, npr. lokalnih, u druge, globalne ili adrese na drugom nivou hijerarhije, veoma je bitno za razvoj globalnih komunikacija.

Prethodno je rečeno da se sadržaj pisama obično nalazi u koverti, a adrese (zaglavlja) na koverti. Jedan aspekt ovog kovertiranja je da je sadržaj je potpuno nebitan za poštu, posrednika, i nepotreban za isporuku pisma. Pakovanjem u kovertu se naglašava nebitnost sadržaja koverte i pažnja onoga koji vrši isporuku fokusira se na informacije koje su bitne za isporuku i koje se nalaze na koverti. Uobičajeni naziv za ovaj proces gdje se dijelovi poruke koji nisu bitni za isporuku na neki način sklanjaju od pošiljaoca i apstrahuju je **enkapsulacija**.

Drugi aspekt kovertiranja tiče se sigurnosti sadržaja pisma. Sadržaj kovertiranog pisma nije, lako, moguće pročitati. Ovim se ostvaruje **povjerljivost** poruke. Također, ovaj sadržaj nije, lako, moguće izmijeniti. Ovim se ostvaruje **integritet** poruke. Pomenute osobine su dvije bitne karakteristike sigurne komunikacije, razmjene informacija.

Dobro je ukazati da je pisana komunikacija *unicast*, između jednog pošiljaoca i jednog primaoca. Zbog svoje prirode da je poruka vidljiv trag na trajnom mediju za njeno dostavljanje većem broju primalaca, *broadcast* i *multicast*, bilo bi potrebno izvršiti njeno fizičko kopiranje u više primjeraka, te svaki od primjeraka adresirati na svakog od primalaca. Poštanska usluga standardno ne uključuje mogućnost slanja iste poruke na više adresa.

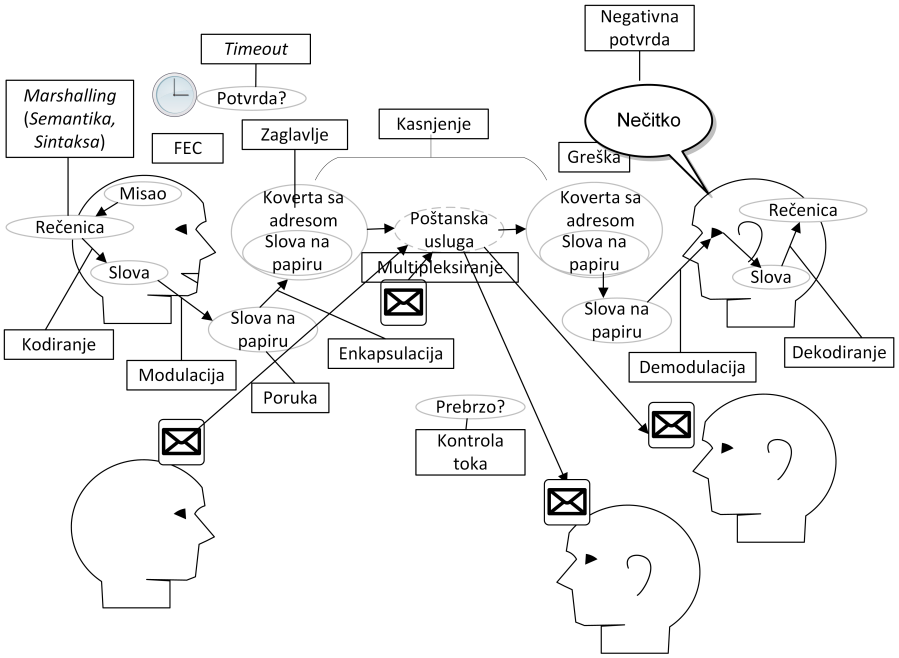
Otklanjanje grešaka u pisanoj komunikaciji je drugačije nego u verbalnoj. Zbog procesa pisanja poruka one nisu podložne slučajnim smetnjama u komunikaciji. Moguće je da neko namjerno nešto dopiše na razglednice, ali to već nije greška u komunikaciji i spada u domen sigurnosti gdje se ova pitanja

rješavaju na drugi način. Poruka kad stigne, stigne onako kako je napisan, osim ako pismo ili razglednica nisu oštećeni tokom slanja. Postoji mogućnost da poruka uopšte ne stigne do odredišta. Način borbe sa ovim zavisi od prirode komunikacije, ali se može oslanjati na iste mehanizme koji su pomenuti za verbalnu komunikaciju, kao što su potvrde i *timeout*, te ponovno slanje. Ono što je bitna razlika je vrijeme koje je potrebno da potvrda stigne, koje je uglavnom mnogo duže nego kod verbalne komunikacije.

Kod pisane komunikacije ne postoji stvarna potreba za kontrolu toka, jer je proces pisanja sporiji od procesa čitanja, a brzina slanja pošiljaoca je dodatno ograničena poštanskim sistemom koji fizički prenosi poruke od pošiljaoca do primaoca, što traje relativno dugo.

Iako pisane poruke dijele mediji, poštara i njegovu mrežu, koji koriste za prenos, kod pisane komunikacije uglavnom nema potreba za kontrolom pristupa mediju. Poruke mogu putovati zajedno bez ometanja jedna druge. Jedino, mada uglavnom teoretsko, ograničenje je kapacitet poštanskog sandučeta za prijem pošte, poštarove vreće ili nekog od sredstava prevoza pošte. Poštanski sistem je iskustveno kapacitirao ove resurse da izbjegne ovu situaciju.

Osnovni elementi pisane komunikacije, pomenuti u prethodnom tekstu. prikazani su na slici 1.2. Sa slike je očigledno koliko je i ova komunikacija daleko od jednostavne, a mi i nju ipak obavljamo sa lakoćom.



Slika 1.2: Elementi pisane komunikacije

1.3 Telefonska komunikacija

Telefonska komunikacija omogućava verbalnu komunikaciju na veće udaljenosti upotrebom telefonskog sistema. Ovaj oblik komunikacije direktno se oslanja na tehnologiju više od pisane komunikacije. Iz tog razloga predstavlja dobar uvod u pitanja komunikacije putem računarskih mreža.

Pošto telefonska komunikacija omogućava verbalnu komunikaciju mnogi aspekti ove komunikacije su isti kao i kod verbalne. Iz ugla pošiljaoca, on svoje misli iskazuje kao govor, a primalac ih čuje kao govor. Sintaksa i semantika su iste kao i kod verbalne komunikacije. Može se reći da ovdje, kao i kod verbalne komunikacije, nema kodiranja. Pošiljalac ne pretvara svoje glasove u posebne simbole, kao kod pisane komunikacije.

Ono što je različito je način kako govor dolazi od pošiljaoca do primaoca. Pošto su oni, uglavnom, udaljeni više nego što je domet ljudskog glasa, taj ljudski glas je potrebno pretvoriti u nešto drugo što može putovati na veću daljinu po korištenom mediju. Tradicionalne telefonske mreže, u terminologiji računarskih mreža najčešće zvane PSTN (*Public Switched Telephone Network*), koriste bakarne žice kao mediji po kom prenose podatke. Govornik govori u slušalicu/telefonski aparat, zapravo u mikrofon na slušalici. Slušalac čuje zvuk govora govornika iz slušalice, tačnije iz zvučnika na slušalici. Komunikacioni sistem telefonske mreže povezuje telefonske slušalice učesnika u komunikaciji. Mikrofon u slušalici pošiljaoca pretvara vibracije membrane uzrokovane govorom u električne signale koje šalje u telefonski sistem. Zvučnik u slušalici primaoca pretvara električne signale koje dobije iz telefonskog sistema u vibracije svoje membrane koje generišu govor koji primalac čuje. Ovdje je modulacija, utiskivanje u mediji podataka koji se prenose, pretvaranje vibracija membrane u električni signal. Kod prijemnika se radi demodulacija, pretvaranje tih električnih impulsa u vibracije membrane, zvuk govora. Kao i kod pisane komunikacije, modulacija je omogućila da se poruka pošalje na veću udaljenost. Modulisani električni signali, koji predstavljaju govor, putuju po izolovanoj bakarnoj žici. Bakar je dobar provodnik električne struje, što znači da električni signal koji uđe na jednu stranu žice, može uz prihvatljivo slabljenje, koje ne sprečava demodulaciju, izaći na drugu stranu žice čija dužina može biti desetak kilometara, što je mnogo više od dometa ljudskog glasa. Izolacija žice smanjuje vanjski uticaj na modulisani električni signal čime smanjuje smetnje. Izolovana žica koja se koristi za telefonske komunikacije naziva se telefonska parica jer je čini par upredenih žica unutar izolacije.

Ovaj osnovni način rada omogućava povezivanje dva učesnika na daljinu do 10 kilometara. Pozabavimo se prvo pitanjem omogućavanje telefonske komunikacije većem broju učesnika, unutar dometa žice. Naivan pristup bi bio da između svaka dva učesnika postoji posebna žica koja samo njima omogućava komunikaciju. Ovo rješenje je skupo, troši mnogo žice, i ne skalira dobro. Za svakog novog korisnika potrebno je razvući žicu do svih postojećih korisnika. Logično rješenje, koje se i koristi u praksi, je da svi korisnici budu povezani na jednu centralnu tačku na kojoj se može vršiti **prespajanje** (***switching***)

između parova koji treba žele da razgovaraju. Ovo prespajanje, koje se često naziva i **komutacija**, su prvo radili operatori (osobe), kako se može vidjeti u filmovima. Operatori i kablovi imaju ulogu posrednika u komunikaciji. Operatori su kratkim kablom povezivali kabl koji dolazi od pozivaoca sa kablom koji vodi do pozvanog. Time se zatvarao krug između strana u komunikaciji. Po tom električnom krugu su mogli da putuju električni impulsi koji predstavljaju ljudski glas. Ovaj način povezivanja učesnika u komunikaciji se naziva **komutacija krugova** (***circuit switching***).

Komutacija krugova, koja se koristi u telefonskoj komunikaciji, omogućava da kad se veza jednom uspostavi (povežu se kablovi pozivaoca i pozvanog) po mediju se šalje samo sadržaj poruka. Pošto mediji direktno povezuje samo učesnike u komunikaciji nema potrebe za adresiranjem svakog dijela govora koji se upućuje drugoj strani. Ovo je različito od pisane komunikacije u kojoj je svako pismo moralo imati adresu, jer je putovalo skupa sa drugim pismima. Kod telefonske komunikacije prenos poruka je efikasniji, jer ima manje zaglavlja, ali je potrebna uspostava konekcije. Kod pisane komunikacije nema uspostave konekcije, ali su potrebne adrese na svakoj poruci. Takav pristup, gdje se svaka poruka pakuje i adresira prije slanje, se, prigodno, naziva **komutacija paketa** (***packet switching***). Postojanje električnog kruga između dva učesnika u komunikaciji znači da se po kablovima koji vode od pozivaoca i do pozvanog, tokom razgovora, ne mogu slati poruke za druge. Mediji je tokom poziva zauzet samo za taj poziv. Kod pisane komunikacije poštar iz jedne kuće može ponijeti više pisama za različite primaoce i donijeti više pisama od različitih pošiljalaca odjednom. Ove razlike između komutacije krugova i paketa su bitne za razvoj računarskih mreža.

Centralna tačka u kojoj se vrši povezivanje naziva se, prigodno, **centrala** (***switch board***). Operatori su vršili povezivanje na osnovu informacija koje su dobili od pozivaoca. Te informacije su predstavljale adresu odredišta sa kojim se želi komunicirati. Pozivalac bi, zapravo, prvo uspostavio vezu sa operatorom, podizanjem slušalice na svom telefonskom aparatu. Zatim bi po toj vezi rekao operatoru sa kim želi da razgovara. U maloj sredini, u kojoj operator poznaje sve korisnike centrale, bi bilo dovoljno reći ime (i prezime) onoga koga se poziva. Pošto ovo nije realno očekivanje, za adresiranje pozivaoca se koristi broj. Operator, u ovom slučaju, ne mora pamtiti vezu između imena i kablova. Na tabli su svi kablovi označeni brojem i on samo povezuje kabl pozivaoca sa kablom čiji je broj tražen. Bitan aspekt ovog pristupa, koji omogućava bolje skaliranje, je da ne zahtjeva od operatora da pamti informacije i pretvara imena u brojeve. Jedan operator opslužuje veći broj korisnika i ima dovoljno posla da izvrši prespajanje poziva. Teret pamćenja preslikavanja imena u brojeve prebačen je na korisnike, pozivaoce. Kao i kod pisan komunikacije, pozivaoci koriste **imenike** u kojima su zapisana imena i odgovarajući telefonski brojevi. Ove imenike su objavljivale telefonske kompanije i obično su obuhvatale jednu državu. Za zvanje nekoga čiji broj nije u objavljenim imeniku, jer je recimo u drugoj državi, bilo je potrebno doći do njegovog broja, bilo direktno od njega, bilo putem nekog sistema globalne pretrage. Ovo pitanje pretvara-

nja jedne vrste adresa (ovdje imena) u drugu vrstu adrese (ovdje telefonski broj) od velike je važnosti i često je potrebno i u računarskim mrežama.

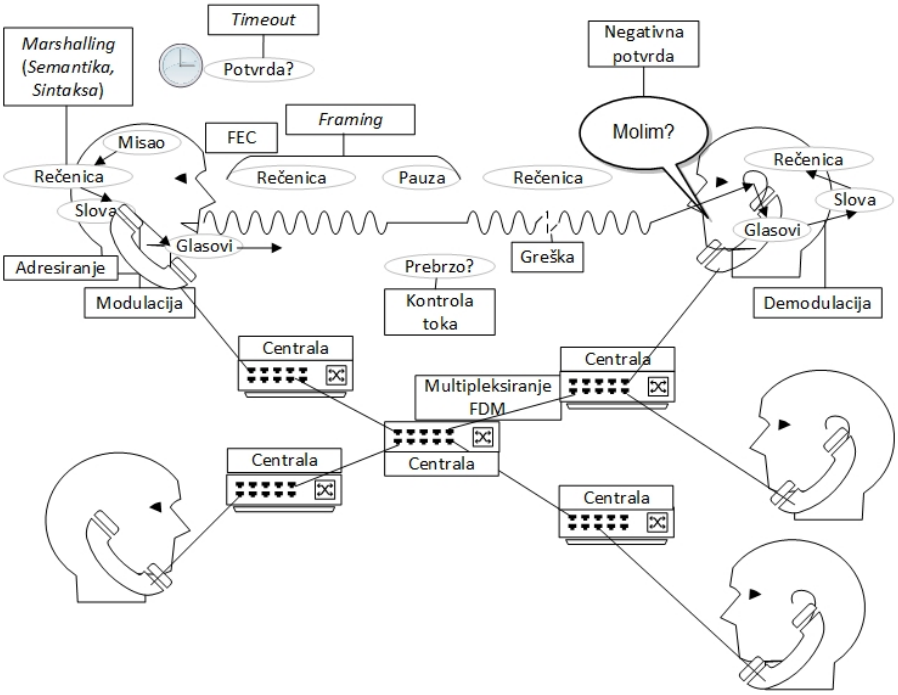
Naredni korak u proširenju, i globalizaciji, telefonske komunikacije je povezivanje sa udaljenim korisnicima koji nisu povezani na istu centralu. Povezivanje telefonskih centrala omogućava ovakvu komunikaciju. Pitanje slabljenja signala na velikim udaljenostima se rješava dodavanjem pojačavača signala na pogodnim mjestima. Najčešće su to centrale. Centrale su međusobno povezane kablovima. Poziv od korisnika jedne centrale do korisnika druge centrale ide od pozivaoca, preko njegove centrale, pa kablom do centrale pozvanog, pa do pozvanog. Centrala pozivaoca i pozvanog ne moraju biti povezane direktno. Veza može ići preko većeg broja centrala. Bitno je da se poziv ispravno prespoji u svakog od njih i omogući stvaranje kruga između pozivaoca i pozvanog. Taj posao odavno ne rade operatori. Centrale su automatizovane i vrše prespajanje na osnovu broja koji je pozivalac birao. Globalna telefonska mreža ima četiri nivoa hijerarhije koji se preslikavaju u telefonske brojeve. Prvi nivo je pozivni broj države, drugi je pozivni broj grada/oblasti, zatim broj koji definiše centralu u toj oblasti, te posljednji broj koji definiše korisnika te centrale.

Postavlja se praktično pitanje, sa koliko kablova treba povezati centrale. Očigledno je da broj kablova određuje maksimalan broj istovremenih poziva između korisnika tih centrala. Veći broj omogućava bolju uslugu, ali podiže cijenu izgradnje i održavanja. Nije za očekivati da će svi korisnici jedne centrale istovremeno zvati sve korisnike druge centrale, pa taj broj može biti mnogo manji. Koliko manji zavisi od procjene planiranog opterećenja koje je uglavnom zasnovano na statističkim istorijskim podacima. Pitanje dimenzionisanja komunikacione infrastrukture da uz minimalne troškove zadovolji potrebe korisnika je ključno za razvoj telekomunikacija. Stalno poboljšavanje rješenja ovog pitanja dovelo je do stalnog razvoja ove oblasti.

Jedno od prvih rješenja bilo je uvođenje drugačijih kablova za povezivanje centrala. Ovi kablovi imali su veći domet i veći kapacitet. To znači da je trebalo manje pojačavača i da se po jednom kablu moglo prenositi više telefonskih poziva odjednom. Prenos više telefonskih poziva po jednom kablu znači da se više ne može koristiti modulacija u kojoj se vibracije membrane direktno pretvaraju u električni signal koji putuje po žici. Potrebno je uraditi multipleksiranje ovih poziva da bi se očuvala njihova odvojenost jednih od drugih dok putuju po istom mediju. Zbog načina kako telefonske rade, nije bilo moguće raditi komutaciju na osnovu adresa poruka, kao kod poštanske komunikacije, jer govor koji se prenosi preko telefonskog sistema nema adrese. Jedan od načina je da se svaki od razgovora prenosi na različitoj osnovnoj frekvenciji. Takvo multipleksiranje se naziva podjela po frekvenciji (**FDM** - *Frequency Division Multiplexing*) Varijacije oko te noseće frekvencije predstavljaju električni signal koji predstavlja govor koji se prenosi. Ovaj način prenosa naziva se širokopojasni prenos ***broadband***. Noseće frekvencije moraju biti dovoljnu udaljene da se u svaku od njih može utisnuti govor. Broj razgovora koji se mogu prenijeti po jednom kablu zavisi od frekventne propusnosti

kabla. U stvarnim telefonskim sistemima se odavno koriti digitalizacija glasa prije slanja i optički kablovi koji mogu prenositi hiljade razgovora odjednom. Postoje i drugi načini multipleksiranja osim FDM. Bitno je napomenuti da, nezavisno od načina multipleksiranja, telefonski pozivi se i dalje šalju komutacijom krugova, jer svaki poziv dobija svoj do medija koji samo taj poziv koristi.

Osnovni elementi telefonske komunikacije, pomenuti u prethodnom tekstu. prikazani su na slici 1.3. To je govorna komunikacija proširena sa komunikacionim sistemom.



Slika 1.3: Elementi telefonske komunikacije

U nastavku će biti napravljena i kratka analiza televizijskog emitovanja. Takvo emitovanje ima svoje posebnosti u odnosu na do sada pomenute načine komunikacije. Savremene računarske mreže preuzele su i ovu ulogu na sebe, pa je stoga korisno razmotriti te posebnosti.

1.4 Televizijsko emitovanje

Televizijsko emitovanje, a slično se može reći i za radio i druga emitovanja, je jednosmjerna komunikacija.[7] Jedan pošiljalac emituje informacije za više prijemnika. Kod televizije emituje se video sadržaj, a kod radija samo zvuk. Ovakva komunikacija je *broadcast*, pa je i jedan od prevoda ovog termina emitovanje. Pošto je komunikacija jednosmjerna primalac nema mogućnosti da obavijesti pošiljaoca o greškama u prenosu sadržaja. Iz tog razloga, pristup ispravku grešaka zasnovan na povratnim informacijama od primaoca ne može biti primijenjen u ovom slučaju.

Slično kao kod verbalne komunikacije, kod tradicionalne TV ne postoji poseban mediji za prenos, već se koristi zrak. Za razliku od verbalne komunikacije, kroz zrak se ne prenose mehaničke vibracije već elektromagnetni talasi. Slično kao kod telefonske komunikacije poruka, TV slika i zvuk, koji se prenose, moduliraju noseću frekvenciju elektromagnetnih talasa. Ono što je različito kod TV emitovanja je kontrola pristupa mediju. TV signali ne putuju po kablovima koji su međusobno izolovani, već se šire slobodno. Za razliku od glasa koji ima domet desetina metara, elektromagnetni signali koje emituju TV stanice imaju mnogo veći domet. Ovaj domet se dodatno povećava repetitorima koji primaju TV signal i distribuiraju ga dalje. Očigledno je da ako dvije TV stanice koriste istu noseću frekvenciju njihovi signali će se miješati i sprečavati uspješnu demodulaciju. Neophodno je osigurati da svaka TV stanica, a isto se odnosi i na radio stanice, emituje na različitoj, i dovoljno udaljenoj, nosećoj frekvenciji. Ovo je ranije pomenuto multipleksiranje. Ovo pitanje se rješava administrativno. U svakoj državi postoji tijelo, obično se naziva regulatorna agencija za telekomunikacije, koja se bavi dodjelom frekvencija. Frekvencije se dodjeljuju TV i radio stanicama. Frekventni spektar je ograničen, jer nisu sve frekvencije jednako pogodne za prenos slike i glasa na daljinu, a broj stanica raste. Očigledno je da je ovo važan resurs, i potencijalno dobar izvor prihoda za svaku državu. Važno je spomenuti da i operatori mobilne telefonije koriste radio talase za prenos telefonskih razgovora, a danas sve više i podataka. I oni od regulatornih agencija dobivaju frekvencije za sebe i za to plaćaju značajne iznose novca. Međunarodna telekomunikacijska unija (ITU) koordinira raspodjelu frekventnog spektra na globalnom nivo.

Ovdje je pogodno spomenuti da je ITU definisala dio spektra za čiju upotrebu nisu potrebne licence. Taj dio spektra naziva se ISM zbog namjene za industrijske, naučne i medicinske primjene. Na tom dijelu spektra rade, na primjer, WiFi mreže.

Na prijemnoj strani, TV prijemnik na osnovu izbora kanala radi demultipleksiranje, odnosno bira noseću frekvenciju koju će demudulisati i iz nje dobiti sliku koju treba da prikaže na ekranu.

U savremenim izvedbama TV emitovanja, uglavnom se koriste kablovi za dostavljanje sadržaja do korisnika.

[7] Ovo se odnosi na tradicionalno TV emitovanje.

Kodiranje, način pretvaranja slike i zvuka koji se žele prenijeti u signale koji se utiskuju u noseće frekvencije se mijenjao kroz razvoj televizije. Rezultat ovog razvoja je da je potreban sve manji frekventni opseg za prenos jednog TV kanala. Ovim se povećava broj TV kanala koji se mogu istovremeno prenositi na jednom području. Termin koji se koristi za uštede spektra koje su postignute prelaskom na, efikasnije, digitalno kodiranje naziva digitalna dividenda. U praksi se oslobođeni dio spektra ne koristi samo za dodatne TV stanice, nego i za mobilne operatore.

* * *

Ovim je završen pregled tradicionalnih ne-digitalnih komunikacije. Kroz pregled je ukazano na otvorena pitanja koja je trebalo riješiti za različite oblike komunikacija. Ista pitanja je bilo neophodno riješiti za tradicionalne mreže koje će biti obrađene u narednom poglavlju. Što je još važnije ista pitanja su riješena i za savremene računarske mreže, a biće ih potrebno riješiti i za sve buduće oblike komuniciranja. Iz tog razloga je bitno shvatiti principe i ideje na kojim su rješenja zasnovana, jer su oni trajni.

2

Tradicionalne računarske mreže

Tradicionalne računarske mreže omogućavaju uvezivanje računara i razmjenu podataka između njih. Može se reći da su pojavile sredinom 20. vijeka i do kraja tog vijeka omogućile uvezivanje cijele zemaljske kugle u globalnu mrežu Internet. Infrastruktura ove globalne mreže i principi prenosa podataka na kojim je zasnovana predstavljaju osnovu za savremene računarske mreže koje su primarni način svih oblika komunikacije danas. Da bi se shvatilo kako je došlo do ove konvergencije, korisno je razmotriti kako su tradicionalne računarske mreže omogućile globalno uvezivanje različitih računara, sistema veze i lokalnih mreža u cjelovit funkcionalan sistem. Očigledno je da su morale riješiti sva otvorena pitanja koja su rješavali tradicionalni ne-digitalni sistemi komunikacije pomenuti u prethodnom poglavlju. U nastavku ovog poglavlja dat je kratak pregled pristupa koje tradicionalne računarske mreže koriste i koji su se pokazali skalabilnim za globalni rast i dodavanje novih usluga.

2.1 Globalizacija i pitanje raznolikosti

I prije pojave Interneta postojali su sistemi komuniciranja koji su se prostirali na cijeli svijet, poput ranije pomenutog poštanskog i telefonskog sistema. Ono u čemu se ovi sistemi razlikuju od računarskih mreža je što su prilično uniformni. Pisma i razglednice, a posebno adrese na njima, izgledaju vrlo slično, svugdje u svijetu. Lokalno preuzimanje i dostavu pošte i dalje rade poštari, svuda na isti način. I ostatak poštanskog sistema je prilično uniforman. Telefonski aparati su se ponešto mijenjali, ali su svi u principu isti jer samo pretvaraju glas u električni impuls i obratno. Pogotovo se svi telefonski aparati na isti način povezuju sa centralama, putem telefonskih parica. Uvezivanje centrala, takođe se radi na standardan način svugdje u svijetu. Računarske mreže, čak i tradicionalne, su mnogo raznolikije.

Korisnički uređaji u računarskim mrežama, računari, nastali su mnogo ranije i korišteni su za mnoge različite stvari, prije nego što su počeli biti uvezivani u računarske mreže. Njihova prvobitna i osnovna namjena nije, kao kod

telefonskih aparata, bila da komuniciraju. Ova mogućnost dodana je tek kasnije već postojećim uređajima. Tačnije, novi računari dobivali su mogućnost komuniciranja, ali se nisu mogli mnogo razlikovati od starih radi kompatibilnosti unazad. Ona je trebala osigurati da postojeći korisnici računara mogu na isti način nastaviti koristiti svoje računare, a da samo dobiju i dodatnu mogućnost komuniciranja. Ti su računari imali različit hardver, različite operativne sisteme i različite instalirane softvere.

Nadalje, prva povezivanja računara bila su ograničena na fizički prostor ili neku organizaciju. Korišteni su različiti kablovi i različiti protokoli za lokalno uvezivanje. Ovi različiti protokoli su na različit način rješavali pitanja ostvarivanja komunikacije opisana u prethodnom poglavlju. Za globalno uvezivanje bilo je potrebno prihvatiti ove različitosti, a ipak omogućiti razmjenu podataka između njih. Ta situacija je zapravo bila dobra, jer je računarske mreže od početka učinila otvorenim i prilagodljivim različitim mrežama i načinima uvezivanja. To je stimulisalo razvoj novih tehnologija i protokola, jer je sistem osmišljen da ih očekuje, prihvata i uklapa u postojeću mrežu.

Da se ne bi stekao pogrešan utisak o velikom broju različitih načina povezivanja i protokola, treba reći da ih je u nekom trenutku uvijek bilo samo nekoliko, ali da su se kroz, relativno kratak vijek računarskih mreža brzo i često mijenjali, a da principi na kojim radi globalna mreža, Internet, nisu morali biti mijenjani. To najbolje pokazuje savremeni Internet koji povezuje milijarde uređaja i korisnika i za koji se očekuje da i dalje raste nesmanjenom brzinom, a također i da radi bez zastoja.

Način na koji je ostvarene ovakva skalabilnost je pristup da se složeni problemi mnogo lakše rješavaju kad se podijele na podprobleme. Dobra podjela omogućava da se svaki od podproblema može rješavati, gotovo potpuno, odvojeno. To čini ukupni problem lakšim za rješavanje. Štaviše, rješenja podproblema se mogu i mijenjati, a da to ne utiče na druge podprobleme.

U računarskim mrežama prihvaćena je činjenica da lokalne mreže mogu biti različite i da je njihovo unutrašnje uvezivanje očigledno različito pitanje od globalnog uvezivanja više lokalnih mreža. Uobičajen naziv za podproblem u računarskim mrežama je **sloj (*layer*)**. Razlog za naziv je što se svako od rješenja podproblema, sloja, oslanja na sloj "ispod", da je taj podproblem nekako riješen, a pruža oslonac, usluge, sloju iznad. U objašnjenju rada poštanskog sistema uvedeni su pojmovi apstrakcije i enkapsulacije. Neka je dijete pošiljalac pisma, a roditelj onaj koji ga adresira i ubacuje u sanduče. Pošiljalac pisma (dijete) brine se samo za to da njegovo pismo može pročitati primalac, ne zanima ga, i ne treba da ga zanima, kako će poruka doći do primaoca. Jedino mu je bitno da je preda odgovarajućem posredniku (ovdje roditelju), sloju ispod, koji adresira pismo. Posredniku sadržaj pisma nije bitan, bitno mu je samo da ga pravilno upakuju u kovertu i dobro napiše adrese. Time je njegov zadatak lakši. Sličan proces odvija se dalje preko poštara, lokalnih pošta i centra pošta. Gornji slojevi apstrahuju slojeve ispod, ne zanima ih kako rade. Slojevi ispod enkapsuliraju ono što dobiju od slojeva iznad, ne zanima ih ni sadržaj ni značenje.

Očigledno je da u računarskim mrežama postoje bar dva sloja. Jedan se bavi pitanjem uvezivanja uređaja u lokalnoj mreži, a drugi pitanjem uvezivanja ovih lokalnih mreža. Rješenjem ova dva pitanja mogu se povezati svi računari na svijetu. To ipak nije dovoljno. Komunikacija u računarskim mrežama treba da omogući korisnicima razmjenu poruka. Korisnici su, makar inicijalno bili, ljudi. Ljudi koriste aplikacije da komuniciraju sa računarom i putem njega sa drugim računarima, aplikacijama na njima, odnosno njihovim korisnicima. Iz ovoga se definišu još dva podproblema, sloja. Jedan se bavi pitanjem kako povezati, omogućiti razmjenu poruka, dvije aplikacije na dva različita računara koja se mogu povezati pomoću dva prethodno pomenuta sloja. Drugi, najviši, se bavi pitanjem kako da aplikacija koju koristi korisnik pošiljalac, napravi poruku koju će razumjeti aplikacija koju koristi primalac i prikaže je primaocu. Ovaj sloj se oslanja na prethodno pomenuti da dostavi poruku do odredišne aplikacije.

Postoji više podjela na slojeve u računarskim mrežama, ali su dvije dominantne: ISO/OSI na teoretskom nivou, a TCP/IP u praktičnoj upotrebi. Prethodno opisana četiri sloja su prema TCP/IP modelu. Ovaj model je toliko dominantan da u praksi potiskuje sve druge modele. Iz tog razloga je izabran kao primjer.

Najviši sloj, koji se naziva aplikativni, bavi se pitanjem kako napraviti jezik (protokol) kojim pričaju aplikacije koje mogu razmjenjivati poruke. Svoje poruke ovaj protokol predaje sloju ispod, koji se naziva transportni.

Transportni sloj bavi se pitanjem kako omogućiti razmjenu poruka između aplikacija na dva povezana računara. Ovaj sloj dodaje svoje zaglavlje i prosljeđuje poruku sloju ispod, koji se naziva mrežni.

Mrežni sloj bavi se pitanjem kako povezati različite, lokalne mreže. Ovaj sloj dodaje svoje zaglavlje na ono što dobije od transportnog sloja i prosljeđuje sloju ispod, koji se naziva podatkovni (*data link*) sloj.

Podatkovni sloj bavi se pitanjem kako povezati uređaje u lokalnoj mreži i omogućiti dostavljanje i prijem paketa sa porukama.

U nastavku poglavlja svaki od navedenih slojeva je dodatno opisan.

2.2 Aplikativni sloj

Uloga aplikativnog sloja je da omogući korisnicima upotrebu usluga računarske mreže. To znači da mogu razmjenjivati informacije sa drugim računarima u mreži. Ta razmjena može biti dohvatanje objekata, kao kod pregleda web dokumenata ili multimedijalnih sadržaja, komunikacija, putem e-pošte, dopisivanja ili razgovora u realnom vremenu ili bilo šta što podrazumijeva korištenje usluga drugih računara u mreži. Da bi ostvarile ovu funkciju mrežne aplikacije prave, adresiraju i šalju poruke. Poruke šalju aplikacijama na drugim računarima. Aplikativni sloj omogućava i prijem tih poruka, njihovu analizu i odgovarajuću akciju u skladu sa sadržajem poruke. Sadržaj poruka, njihov

format i akcije na prijemnoj strani koje izazivaju određene su namjenom aplikacije i protokolom, aplikativnog sloja, koji se koristi.

Poruka aplikativnog sloja ima sadržaj i zaglavlja. Sadržaj poruke, očigledno, mora biti napravljen tako da ga aplikativni sloj primaoca razumije. U govornoj komunikaciji aplikativni sloj bi bio naš govorni aparat koji misli pretvara u izgovorene rečenice na strani pošiljaoca, te slušni aparat koji čuje izgovorene rečenice i pretvara ih u misli ("prikazuje") na strani primaoca. U pisanoj komunikaciji aplikativni sloj su naše ruke koje misli pretvaraju u napisane rečenice na strani pošiljaoca, te naše oči koje vide i čitaju (pretvaraju u misli, "prikazuju") napisane rečenice. U telefonskoj komunikaciji aplikativni sloj našeg govornog i slušnog aparata nadopunjen je telefonskim aparatima koji omogućavaju biranje broja (adresiranje) sa kojim želimo komunicirati. Kod televizijskog emitovanja aplikativni sloj pošiljaoca su kamere u studiju koje prihvataju sliku i zvuk koji se šalju i TV aparati koji tu sliku i zvuk emituju gledaocu.

Aplikativni sloj uz sadržaj poruku dodaje i zaglavlja. Ta zaglavlja su namijenjena primaocu, odnosno aplikativnom sloju primaoca, i pomažu pri prikazivanju poruke na način kako je to pošiljalac namijenio.

Aplikativni sloj ne brine o načinu dostavljanja poruke do krajnjeg primaoca. To je zadatak slojeva ispod. TV kamera šalje sliku, a TV aparat je prikazuje, nezavisno od toga da li je TV signal poslan zrakom ili kablom. Slično ovome telefonski aparati prihvataju od pošiljaoca i reprodukuju primaocu govor nezavisno od detalja telefonskog sistema koji prenosi signala od jednog do drugog aparata. Mi pišemo i čitamo pisma na isti način bilo da su poslana avionom, brodom, kopnom ili da su nam predata ručno. Za pisma, kao i za sve aplikativne slojeve, važno je da su napisana tako da ih primalac može pročitati i da su pravilno adresirana. Ostalo nisu pitanja kojim se aplikativni sloj bavi.

Da bi poruke pošiljaoca bile razumljive primaocu, aplikativni sloj ih mora zapisati u odgovarajućem obliku. Ovo pretvaranje iz izvornog oblika poruke u onaj razumljiv aplikaciji primaocu je, ranije pomenuto, kodiranje. Kodiranje se odnosi kako na sadržaj poruke, tako i na zaglavlja. Kodiranje je definisano aplikativnim protokolom koji se koristi. Protokol, definiše strukturu i pravila formiranja poruka (sintaksa), i njihovo značenje (semantika).

Prvi aspekt kodiranja je skup simbola koji se koristi u porukama. To mogu biti slova, brojevi i znakovi iz alfabeta koji se koristi. Kako većina tradicionalnih protokola potiče iz engleskog govornog područja ovi simboli su obično znakovi engleskog alfabeta. To može biti osnovni skup znakova koje je moguće prikazati, 7-bitni ASCII kao kod SMTP, ili puni skup 8-bitnih ASCII znakova. Internacionalizacijom Interneta sve češće se, u sadržaju poruka, koristi UTF skup znakova koji obuhvata znakove iz svih, ili bar velike većine, alfabeta koji se koriste u svijetu. Kodiranje može uključivati i kompresiju, radi smanjenja opterećenja na mrežu, ili neko drugu transformaciju koju autori aplikativnog protokola smatraju potrebnom. Da bi se osiguralo pravilno dekodiranje simbola na prijemnoj strani u zaglavlju se navodi način kodiranja. Zaglavlja

poruka su još uvijek u osnovnom ASCII skupu radi kompatibilnosti unazad, U novije vrijeme sve više se umjesto znakova koristi binarno kodiranje. Sve manje postoji potreba da ljudi mogu pročitati poruke aplikativnog sloja, što je bila ideja upotrebe znakova alfabeta. Binarno kodiranje je efikasnije, manje bita je potrebno za prenos iste informacije nego kad se koriste znakovi alfabeta.

Za format poruka bitan je i model razmjene poruka (***messaging pattern***) koji se koristi. Ovaj model definiše ko kome i kada šalje poruku. Postoji veći broj ovih modela, ali u upotrebi u tradicionalnim računarskim mrežama dominira model **zahtjev/odgovor** (***request/response***). Kod ovog modela jedna strana šalje zahtjev drugoj. Druga strana odgovara na ovaj zahtjev sa traženim podacima, ili nekim drugim odgovorom ako je potrebno. Moguće su varijacije ovog modela, ali je ovo osnovni princip rada. Kod ovog modela odgovor služi i kao potvrda, pozitivna ili negativna u zavisnosti od sadržaja. Na osnovu odgovora pošiljalac može znati da li je došlo do greške u komunikaciji, i otkloniti je. Model osigurava i kontrolu toka jer omogućava da primalac odgovori kad završi procesiranje zahtjeva. Pošiljalac ne šalje novi zahtjev dok ne dobije odgovor na prethodni. Nedobivanje odgovora nakon isteka vremena čekanja ima efekat negativne potvrde. Model je pogodan za komunikaciju jedan na jedan (*unicast*) koja dominira u tradicionalnim računarskim mrežama.

Sa pojavom IoT, sve više se koristi model **objavi/pretplati** (***publish/subscribe***). Kod ovog modela pošiljalac šalje svoje poruke u trenucima kad ima nešto da pošalje ili kad mu odgovara nezavisno od odgovora primaoca. Nema čekanja na odgovor primaoca prije slanja sljedeće poruke. Primalac čita poruke u trenucima koji njemu odgovaraju i ne mora odgovarati na njih. Izvedbe ovog modela uglavnom podrazumijevaju postojanje posrednika (*proxy*) između pošiljaoca i primaoca. Model je pogodan za komunikaciju više pošiljalaca sa više primalaca (*broadcast, multicast*). O njemu će biti više riječi u poglavlju 4.

Uz model razmjene poruka aplikativni protokoli se mogu razlikovati i po arhitekturi. Dominantna arhitektura u tradicionalnim računarskim mrežama je **klijent/server**. Kod ove arhitekture postoje klijenti koji šalju zahtjeve i serveri koji na njih odgovaraju. Serveri su stalno dostupni na adresama koje su poznate, ili se mogu saznati. Njihova uloga je da odgovaraju zahtjeve klijenta. Klijenti samo konzumiraju usluge servera. Ne moraju biti stalno dostupni niti na stalnim adresama. Serveri šalju poruke klijentima samo kao odgovor na njihove zahtjeve. Odgovori se šalju na adresu klijenta sa koje je došao zahtjev. Tradicionalni Internet je organizovan po ovom principu, a posebno dominantna Internet usluga WWW. Web preglednici, klijenti, šalju zahtjeve web serverima za objektima, na koje ovi odgovaraju slanjem traženih objekata. U ovoj arhitekturi uloge klijenta i servera su jasno razgraničene i čvor je ili klijent ili server.

Druga arhitektura koja je stekla popularnost je **P2P** (***peer-to-peer***). U P2P arhitekturi čvorovi mogu biti i klijenti i serveri. To znači da čvor koji konzumira usluge nekog drugog čvorova istovremeno može pružati usluge trećem

čvoru. Ova arhitektura dobro skalira jer povećanje broja broja klijenata znači i povećanje broja servera. Kod klijent/server arhitekture povećanje broja klijenata povećava opterećenje na servere, kao i na mrežu ka serveru. Poteškoća sa P2P arhitekturom je nestalnost prisutnosti i adresa servera, *peer*-ova, koji pružaju usluge.

Još jedno bitno pitanje aplikativnih protokola je **čuvanje stanja**. Protokoli mogu čuvati stanje ili ne. Čuvanje stanja znači da se razmjena poruka između dvije strane može oslanjati na prethodne poruke iz te razmjene. Svaka poruka u sebi nosi samo nove informacije, a ranije poslane koje su stvorile kontekst se čuvaju u krajnjim čvorovima. Ovo može biti pogodno za konverzacijsku komunikaciju, ali je nepogodno za zahtjev/odgovor klijent/server način rada. Ako jedan server opslužuje mnogo klijenata, što je uobičajeno u tradicionalnim računarskim mrežama, onda on mora čuvati informacije o stanju za svakog od klijenata, što može biti veliko opterećenje za server, Iz tog razloga se u tradicionalnim računarskim mrežama dosta koriste protokoli **bez stanja** (*stateless*). Kod ovih protokola svaki zahtjev klijenta treba u sebi imati sve informacije koje su potrebne serveru za odgovor, bez potrebe za čuvanjem stanja. Ovim pristupom se smanjuje opterećenje na server i prebacuje na klijente obaveza dostavljanja potpunih zaglavlja. Protokoli bez stanja predstavljaju i veće opterećenje za mrežu jer zahtijevaju slanje više podataka. Primjer ovakvog protokola je HTTP.

Ideja arhitekturalnog stila softvera koji ne čuva stanje danas se široko koristi i naziva se **REST** (*Representational state transfer*), Iskazana je u doktorskoj disertaciji Roy Fielding 2000, godine [18]. REST je koncept rada i nije protokol, mada postoji dosta sličnosti sa načinom rada HTTP. Roy Fielding je i prvi autor HTTP RFC. REST softverski arhitekturalni stil podrazumijeva da su klijent i server odvojeni entiteti koji komuniciraju putem jasno definisanih poruka. Ovo znači da se svaki od njih može nezavisno razvijati dok god koriste definisane poruke. Kako je napomenuto, nema čuvanja stanja i svaki zahtjev klijenta ka serveru mora imati sve informacije, od klijenta, koje su potrebne serveru da bi odgovorio. Resursi koje server šalje na osnovu zahtjeva mogu se privremeno čuvati i ponovo koristiti umjesto ponovnog slanja zahtjeva (***cache***-iranje). Odgovor u kom se, na osnovu zahtjeva, dostavlja resurs treba reći da li se taj resurs može ili ne privremeno čuvati. Vrlo bitna karakteristika za REST je uniformnost interfejsa. Ograničenja ovog interfejsa su:

- Identifikacija resursa u zahtjevima - zahtjev na jedinstven način definiše resurs koji traži (URI)
- Manipulacija resursa kroz predstavu o njima - klijent može izmijeniti ili obrisati resurs (poslati zahtjev za izmjenu/brisanja resursa na serveru) na osnovu predstave koju ima o resursu iz onog što mu je server dostavio. Nije mu potrebno ništa više.
- Samo-opisne poruke - svaka poruka ima uz sebe (u zaglavlju) dovoljno informacija da se može pravilno protumačiti i obraditi. Kada server dostavi

resurs uz njega dostavlja informacije i o tipu resursa iz kojih klijent može znati koju aplikaciju upotrijebiti za njegovo prikazivanje.
- Hiper-media za interakciju sa aplikacijom - klijent ne mora znati nikakve posebne komande za rad sa aplikacijom na serveru, jer se sva interakcija sa akcijama i resursima koje aplikacija nudi odvija preko hiper-linkova.

Uz navedeno pripremanja sadržaja poruke i zaglavlja koja omogućavaju njeno tumačenje, aplikativni sloj mora i adresirati poruku. To znači da sloju kom predaje poruku, što je transportni sloj, mora saopštiti za koga je poruka. Ova adresa mora biti jedinstvena u mreži na koju se odnosi. U slučaju Interneta, adresa mora biti globalno jedinstvena. Da bi mogao pravilno pripremiti adresu, aplikativni sloj mora znati način adresiranja koji sistem prenosa, računarska mreža - Internet, koristi. Iako se aplikativni sloj ne mora baviti pitanjem kako će poruka stići, ipak je mora znati adresirati. To je slično kao kod slanja pisama ili telefoniranja, gdje je neophodno pravilno napisati adresu, odnosno birati pravi broj.

Računarske mreže koriste adrese aplikacija koje se sastoje od adrese čvora na kom se izvršava aplikacija i identifikatora aplikacije na tom čvoru. Na javnom Internetu se čvorovi globalno adresiraju na jedinstven način putem IP adrese. Identifikator aplikacije na čvoru je broj porta i transportni protokol. Više detalja o portovima i IP adresama dato je u potpoglavljima o transportnom i mrežnom sloju.

Korisnik aplikacije ne mora poznavati i razumjeti način adresiranja na mreži koju aplikacija koristi za slanje poruka. Takođe korisnik ne bi trebao mijenjati svoj način adresiranja, ako se mreža, ili neki njen dio, koji aplikacija koristi za prenos poruka izmijeni. Ipak korisnik mora biti u mogućnosti na jedinstven način identifikovati kome želi da pošalje poruku. Savremene mrežne aplikacije koriste, u objašnjenju REST pomenuti, URI (*Uniform Resource Identifikator*). URI omogućava identifikaciju čvora na kom se resurs nalazi, identifikator aplikacije/porta na tom čvoru, te identifikaciju resursa koji se želi dobaviti, ako je to potrebno naznačiti.

URI su definisani u RFC3986 [61], gdje se može naći više detalja. Sastoje se od nekoliko dijelova. Svaki od dijelova ima u ulogu u identifikaciji resursa. Nisu svi dijelovi neophodni u svim situacijama. Generička sintaksa URI je:

```
šema://autoritet/putanja?upit#fragment
```

`Šema` i `putanja` su obavezne, mada putanja može biti i prazna. `Šema` u principu definiše korišteni protokol. Izborom `šeme` bira se i sintaksa URI. Protokoli imaju standardne portove na kojima aplikacije koje ih podržavaju očekuju poruke. Na taj način je `šema`, najčešće, identifikator aplikacije na čvoru. `Autoritet` se može sastojati od `korisnika` i `hosta`, te dodatno identifikacije `porta`.

```
autoritet = korisnik@host:port
```

Obavezno je jedino polje `host`, ako se `autoritet` nalazi u URI. `Korisnik` definiše korisnika na `host`-u. `Port` je alternativni način definisanja porta, odnosno, aplikacije na `host`-u. Uglavnom se koristi kad je port drugačiji od onoga podrazumijevanog šemom URI. `Host` je jedinstveni identifikator čvora. To može biti IP adresa, v4 ili v6, ili mnogo češće domensko ime. Domenska imena se putem DNS (*Domain Name System*) pretvaraju u IP adrese. Koriste se jer su, kao hijerarhijski organizovana imena, lakša ljudima za pamćenje od IP adresa, koje su brojevi. `Putanja` određuje adresu, putanju kroz datotečni sistem, do resursa. `Upit` omogućava dostavljanje parametara aplikaciji na osnovu kojih će vratiti specifičniji dio resursa. `Fragment` je podadresa unutar resursa kojom se referencira neki dio resursa.

Primjeri dva standardna i često korištena URI su:

```
http://neka.adresa.ba/folder/datoteka.html
```

```
mailto:korisnik@adresa.ba
```

U prvom primjeru šema je `http`, što znači da se koristi HTTP protokol, opisan niže. Standardni port na kom aplikacije koje koriste HTTP očekuju poruke je 80, pa se ovim definiše port 80 kao identifikator aplikacije na čvoru. `Autoritet` u prvom primjeru je `neka.adresa.ba`. Ovo domensko ime se putem DNS pretvara u odgovarajuću IP adresu. Ta IP adresa na jedinstven način definiše čvor na kom se nalazi aplikacija kojoj se želi poslati poruka. `Putanja` u prvom primjeru je `folder/datoteka.html`. Nju koristi aplikacija koja šalje poruku da navede da je to resurs koji želi od aplikacije kojoj šalje poruku, definisane sa IP adresom i portom.

U drugom primjeru šema je `mailto`, što znači da se koristi SMTP protokol, protokol za slanje poruka e-pošte. Standardni port na kom aplikacije koje koriste SMTP očekuju poruke je 25, pa se ovim definiše port 25 kao identifikator aplikacije na čvoru. `Autoritet` u drugom primjeru je `korisnik@adresa.ba`. Ovim se identificira korisnik na domenu e-pošte. Naziv domena `adresa.ba` se putem DNS pretvara u IP adresu čvora na tom domenu zaduženog za prijem e-pošte. Ta IP adresa na jedinstven način definiše čvor na kom se nalazi aplikacija kojoj se želi poslati poruka. Obavezni dio URI `putanja` je drugom primjeru prazna.

Korisnici mrežnih aplikacija, uglavnom, ne moraju pamtiti i pisati URI. Najčešće su URI dostupni kao hiper-linkovi putem aplikacija. Klikom na hiperlink aplikaciji se naznačava koji resurs korisnik želi. Na osnovu toga aplikacija računa adresu na koju će poslati poruku, a moguće i sadržaj poruke. Za adrese e-pošte često se koriste adresari unutar aplikacija, a za pronalazak WWW resursa korisni su pretraživači. Ove aplikacije igraju ulogu imenika, spomenutih u poglavlju o ne-digitalnim komunikacijama.

U nastavku su dati primjeri dva protokola aplikativnog sloja koja su po mišljenju autora najviše korištena.

2.2.1 DNS

DNS (*Domain Name System*) je sistem koji omogućava povezivanje domenskih imena i odgovarajućih IP adresa. DNS protokol definiše način rada i poruke koje se razmjenjuju tokom rada sistema domenskih imena. Ovaj protokol je inicijalno definisan u RFC 1034 [36] i RFC 1035 [37], te dopunjavan kroz druge RFC. U nastavku je kratko opisan sistem domenskih imena i njegov način rada. Za više detalja potrebno je konsultovati RFC-ove.

DNS je poseban jer ga, uglavnom, ne koriste korisnici direktno, već ga koriste aplikacije. Po tome ne bi pripadao aplikativnom sloju, jer pruža usluge aplikacijama. Ipak, po svom načinu rada je pravi protokol aplikacijskog sloja koji se oslanja na transportni sloj za razmjenu poruka koje omogućavaju njegov rad. Kako aplikacije uglavnom koriste URI da korisnicima omoguće adresiranje, a *host* dio URI je uglavnom domensko ime, aplikacijama su neophodne DNS usluge. To je vjerovatno usluga na aplikativnom sloju bez čijeg rada bi Internet stao.

DNS je napravljen da dobro skalira i bude fleksibilan. Broj registrovanih domenskih imena u 2020. godini je premašio 370 miliona [60]. Sistem koji je za toliki broj zapisa u mogućnosti da brzo pronađe odgovarajuće IP adrese očigledno dobro skalira. Dodavanje, brisanje i izmjene domenskih imena vrši organizacija nadležna za neki dio prostora domenskih imena. Taj prostor se uglavnom prostire na jedan nivo hijerarhije. Na primjer svi poddomeni domena `etf.unsa.ba`, poput `www.etf.unsa.ba` ili `mail.etf.unsa.ba`, definišu se samo na serveru imena `etf.unsa.ba` domena. DNS omogućava da ova promjena bude dostupna svim korisnicima Interneta, što je vrlo fleksibilno.

Sistem domenskih imena organizovan je hijerarhijski. U korijenu je osnovni domen koji se označava sa ".", koja se uobičajeno izostavlja iz domenskog imena. Ostatak domenskog imena sastoji se od naziva poddomena osnovnog domena, odvojenih tačkom i poredanih sa desna na lijevo od prvog poddomena osnovnog domena, pa do posljednjeg. Domensko ime `www.etf.unsa.ba`, sastoji se od naziva vrhovnog domena (TLD - *Top Level Domain*), "ba", njegovog pododmena "unsa", zatim poddomena od "unsa.ba" koji se zove "etf", te konačnog, krajnjeg lijevog, elementa domenskog imena "www", koji obično označava čvor sa IP adresom unutar domena "etf.unsa.ba". Ova hijerarhijska organizacija omogućava spomenutu skalabilnost i fleksibilnost.

Veza između domenskih imena i odgovarajućih IP adresa pohranjena je na serverima (domenskih) imena, koji se često nazivaju i DNS serveri. Ovi serveri odgovaraju na upite klijenata za preslikavanje domenskog imena u IP adrese. Kompletan prostor imena podijeljen je na zone. Zone mogu obuhvatati jedan ili više nivoa hijerarhije. Korijenski (*root*) domen je jedna zona. Vrhovni (TLD) domeni su takođe jedna zona. Ostale zone mogu obuhvatati više nivoa hijerarhije. Svaka zona ima server imena koji je nadležan za tu zonu. Taj

server se naziva autoritativnim serverom za tu zonu. Na tom serveru se nalaze preslikavanja svih domenskih imena iz te zone u IP adrese. Ova preslikavanja se nalaze u datoteci DNS zone (DNS *zone file*). Ako ispod neke zona postoji još domena, koji nisu obuhvaćeni tom zonom, server imena za tu zonu ima adresu servera imena za zonu ispod. Server korijenske zone zapravo samo sadrži IP adrese servera imena za TLD zone. Serveri imena TLD zona imaju IP adrese servera imena za zone ispod, ako postoje.

Zapisi u datoteci DNS zone nazivaju se zapisi o resursima (RR - *resource records*). Pored osnovne uloge povezivanja domenskih imena sa IP adresama, ovi zapisi imaju informacije o serverima imena za tu zonu i podzone, ako postoje, serveru zaduženom za prijem e-pošte za domen, alternativnim domenskim imenima (*alias*), te još neke dodatne informacije. Koji zapis će server vratiti zavisi od vrste i sadržaja upita.

Organizacija prostora domenskih imena u zone omogućila je podjelu dužnosti za održavanje i odgovaranje na upite, po serverima zona. Svaki server ažurira podatke za svoju zonu i odgovara na upite za domenska imena iz svoje zone. Na upite za zone niže u domenskoj hijerarhiji odgovara sa IP adresom servera imena za zonu neposredno ispod njega. U principu pretraga prostora domenskih imena kreće od korijena. Korijenski serveri imena vraćaju IP adrese TLD servera imena za domensko ime iz upita. TLD serveri imena vraćaju IP adrese servera imena za zonu ispod. Ovaj postupak se nastavlja dok se ne dođe do IP adrese autoritativnog servera imena za domensko ime iz upita. Taj server odgovara sa IP adresom koja odgovara domenskom imenu iz upita. Svi serveri imena imaju adrese korijenskih servera imena, tako da uvijek postoji početna informacija za pretragu.

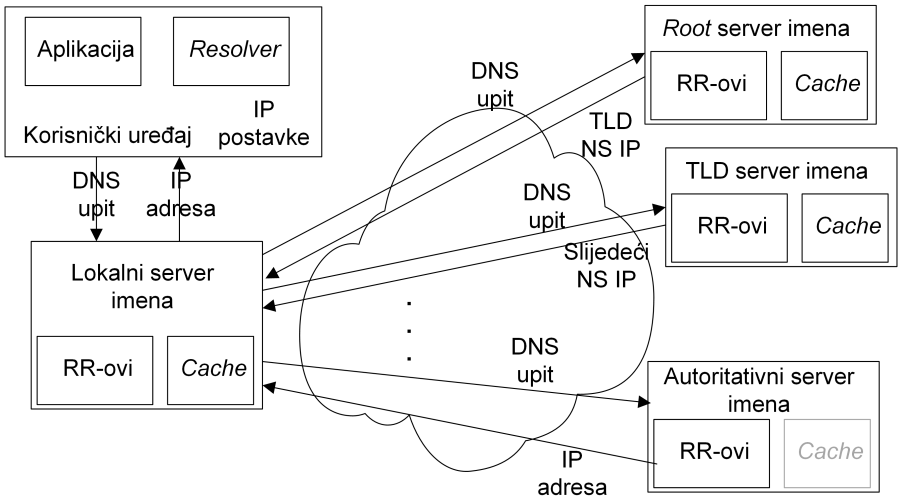
DNS upit zapravo kreće od aplikacije na čvoru koji želi poslati poruku. Aplikacija se obraća procesu na tom čvoru koji je zadužen da odgovara na DNS upite. Taj proces se naziva ***resolver***, i ima ulogu DNS klijenta. DNS klijent iz mrežnih postavki čvora čita IP adresu servera imena kom treba da proslijedi upit. Taj server imena naziva se lokalni, jer je obično najbliži, ali ne mora biti. Lokalni server imena dobavlja odgovor na upit DNS klijenta i vraća ga klijentu. Ako je domensko ime iz upita u njegovoj zoni može odgovoriti odmah. Ako nije obraća se korijenskom serveru imena, od kog dobiva adresu sljedećeg servera imena kome se treba obratiti. Kako je ranije opisano, postupak se nastavlja dok lokalni server imena ne dobije odgovor na svoj upit.

Na svim čvorovima kroz koje upit prolazi, od aplikacije koja ga je inicirala, preko DNS klijenta, lokalnog servera imena i svih ostalih servera imena, pamte se nedavno dobiveni odgovori na upite (*cache*). Ako se neki upit ponovi u periodu definisanom za pamćenje odgovora, onda se na upit odgovara sa zapamćenim odgovorom. Takav odgovor se naziva neautoritativnim, jer nije došao od autoritativnog servera direktno, već je zapamćen raniji odgovor. To ne znači da je odgovor netačan, samo da nije svjež. Uobičajeno vrijeme definisano za pamćenje odgovora, koje se dostavlja u sklopu odgovora, je 24 sata. To se vremenom pokazala kao prava mjera odnosa između smanjenja opterećenje na DNS i svježine odgovora. Zbog ove veličine, vrijeme potrebno da

se izmjena domenskog imena propagira od autoritativnog servera po cijelom Internetu (prebriše stara zapamćena vrijednost) je 48 sati.

Radi pouzdanosti rada DNS, koja je neophodna za rad Interneta, obično postoji više servera imena za jednu zonu. Pogotovo što je zona bliže korijenu. Svi serveri imena za jednu zonu imaju iste informacije, ali dijele opterećenje odgovaranja ili služe kao rezerva. Korijenskih servera imena je 13. Zapravo to je 13 IP adresa koje su putim IP *anycast* distribuirane na više stotina računara.

Rad DNS prikazan je na slici 2.1.



Slika 2.1: Rad DNS

Sve DNS poruke, bilo da su upiti ili odgovori, imaju isti format. Poruka se sastoji od obaveznog zaglavlja, dijela poruke za upite, dijela poruke za odgovore (ako je poruka odgovor), te opcionalnih dijelova sa informacijama o autoritativnom serveru imena i dodatnim informacijama. U zaglavlju su i polja koja definišu tip poruke. Tu je navedeno, putem jednog bita (*QR flag*), da li radi o upitu ili odgovoru. Definisan je tip upita ako se radi o upitu ili dodatne informacije o odgovoru, ako je poruka odgovor. U zaglavlju se nalazi i identifikator koji se generiše za svaki upit, i kopira se u odgovor. U zaglavlju su i polja koja ukazuju koji ostali dijelovi postoje u poruci i koja je njihova veličina. U dijelu poruke za upite navedeni su tip i klasa upita i domensko ime na koje se upit odnosi. Dio za odgovore sastoji se od jednog ili više zapisa o resursima (RR) koji su dobiveni iz datoteke DNS zone kao odgovor na upit. Dijelovi o autoritativnom serveru i dodatnim informacijama, ako postoje su u formatu RR zapisa.

Opšti izgled DNS poruke prikazan je ispod 2.2.

Slika 2.2: Format DNS poruke

Model razmjene poruka kod DNS je zahtjev/odgovor, pri čemu se zahtjev kod DNS naziva upit. Očigledno je da je arhitektura klijent/server. DNS je protokol bez stanja.

Kao i većina tradicionalnih mrežnih protokola DNS nema ugrađene mehanizme zaštite od zlonamjernog ponašanja učesnika u komunikaciji. Ti protokoli su pravljeni da rade između kooperativnih strana i jedine zaštite koje možda imaju su od slučajnih grešaka, uglavnom, u prenosu poruka. Strane u DNS komunikaciji ne provjeravaju ni porijeklo poruke ni njenu neizmijenjenost od trenutka slanja. Poruke nisu zaštićene od prisluškivanja i svi mrežni čvorovi kroz koje prolaze poruke mogu pročitati njihov sadržaj.

Kako je rasla potreba za zaštitom DNS saobraćaja pojavljivali su se novi protokoli. Prvi zvanični standard bio je DNSSEC [45]. Omogućio je zaštitu integriteta sadržaja poruke i autentičnost pošiljaoca. To je postignuto digitalnim potpisivanjem poruka. Upotreba DNSSEC nije, u opštem slučaju, obavezna, ali je od 2013. godine obavezna za TLD servere imena. Upotreba DNSSEC ne štiti povjerljivost DNS poruka, pa su se pojavili drugi protokoli za tu namjenu. Dva koja su aktuelna u vrijeme pisanja su DoT (DNS *over* TLS) [67] i DoH (DNS *over* HTTPS) [39]. Prvi koristi usluge TLS direktno, dok drugi koristi ove usluge posredno preko HTTP. I za jedan i drugi protokol potrebno je da ih DNS serveri koji odgovaraju na upite podržavaju. Drugi se obično ostvaruje uz podršku web preglednika, koja se u vrijeme pisanja pojavila u Mozilla Firefox i Google Chrome. Ovi protokoli nisu bez kontraverzi jer mrežnim administratorima otežavaju nadzor i kontrolu saobraćaja.

2.2.2 HTTP

HTTP je jednostavan protokol namijenjen za dobavljanje web objekata. HTTP klijent šalje zahtjev HTTP serveru i navodi objekat koji želi. HTTP

server šalje u odgovoru objekat, u obliku niza bajta, uz objašnjenje klijentu, u zaglavlju, kako interpretirati objekat.

U odnosu na DNS, rad HTTP je mnogo jednostavniji. Inicijalno, HTTP je bio namijenjen za dobavljanje jednostavnih statičnih objekata koji su bili tekst ili slike. Danas se ovaj protokol koristi za prenos gotovo svih informacija preko weba. Primjer razvoja upotrebe HTTP je *streaming* video sadržaja. Iako HTTP ne bi uopšte trebao biti pogodan za to, njegovom pametnom upotrebom, kroz DASH (*Dynamic Adaptive Streaming over HTTP*) ili HLS (*HTTP Live Streaming*) danas se masovno koristi i za ovu namjenu.

Razlozi za opšte prihvaćanje i masovnu upotrebu HTTP mogu se pronaći u njegovoj jednostavnosti i univerzalnoj dostupnosti. Prevladavajući HTTP klijent je web preglednik koji je dostupan na svim operativnim sistemima i platformama u inicijalnoj postavci. Korisnici ne trebaju instalirati nikakav novi softver. Upotrebom web preglednika mogu konzumirati većinu usluga dostupnih na Internetu. Zbog ovoga je na većini *firewall* omogućen saobraćaj po portovima 80, HTTP, i 443, HTTPS (HTTP preko TLS).

U nastavku je kratko objašnjena, tradicionalna, HTTP 1.1, verzija koja se u vrijeme pisanja još uvijek koristi na više od 50% web lokacija. Više detalja o ovoj verziji HTTP može se naći u RFC-ovima 7230 do 7235 [48][49][47][51][50][46]. HTTP, u posljednjih nekoliko godina, prolazi kroz veće promjene koje nisu ovdje obrađene, jer ne bi spadale u tradicionalne računarske mreže. Promjene i nove verzije HTTP su obrađene u posebnom poglavlju 3.

Iz ranije rečenog može se zaključiti da HTTP koristi klijent/server arhitekturu u kojoj se poruke razmjenjuju po modelu zahtjev/odgovor. HTTP je protokol bez stanja. To smanjuje opterećenje na server, ali se pokazalo ograničavajućim za neke namjene HTTP. To ograničenje je prevaziđeno upotrebom *cookie*. što je kasnije objašnjeno. Identifikacija resursa, koji se u HTTP ponekad nazivaju objekti, radi se upotrebom URI. Za standardne web objekte koristio se URL (*Uniform Reosurce Locator*) specifičan tip URI.

HTTP ima dvije vrste poruka: zahtjev i odgovor. Obje poruke imaju sličan format. Prvi red (*start line*) se, po sintaksi, razlikuje između zahtjeva i odgovora. Nakon prvog reda slijedi jedan ili više redova zaglavlja. Svaki red zaglavlja sastoji se od naziva polja zaglavlja koji je dvotačkom odvojen od njegove vrijednosti. Nakon redova zaglavlja slijedi jedan prazan red koji odvaja, opcionalno, tijelo (sadržaj) poruke od zaglavlja. Tijelo poruke je niz bajta. Zaglavlja omogućavaju tumačenja tijela poruke. HTTP poruke su, u principu, ASCII kodirane. Preciznije, prvi red i zaglavlja poruke su ASCII kodirani, dok tijelo poruke može koristiti i drugo kodiranje koje je onda navedeno u zaglavlju.

Prvi red HTTP zahtjeva, red zahtjeva (*request-line*) sastoji se od metode, cilja zahtjeva (resursa koji se zahtjeva) i HTTP verzije. Metod definiše namjenu zahtjeva iz koje server zaključuje šta klijent želi. Postoji osam metoda (GET, HEAD, POST, PUT, DELETE, CONNECT, OPTIONS, TRACE), ali se najviše koriste dvije. Metoda GET služi da kaže da klijent traži neki

objekat. Metoda POST omogućava klijentu da u tijelu zahtjeva dostavi serveru parametre na osnovu kojih server formira odgovor. Parametri se mogu dostaviti i u cilju zahtjeva GET metode, ali to nije preferirani način rada. Cilj zahtjeva, resurs koji se zahtjeva, je `putanja` iz URI, te `upit` i `fragment`, ako postoje u URI. Prazna putanja iz URI se pretvara u "/".

HTTP zahtjev može imati više zaglavlja, od kojih je većina opcionalna. Jedino obavezno zaglavlje u HTTP 1.1 je `host`. Vrijednost zaglavlja `host` je polje `host` iz polja `autoritet` iz URI. Ostala zaglavlja, kojih ima 20, nisu obavezna. Podijeljena su pet grupa: Kontrolna zaglavlja, uslovna zaglavlja, zaglavlja pregovaranja o sadržaju, zaglavlja sa prijavnim podacima i zaglavlja konteksta zahtjeva. Ovdje su navedena neka od zaglavlja koja se najčešće koriste.

Uobičajeno zaglavlje HTTP zahtjeva je `User-Agent`, čija vrijednost je identifikacija softvera koji je napravio HTTP zahtjev, uglavnom web preglednika. Ovo zaglavlje je iz grupe konteksta zahtjeva. Na osnovu vrijednosti u ovom polju HTTP server može poslužiti verziju objekta najpogodniju za softver koji je tražio resurs, ako server ima više verzija resursa. Danas se to najčešće koristi da se mobilnim uređajima posluži mobilna, a računarima puna verzija web stranice. Zaglavlja pregovaranja o sadržaju poput `Accept`, `Accept-Language` i `Accept-Encoding` omogućavaju da se u HTTP zahtjevu navede preferencija za prihvatljive tipove resursa, preferirani jezik resursa i prihvatljiv tip kodiranja tijela HTTP odgovora, respektivno. Server će udovoljiti ovim zahtjevima ako je u mogućnosti. Ako nije može poslati odgovor da ne može udovoljiti preferencijama ili poslati traženi resurs u obliku u kom može, što se uglavnom radi.

Primjer stvarnog HTTP zahtjeva naveden je ispod.

```
GET / HTTP/1.1
Host: www.etf.unsa.ba
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
    rv:71.0) Gecko/20100101 Firefox/71.0
Accept: text/html,application/xhtml+xml,application/xml;
    q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

Ovaj zahtjev traži od HTTP servera da mu dobavi (GET) svoju početnu stranicu (/) sa web lokacije `www.etf.unsa.ba` (zaglavlje `host`). Web preglednik koji je napravio zahtjev je Mozilla Firefox verzija 71 na Windows 10 64 bitnom operativnom sistemu. Preferirani tipovi resursa su html, xhtml, xml, ali će se prihvatiti i svi drugi tipovi. Preferirani jezik odgovora je američki engleski, ali su i druge verzije engleskog prihvatljive. Prihvatljiva kodiranja su gzip i deflate (zlib).

U zahtjevu se pojavljuju i neka dodatna zaglavlja. Zaglavlje DNT (*Do Not Track*), je proširenje standardnog HTTP 1.1 definisanog sa RFC 7231 i ne mora biti, i nije, podržano u svim web preglednicima. Za vrijednost 1, HTTP serveru se prenosi informacija da klijent ne želi da bude praćen (putem *cookie* trećih strana). Zaglavlje `Connection`, za vrijednost `keep-alive` traži od servera da ne prekida konekciju nakon odgovora jer će uslijediti još HTTP zahtjeva od klijenta (perzistantne konekcije). `Upgrade-Insecure-Requests` zaglavlje, je takođe proširenje standardnog HTTP 1.1. Vrijednošću 1 u ovom zaglavlju se indicira serveru da klijent preferira šifrirani odgovor sa potvrđenim identitetom servera.

Prvi red HTTP odgovora, red statusa (*status-line*) sastoji se od HTTP verzije, brojčanog statusnog koda i tekstualnog opisa statusnog koda. Tekstualni opis statusnog koda služi da ljudi mogu pročitati opis, a HTTP klijent ga treba ignorisati. Statusni kod je trocifreni broj kojim server iskazuje da li je razumio zahtjev i da li ga je mogao ispuniti. Prva cifra definiše klasu odgovora. Klijent mora razumjeti bar klasu odgovora, ako ne i cijeli statusni kod. Klase odgovora definisane prvom cifrom su:

- 1xx (Informacija): Zahtjev primljen, nastavlja se obrada
- 2xx (Uspjeh): Zahtjev je uspješno primljen, protumačen i prihvaćen.
- 3xx (Preusmjeravanje): Potrebno je poduzeti dalje korake da bi se zahtjev ispunio.
- 4xx (Klijentska greška): Sintaksa zahtjeva nije ispravna ili se zahtjev ne može ispuniti.
- 5xx (Serverska greška): Server ne može odgovoriti na zahtjev koji izgleda ispravno.

Neki od najčešćih statusnih kodova su 200 i 404. Statusni kod 200, opisa OK, znači da je zahtjev bio uspješan. Uz ovaj statusni kod server u tijelu HTTP odgovora šalje niz bajta koji predstavlja traženi resurs, ako se radi o zahtjevu za resursom poput GET, ili ono što je odgovor na zahtjev. U zaglavlju tog HTTP odgovora server šalje dodatne informacije potrebne za adekvatno predstavljanje resursa. Statusni kod 404, opisa *Not Found* (nije pronađen), znači da server nije našao resurs, ili ne želi potvrditi njegovo postojanje. Uz ovaj, i druge odgovore čiji statusni kod počinje sa 4, server bi u tijelu poruke trebao poslati objašnjenje uzroka greške

Zaglavlja HTTP odgovora mogu dopuniti statusni kod, upravljati *cache*-iranjem ili dati dodatne informacije o tijelu odgovora. Podijeljena su u četiri grupe: kontrolni podaci, polja zaglavlja za validaciju, zaglavlja sa zahtjevima za prijavne podatke i zaglavlja konteksta odgovora. Ovdje su navedena neka od zaglavlja koja se najčešće koriste.

Uobičajeno zaglavlje HTTP odgovora je `Date` čija vrijednost je datum i vrijeme, na serveru, kad je odgovor poslan. HTTP odgovori uglavnom sadrže i polje `Server`, čija vrijednost je identifikacija softvera, web servera, koji je napravio HTTP odgovor. Polje `Content-Encoding` označava koje kodiranje je urađeno nad sadržajem resursa prije nego što je poslan u tijelu odgovora.

Ako je u ovom polju naznačeno neko kodiranje HTTP klijent treba uraditi dekodiranje prije nego što pristupi tumačenju sadržaja tijela odgovora. Za podršku tumačenju sadržaja tijela HTTP odgovora server koristi polje zaglavlja `Content-Type`. Vrijednost ovog polja definiše format podataka i kako bi ih klijent trebao obraditi. Ovo polje, uz mogućnosti web preglednika, omogućava da savremeni web preglednici mogu prihvatiti i prikazati različite vrste resursa. Na osnovu polja `Content-Type` web preglednik može znati da li sadržaj može otvoriti sam (recimo tip `text/html`), da li će koristiti neki od svojih dodataka ili vanjsku aplikaciju. Recimo, sadržaj HTTP odgovora čija vrijednost `Content-Type` je `application/pdf` može biti otvoren u web pregledniku ili vanjskoj aplikaciji, u zavisnosti od podešenja web preglednika.

Primjer dijela stvarnog HTTP odgovora na gornji HTTP zahtjev naveden je ispod. Kompletno tijelo odgovora bi zauzelo previše prostora za prikazivanje, a u svakom slučaju je nečitko zbog primijenjenog kodiranja.

```
HTTP/1.1 200 OK
Date: Wed, 08 Jan 2020 08:49:18 GMT
Server: Apache/2.2.3 (CentOS)
X-Powered-By: PHP/5.1.6
Content-Type: text/html; charset=utf-8
Set-Cookie: fe_typo_user=a0d2e88cb5; path=/
Vary: Accept-Encoding
Content-Encoding: gzip
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked

1146
..... <.R.I.....c.x....!1+@.e@.H.c..
RU)).ReMU...;b..e.i.'.C.....*...a.^b.8lTy9.kV.....
v^.....k...'

... (previse bajta za ispis) ...

(o.'._.=.a.b....O..p...mvVe._.....
a
....B3.s..
0
```

Server u statusnoj liniji HTTP odgovora potvrđuje da je zahtjev bio uspješan (200 OK). Navodi datum i vrijeme slanja odgovora u zaglavlju `Date`. Identificira se kao Apache web server verzija 2.2.3 na operativnom sistemu CentOS. U tijelu odgovora je html kod, pri čemu je korišten UTF-8 skup znakova, naznačeno vrijednošću zaglavlja `Content-Type`. Tijelo odgovora je kompresovano korištenjem gzip-a, vrijednost zaglavlja `Content-Encoding`.

U odgovoru se pojavljuju i neka dodatna zaglavlja. Server prihvata da ne zatvara konekciju nakon ovog odgovora odgovarajući na `Connection` zahtjev sa istom vrijednošću `Keep-Alive`. Uz ovo zaglavlje dodaje se i zaglavlje `Keep-Alive` koje definiše koliko dugo će server čekati između zahtjeva prije nego što zaključi da treba da ih više neće biti i zatvori konekciju, kao i maksimalan broj zahtjeva koji očekuje prije prekidanja konekcije. Zaglavlje `X-Powered-By`, je uobičajeno, ali nestandardno, koje specificira tehnologiju koju koristi web aplikacija. U ovom slučaju to je PHP verzije 5.1.6. Zaglavlje `Vary` je indikacija posrednicima (*proxy*) kroz koje možda prolazi odgovor, da ako ga *cache*-iraju, čuvaju kodiranu i ne kodiranu verziju, tako da mogu odgovoriti klijentima koji traže jednu ili drugu.

Zaglavlje `Set-Cookie` zahtjeva posebnu pažnju. Ranije je rečeno da je, radi rasterećenja servera, HTTP protokol napravljen da ne čuva stanje. To znači da kad HTTP server odgovori na zahtjev klijenta, on zaboravlja tog klijenta. HTTP server ne može povezati uzastopne zahtjeve od istog klijenta. Za ostvarivanje interaktivnosti koja je danas uobičajena na webu, ovo nije pogodno. Da bi se zaobišlo ovo ograničenje bez promjene HTTP protokola uveden je *cookie*. *Cookie* je niz znakova koje HTTP server dostavlja klijentu u zaglavlju HTTP odgovora `Set-Cookie`. Ovaj niz znakova je različit za svakog klijenta. HTTP klijent, koji je dobio *cookie* od servera, u svim sljedećim zahtjevima serveru dostavlja taj *cookie* (niz znakova) u zaglavlju HTTP zahtjeva `Cookie`. Na osnovu *cookie* server prepoznaje uzastopne zahtjeva od istog klijenta čim omogućava da web aplikacija koja se na njemu izvršava ostvari interaktivnost. *Cookie* pored vrijednosti može imati i atribute koji definišu, ako su vrijednosti različite od podrazumijevanih, koliko dugo vrijedi (*Expire*), na koji domen se odnosi (*Domain*), na koju lokaciju (putanju) se odnosi (*Path*), da li se dostavlja samo u HTTPS zahtjevima (*Secure*), te da li je dostupan skriptama u web pregledniku (*HttpOnly*). Ovo je osnovna namjena i primjena *cookie*, a za više detalja treba pogledati RFC 6265 [11], posvećen mehanizmu upravljanja HTTP stanjem. Za kraj samo komentar da je u gornjem HTTP odgovoru server dostavio *cookie* u navedenom zaglavlju `Set-Cookie`. Uz ovaj *cookie* dostavio je i atribut `path` sa vrijednošću "/". Taj atribut znači da se *cookie* dostavlja u svim zahtjevima za bilo koju putanju na koju se HTTP dostavlja.

Tijelo gornjeg HTTP odgovora sastoji se od niza bajta koji predstavljaju gzip kompresovani sadržaj html datoteke koja je resurs koji se dostavlja.

Protokoli aplikativnog sloja uglavnom nemaju ugrađene sigurnosne mehanizme koji bi poruke zaštitili od zlonamjernih čvorova u mreži. Kada su ovi protokoli nastajali najvažnije je bilo da obavljaju svoju funkciju. Nije izgledalo da postoji potreba za zaštitom. Prije prelaska na transportni sloj potrebno je pomenuti sigurne verzije aplikativnih protokol poput HTTPS. HTTPS je zapravo HTTP preko TLS. Aplikativni sloj poruke šalje TLS, umjesto slanja transportnom sloju. TLS prihvata poruku, procesira je i obrađenu poruku prosljeđuje transportnom sloju. Na prijemnoj strani transportni sloj poruku

ne prosljeđuje aplikativnom sloju već TLS. TLS vraća poruku u oblik u kom je primio TLS pošiljaoca i takvu isporučuje aplikativnom sloju. Upotreba TLS štiti povjerljivost (šifriranjem) i integritet (*hash*-iranjem) sadržaja poruka, te potvrđuje autentičnost servera (upotrebom kriptografije javnog ključa). Ovo je usluga koju radi TLS i aplikativni sloj ne mora mijenjati svoje poruke već ih samo slati TLS umjesto transportnom sloju. TLS se može koristiti, i koristi se, za zaštitu drugih protokola aplikativnog sloja na isti način. Detaljnije objašnjenje rada TLS dato je u poglavlju 3.

Nezavisno od protokola aplikativnog sloja, poruka koja se napravi dostavlja se transportnom sloju od kog se očekuje da je dostavi do aplikativnog sloja na odredišnom čvoru. Aplikativni sloj ne brine o tome kako će poruka doći na odredište. Jedino brine o tome da je poruka pravilno napisana, u skladu sa korištenim aplikativnim protokolom, i pravilno adresirana. Transportni sloj, za aplikativni, apstrahuje mrežu.

2.3 Transportni sloj

Transportni sloj prihvata poruke aplikativnog sloja i dostavlja ih aplikativnom sloju, odnosno odredišnoj aplikaciji na odredišnom računaru. Transportni sloj enkapsulira poruku sa aplikativnog sloja u svoja zaglavlja, bez potreba da vodi računa o sadržaju i formatu te poruke. Za transportni sloj to je niz bajta koje treba dostaviti.

Ni transportni sloj se ne brine o svim detaljima dostavljanja poruke. On se oslanja na mrežni sloj za pronalazak putanje i dostavljanje do odredišnog računara. Ipak postoji detalj nižih slojeva o kom transportni sloj mora voditi računa. Svi protokoli transportnog, mrežnog i podatkovnog sloja imaju maksimalnu veličinu jedinice podataka koju šalju. Obično je ta veličina najmanja na podatkovnom sloju. To znači da podatkovni sloj ne može prihvatiti za slanje više od te veličine, koja se naziva MTU (*maximum transmission unit*). Ako je paket mrežnog sloja koji se predaje podatkovnom veći od MTU, onda ga je potrebno podijeliti u više manjih prije slanja. Da bi se to izbjeglo, koliko je moguće, transportni sloj računa kolika maksimalna veličina onoga što dostavlja mrežnom sloju može biti. Ta računica je relativno jednostavna. Od MTU se oduzme veličina zaglavlja mrežnog sloja i veličina zaglavlja transportnog sloja. Dobivena vrijednost se naziva MSS (*maximum segment size*). Konkretan primjer računanja, koji se najčešće pojavljuje u mrežama, je kada se koristi TCP transportni, IP mrežni i Ethernet podatkovni protokol. Pošto je Ethernet MTU 1500 bajta, IP paket ne može biti veći od toga. TCP sloj ne može IP poslati više od 1500 - 20 bajta (veličina IP zaglavlja). Sadržaj TCP ne može biti veći od 1500 - 20 - 20 (veličina TCP zaglavlja) = 1460, što je MSS u ovom slučaju.

Transportni sloj poruke aplikativnog sloja, koje mogu biti proizvoljne dužine, ako su duže od MSS mora izdijeliti u segmenta ne veće od MSS. Svako

od tih segmenata se šalje kao poseban segment (uobičajeni naziv jedinice koja se šalje na transportnom sloju). Na prijemnoj strani transportni sloj prihvata sve segmente i dostavlja ih aplikativnom redom kojim su poslani, odnosno kao poruku koju je aplikativni sloj pošiljaoca napravio.

Na svakom od slojeva koristi se adresiranje. Na transportnom sloju adrese su brojevi portova koje odgovaraju aplikaciji primaocu i aplikaciji pošiljaocu. Brojevi portova omogućavaju multipleksiranje, jer se njihovom upotrebom može slati više poruka ka istom odredišnom čvoru, ali da one budu za različite odredišne aplikacije na tom čvoru. Port primaoca transportni sloj dobiva od aplikativnog. Port pošiljaoca bira sam. Ovdje je korisno navesti da je transportni sloj izveden kao dio operativnog sistema. Transportni sloj se ne izvodi u mrežnim čvorovima kroz koje segment prolazi, već samo u krajnjim čvorovima, pošiljaocu i primaocu. Operativni sistem primaoca na osnovu broja porta zna kojoj aplikaciji treba da dostavi segment. Da bi to znao aplikacija se ranije morala registrovati kod operativnog sistema da prima segmente na tom portu. Aplikacija primalac je, takođe, trebala obavijestiti aplikaciju pošiljaoca, na neki način, na koji port da adresira poruke. Kod standardnih protokola, kako je rečeno kod opisa URI, brojevi portova su poznati (80 za HTTP, 443 za HTTPS, 53 za DNS, ...). Port može biti naveden i posebno u okviru URI, odvojen dvotačkom od naziva hosta.

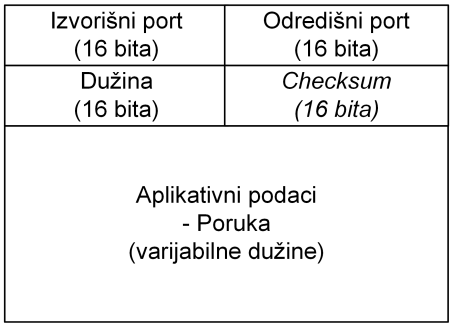
U tradicionalnim računarskim mrežama gdje dominira klijent/server način rada, klijent je taj koji šalje prvu poruku, na poznati (iz URI ili nekako drugo) port. Operativni sistem, odnosno transportni sloj, klijenta generiše slučajan broj porta, od nezauzetih portova, kao port pošiljaoca. Kad server odgovara on šalje sa svog porta na ovaj port. Na osnovu tog broja porta transportni sloj klijenta, originalnog pošiljaoca koji sada dobiva odgovor, zna o kojoj klijentskoj aplikaciji se radi.

Transportni sloj može pružati uslugu pouzdanog prenosa podataka. To može biti na jednostavnom nivou gdje se sa svakim segmentom šalju dodatne informacije koje pomažu u otkrivanju grešaka u prenosu. Pouzdan prenos podataka može osiguravati isporuku svih segmenata i to po redu po kom su poslani. Dva transportna protokola koja se koriste u praksi, UDP i TCP, se razlikuju upravo po nivou pouzdanosti koje nude. Još jedna moguća usluga, dostupna u jednom od ova dva protokola, je kontrola toka kojom se osigurava da pošiljalac ne šalje segmente brže nego što ih primalac može prihvatiti i dostaviti aplikativnom sloju.

2.3.1 UDP

UDP je jednostavniji od dva protokola. Jednostavniji protokol pruža manje usluga. Zapravo UDP pruža najmanji nivo usluga koji transportni sloj može pružati. UDP adresira aplikativnu poruku, dodaje odredišni i izvorišni broj porta. Pored adrese, UDP dodaje još i minimalne podatke za provjeru pouzdanog prenosa segmenta, dužinu segmenta i *checksum* (kontrolni biti koji

omogućavaju otkrivanje nekih izmjena poslanog segmenta na primaocu). Brojevi portova su 16 bitni kod UDP, kao i kod drugog, TCP, protokola. Dužina segmenta i *checksum* su takođe 16-bitni. Ova četiri polja predstavljaju UDP zaglavlje. Format UDP segmenta prikazan je na slici 2.3.



Slika 2.3: Format UDP segmenta

Osim osnovne provjere grešaka na pojedinim segmentima putem *checksum*, UDP ne nudi aplikacijama druge usluge pouzdanog prenosa podataka. Ne nudi garanciju da je segment uredno primljen niti da su segmenti isporučeni onim redom kojim su poslani. UDP ne nudi uslugu kontrole toka. To znači da UDP pošiljalac šalje onom brzinom kojom dobiva podatke od aplikativnog sloja. Aplikacije koje koriste UDP, ako im je potreban pouzdan prenos ili kontrola toka moraju se same pobrinuti za to. UDP je definisan u RFC 768 [41]. Koliko je protokol jednostavan govori činjenica da njegov RFC ima samo tri stranice. Protokol je ažuriran drugim RFC-ovima, ali njegova suština nije mijenjana.

Postoje aplikacije kojima ovakav način rada sasvim odgovara. Postoje i one koje ne bi mogle raditi po UDP. Pogodnost UDP ili TCP za neke namjene razmotrena je nakon objašnjenja TCP protokola.

2.3.2 TCP

TCP je pouzdani transportni protokol koji aplikacijama omogućava dostavu niza bajta po redu. TCP omogućava kontrolu toka. Inicijalna verzija TCP nije omogućavala kontrolu zagušenje, ali se kasnije pokazala neophodnom i dodata je u TCP.

Osnovni mehanizam osiguravanja pouzdanosti prenosa su potvrde. TCP primalac potvrđuje prijem segmenata. Ako TCP pošiljalac ne dobije potvrdu, do isteka vremena čekanja (*timeout*), on segment šalje ponovo. Ovo se ponavlja dok se ne dobije potvrda (u praksi se ponavlja ograničen broj puta koji zavisi od konkretne izvedbe TCP protokola, odnosno operativnog sistema). Vrijeme čekanja na potvrdu nije fiksno. Računa se na osnovu prosječnog trajanja i

varijacije vremena potrebnog da stigne potvrda (RTT), koji se stalno ažuriraju sa tekućim vrijednostima.

Radi bolje efikasnosti, TCP ne šalje segmente jedan po jedan, odnosno ne čeka na potvrdu o prijemu jednog segmenta prije slanja narednog. TCP pošiljalac šalje više segmenata jedan za drugim, bez čekanja na potvrde (*pipeline*). TCP primalac potvrđuje prijem segmenata. Da bi se znalo koji segment se potvrđuje, pošto ne moraju stići istim redoslijedom kojim su i poslani, a i da bi primalac mogao poredati, i aplikativnom sloju isporučiti, segmente oni su numerisani u rastućem redoslijedu. Pošto segmenti ne moraju biti iste veličine, razlika između rednih brojeva dva uzastopna segmenta jednaka je veličini prvog, od ta dva, segmenta. Na primjer, ako je redni broj nekog segmenta 10.000, a veličina 1.000 bajta, redni broj narednog segmenta biće 11.000 (10.000 + 1.000). TCP primalac na isti način numeriše potvrde. Redni broj potvrde za paket sa početnim brojem 10.000 i veličine 1.000 bajta biće 11.000. Redni brojevi segmenata su redni brojevi bajta koji se šalju, a redni brojevi potvrda su redni brojevi narednih bajta koje primalac očekuje. TCP potvrde su kumulativne. To znači da će primalac uvijek poslati redni broj narednog bajta koji očekuje. Na osnovu toga primalac može znati koje segmente je primalac dobio, a koje nije.

Primjerima sa brojevima mogu se pokazati različite situacije. Ako primalac pošalje tri paketa veličine 1.000 bajta svaki, sa početnim rednim brojevima 10.000, 11.000 i 12.000, primalac će po urednom prijemu svakog od paketa poslati potvrde sa rednim brojevima 11.000, 12.000 i 13.000. Ako pošiljalac ne dobije sve potvrde, ali dobije potvrdu sa rednim brojem 13.000, znaće da je primalac primio sva tri segmenta jer inače ne bi poslao potvrdu koja označava da očekuje naredni segment (13.000).

Ako drugi segment iz prethodnog primjera ne stigne do primaoca, on će prilikom pristizanja trećeg segmenta ponovo poslati potvrdu sa rednim brojem 11.000, čime označava da je dobio neki segment, ali da i dalje očekuje drugi (onaj sa rednim brojem 11.000). Kad istekne vrijeme čekanja na potvrdu za drugi segment, on će biti ponovo poslan. Ako uredno stigne do primaoca, i ako je primalac sačuvao treći segment (što ne mora, ali uglavnom radi), primalac će poslati potvrdu sa rednim brojem 13.000 potvrđujući da je sada dobio i drugi i treći segment. Ako je primalac čuvao treći segment, nije ga isporučivao aplikativnom sloju, već je čekao na drugi segment, da bi njega isporučio prije trećeg. Na ovaj način TCP osigurava pouzdanu isporuku bajta po redu kojim su poslani.

Nameće se pitanje koliko segmenata jedan za drugim, bez čekanja na potvrdu može poslati TCP pošiljalac. Ta veličina ograničena je veličinom prostora na primaocu odvojenog za pohranjivanje TCP paketa prije nego što ih aplikativni sloj preuzme. Ovdje je bitno napomenuti da TCP pošiljalac šalje segmente kad mu aplikativni sloj dostavi poruke koje treba poslati, te da TCP primalac zapravo ne prosljeđuje segmente direktno aplikativnom sloju već ih smješta u privremeno spremište (*buffer*) iz kog ih aplikativni sloj preuzima. Ovo preuzimanje, i oslobađanje prostora, nije pod kontrolom transportnog

sloja. Iz tog razloga TCP ima mehanizam koji sprečava da se ovaj prostor prepuni, čime bi se izgubili segmenti koji su uredno poslani i primljeni. To je kontrola toka. Ona se ostvaruje tako da TCP primalac sa svakom potvrdom šalje veličinu, takozvanog, prozora primaoca, dostupnog prostor u prijemnom *buffer*-u. Pošiljalac koristi ovu veličinu da ograniči koliko bajta, odnosno segmenata, može u nekom trenutku poslati bez čekanja na potvrdu. Pri računanju te vrijednosti od veličine prozora primaoca oduzima broj bajta koje je poslao, ali za koje još nije dobio potvrdu.

TCP omogućava istovremenu dvosmjernu komunikaciju (*full duplex*). Obje strane u TCP komunikaciji, i klijent i server, mogu biti i pošiljalac i primalac. Kada web preglednik preko TCP šalje HTTP zahtjev web serveru, taj zahtjev, kao sadržaj segmenta, šalje TCP klijent TCP serveru. U tom trenutku TCP klijent je pošiljalac, a TCP server primalac. Kada web server odgovori na ovaj zahtjev, TCP server će poslati TCP klijentu ovaj odgovor kao sadržaj jednog ili više segmenata. U tom slučaju je TCP server pošiljalac, a TCP klijent primalac. Pored ovoga, segmenti koji prenose podatke, koriste se za prenos potvrda. Kada jedna TCP strana dobije podatke od druge, ako ona ima podatke za tu drugu stranu ona će ih poslati, a uz njih, u odgovarajućem polju zaglavlja, i broj potvrde kojom se potvrđuje prijem podataka. Ovo je efikasnije nego da se potvrde šalju kao odvojeni segmenti. Naravno ako nema podataka za slanje nazad, može se poslati i segment sa potvrdom, bez podataka.

Još jedan dodatak TCP koji pruža podršku pouzdanom prenosu podataka je uspostavljanje konekcije. Za razliku od UDP, koji samo šalje podatkovne segmente, čim ih dobije od aplikativnog sloja, TCP ne šalje segmente sa podacima dok ne uspostavi konekciju sa drugom stranom. Uspostavljanje konekcije inicira TCP klijent. Osnovna namjena je da se potvrdi da je TCP server dostupan i spreman da prihvati segmente adresirane na odredišni port. To znači da na odredišnom čvoru postoji aplikacija koja se registrovala kod operativnog sistema za prijem segmenata koji stižu na taj port. Bez ovoga nema smisla slati segmente. Web preglednik može uzaludno slati HTTP zahtjeve čvoru na kom nema pokrenutog, i kod OS registrovanog, web servera.

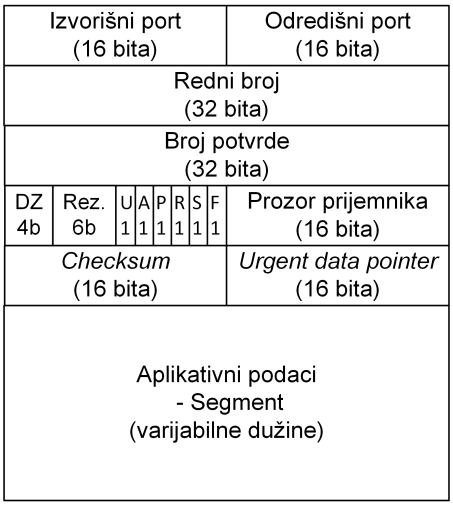
Tokom uspostavljanja konekcije razmjenjuju se i još dvije informacije. Svaka od strana, jer svaka može biti pošiljalac i primalac, šalje svoj početni redni broj i svoju početnu veličinu prozora primaoca. Redni broj prvog segmenta koji TCP šalje nije nula ili jedan, već slučajan broj. Razlog za ovo je osiguravanje da se neki segmenta sa podacima ili potvrda iz prethodne konekcije greškom ne protumače da pripadaju tekućoj konekciji. Pošto su ovi brojevi slučajni primalac mora znati redni broj prvog segmenta koji treba da očekuje. To se rješava ovom razmjenom tokom uspostavljanja konekcije.

Treća informacija koje se razmjenjuje prilikom uspostave konekcije su veličine prozora primalaca. One, kako je objašnjeno, omogućavaju kontrolu toka.

Proces uspostavljanja konekcije sastoji se od tri koraka (*three way handshake*). U prvom TCP klijent šalje TCP serveru segment bez sadržaja u kom je zastavica SYN (odgovarajući bit u zaglavlju) postavljena (bit vrijednost je

jedan). Vrijednost polja redni broj u ovom segmentu predstavlja početni redni broj. U sklopu ovog segmenta obično se šalje i opcionalno polje zaglavlja koje definiše MSS koju TCP klijent može prihvatiti. TCP server, ako je dobio zahtjev i ako je dostupna aplikacija koja očekuje segmente na portu iz zahtjeva, odgovara. U odgovoru postavlja dvije zastavice, SYN i ACK, dostavlja svoj početni reni broj i MSS ka sebi, na isti način kao i klijent. U ovom segmentu server dostavlja i trenutnu veličinu svog prozora. Nakon toga klijent potvrđuje prijem serverskih informacija slanjem segmenta sa postavljenom ACK zastavicom i trenutnom veličinom svog prozora. Od ovog trenutka obje strane imaju sve informacije potrebne da mogu razmjenjivati podatke.

Format TCP segmenta prikazan je na slici 2.4.



Slika 2.4: Format TCP segmenta

U odnosu na UDP, zaglavlje je prošireno sa rednim brojevima segmenata i potvrda, zastavicama i opcijama. Umjesto dužine segmenta u zaglavlju se nalazi dužina zaglavlja (DZ), jer, zbog opcija, zaglavlje nije fiksne dužine. Biti A i S, odgovaraju ACK i SYN zastavicama. Biti R i F, odgovaraju RST i FIN zastavicama koje se koriste za resetovanje i okončavanje TCP konekcije. Biti U i P, odgovaraju zastavicama URG i PSH, koje se, kao i polje *urgent data pointer*, uglavnom ne koriste. Više detalja o TCP može se pronaći u njegovom RFC 793 [43] i drugim RFC koji su ga ažurirali.

Pored kontrole toka, TCP ima, naknadno, ugrađene mehanizme za kontrolu zagušenja. Pokazalo se da kada postoji veći broj krajnjih čvorova koji šalju pakete preko manjeg broja unutrašnjih mrežnih čvorova, što je uglavnom slučaj, unutrašnji čvorovi mogu dobivati više paketa u jedinici vremena nego što mogu poslati. Ova situacija je posljedica ograničene propusnosti izlaznih

veza mrežnih čvorova. Za ovakve situacije mrežni čvorovi imaju prostor za privremeno pohranjivanje (*buffer*) paketa koji čekaju na slanje. Kada se ovaj prostor napuni novi paketi koji dolaze se odbacuju. To je situacija zagušenja mrežnog uređaja. U opštem slučaju, kada krajnji čvorovi putem mreže šalju više paketa nego što mreža može propustiti dolazi do mrežnog zagušenja. Najblaža posljedica zagušenja je sporija isporuka segmenata. Mnogo teža je neisporuka koja izaziva njihovo ponovno slanje od strane TCP pošiljalaca, nakon isteka vremena čekanja, čime se samo povećava opterećenje na mrežu. Da bi se izbjegle ovakve situacije u TCP je uvedena kontrola zagušenja.

TCP kontrola zagušenja se, slično kao i kontrola toka, izvodi definisanjem prozora zagušenja na pošiljaocu. Kao i prozor primaoca, taj prozor ograničava maksimalan broj bajta koje TCP pošiljalac može poslati bez čekanja na potvrdu. TCP pošiljalac je ograničen manjim od prozora primaoca i prozora zagušenja. Za razliku od prozora primaoca, veličinu prozora zagušenja ne može poslati primalac ili neki od mrežnih čvorova na putu. Ovaj prozor TCP pošiljalac računa sam. Logika za računanje veličine prozora zagušenja je da se prozor relativno oprezno (linearno) povećava, dok ne dođe do zagušenja, a onda drastično (multiplikativno) smanjuje. Ovaj pristup se naziva AIMD (*additive increase/multiplicative decrease*). Konkretna izvedba je da na početku konekcije TCP pošiljalac ne šalje odmah onoliko bajta koliko je veličina prozora primaoca. Umjesto toga ograničen je prozorom zagušenja koji se postavlja na broj bajta koji je umnožak MSS. Ovaj umnožak je u prvim verzijama bio jedan, a po novijim preporukama 10. Sa svakom potvrdom koju dobije povećava prozor zagušenja za jedan MSS. Ova faza rada mehanizma kontrole zagušenja naziva se spori početak (*slow start*), jer se šalje sporije u odnosu na kontrolu toka. Efektivno, ovaj početka nije tako spor je prozor zagušenja raste eksponencijalno u vremenu.

Na primjeru rasta prozora zagušenja može se vidjeti ovaj eksponencijalni rast. Ako je u prvom trenutku prozor zagušenja jedan MSS, šalje se jedan segment. Nakon RTT stiže potvrda, prozor se povećava na dva MSS i šalju se dva segmenta. Nakon narednog RTT stižu dvije potvrde, za svaku se prozor zagušenja povečava za jedan MSS, pa je veličina prozora zagušenja četiri MSS. Sad se šalju četiri segmenta. Nakon trećeg RTT stižu četiri potvrde i prozor zagušenja raste na osam MSS. Prozor se uduplava sa svakim RTT, što je eksponencijalni rast.

Očigledno pitanje je dokle prozor zagušenja raste na ovaj način. Ideja je da raste do dostupne propusnosti veze između pošiljaoca i primaoca. Ova propusnost se može računati kao veličina prozora podijeljena sa RTT. Kako pošiljalac ne zna ovu vrijednost (inače bi je odmah koristio), a i ta vrijednost nije konstantna već se mijenja u zavisnosti od stanja na mrežnim uređajima na putu, potrebno je na neki način procijeniti. Ideja je da se, uz uglavnom pouzdane veze, gubitak segmenta javlja kao posljedica zagušenja. Iz ovog razloga se isticanje vremena čekanja na potvrdu smatra indikatorom da se dosegla, i premašila, dostupna propusnost, te da treba smanjiti brzinu slanja. Pošto je segment odbačen, neki od rutera na putu ima pun *buffer* i treba mu dati

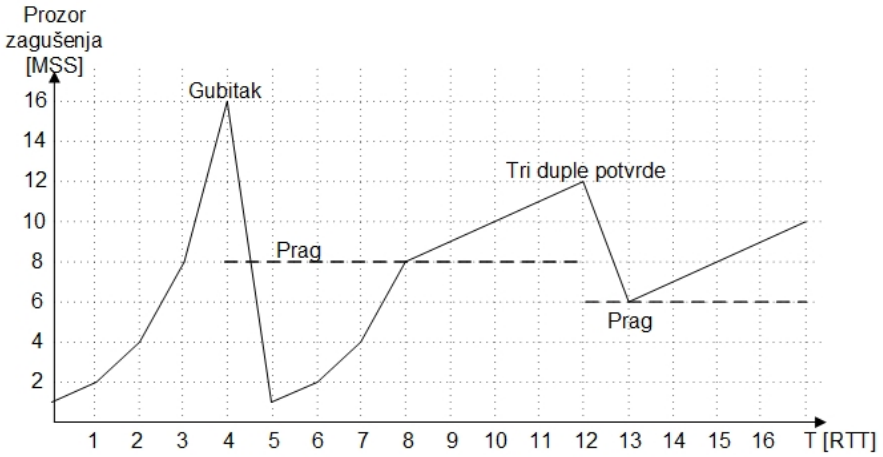
priliku da se isprazni. Iznos trenutne veličina prozora podijeljen sa dva TCP pošiljalac zapamti u lokalnu varijablu koja se naziva prag zagušenja. Veličina prozora zagušenja ponovo se vraća na početnu vrijednost i kreće faza sporog početka. Promjena je što sada eksponencijalan rast traje dok prozor zagušenja ne dostigne veličinu praga zagušenja. Nakon toga prozor zagušenja raste linearno u vremenu, odnosno po jedan MSS sa svakim RTT. Ova faza se naziva izbjegavanje zagušenja (*congestion avoidance*). Nakon novog gubitka paketa proces se ponavlja.

Postoji i drugi metod otkrivanja gubitka paketa koji je brži od čekanja na isticanje vremena čekanja na potvrdu. Taj način otkrivanja gubitaka koristi se i u kontroli zagušenja kao indikator zagušenja. Ako TCP pošiljalac dobije tri puta potvrdu za segment koji je već potvrđen, može se zaključiti da segment nakon onog koji je višestruko potvrđen nije stigao do primatelja, ali jesu tri paketa nakon njega. Ta tri paketa izazvala su tri potvrde čiji redni broj je početni redni broj paketa koji nedostaje. TCP pošiljalac tada ponovo šalje pakete koji nedostaje, bez čekanja na isticanje vremena čekanja na potvrdu, koje bi bilo mnogo duže. Ovo se naziva brzo ponovno slanje (*fast retransmit*).

TCP mehanizam kontrole zagušenja pristizanje tri ponovljene potvrde tumači kao znak da je došlo do zagušenja, ali da je ono manje drastično od onog kad istekne vrijeme čekanja na potvrdu, jer očigledno neki paketi prolaze. Iz tog razloga prozor zagušenja se u tom trenutku prepolovi i od te vrijednosti raste linearno u vremenu (faza izbjegavanja zagušenja).

Ovo je osnovni princip rada nakon koga su razvijene mnoge druge izvedbe. Trenutno nema usklađene verzije TCP kontrole zagušenja koja se univerzalno koristi, ali to ne smeta koegzistenciji.

Promjene TCP prozora zagušenja prikazane su na slici 2.5.



Slika 2.5: TCP prozor zagušenja

Aplikacije prilikom prosljeđivanja poruka transportnom sloju biraju da li žele koristiti TCP ili UDP. Postojanje i upotreba dva transportna protokola ukazuje na to da postoji potreba za oba. Na prvi pogled pouzdani TCP izgleda kao mnogo bolje riješenje od nepouzdanog UDP. I zaista, za aplikacije kojim je pouzdan prenos podataka neophodan za ispravno funkcionisanje TCP je bolji, ili jedini izbor. Nedostatak TCP je što se za pouzdanost troši dodatno vrijeme, propusnost i računarske resurse. Vrijeme se troši na uspostavljanje konekcije. Propusnost se troši na dodatna polja zaglavlja. Računarski resursi, pošiljaoca i primaoca, se troše na obradu segmenata koje TCP traži. Postoje aplikacije kojima ovo nije neophodno. Ranije objašnjeni DNS može raditi i po UDP i po TCP. Uglavnom se koristi UDP jer je dovoljno dobar. DNS klijent šalje numerisani upit i ako ne dobije odgovor šalje upit ponovo. Na ova način je pouzdanost riješena na aplikativnom sloju i može se koristiti nepouzdani transportni protokol, koji je manje zahtjevan. HTTP koristi TCP kao transportni protokol. U principu većina aplikacija koje prenose podatke koriste TCP, jer je njima bitno da svaki bit stigne na odredište neizmijenjen. Multimedijalne aplikacije su primjer onih kojim UDP može biti pogodan. Za ove aplikacije, koje prenose podatke u realnom vremenu, brzina je važnija od pouzdanosti. One ne mogu priuštiti kašnjenje koje izaziva ponovno slanje izgubljenih segmenata kod TCP. Zbog kontinualnosti sadržaja koji se prenosi (zvuk, video) gubitak nekih segmenata moguće je tolerisati, interpolacijom ili sličnim metodama na primaocu.

Današnje stanje je da se TCP koristi gotovo za sve osim DNS. Čak se i multimedijalne aplikacije šalju preko TCP upotrebom metoda poput DASH (*Dynamic Adaptive Streaming over* HTTP) ili HLS (HTTP *Live Streaming*). Jedan od razloga je što UDP nema kontrolu zagušenja i može potrošiti svu dostupnu propusnost. UDP saobraćaj, osim po portu 53 za DNS, je uglavnom zabranjen na većini *firewall*. Ipak stvari se pomalo mijenjaju. Najnovija verzija HTTP/3, obrađena u poglavlju 3.5, koja je u fazi razmatranja, koristi UDP kao transportni protokol i pouzdanost osigurava na aplikativnom sloju.

Nezavisno od protokola transportnog sloja, segment koji se napravi dostavlja se mrežnom sloju od kog se očekuje da ga dostavi do transportnog sloja na odredišnom čvoru. Transportni sloj ne brine o tome kojim putem će segment doći na odredište. Jedino brine o tome da segment ima sva potrebna zaglavlja, u skladu sa korištenim transportnim protokolom, Mrežni sloj, za transportni, apstrahuje izvedbu mreže i dostupne putanje.

2.4 Mrežni sloj

Mrežni sloj prihvata segmente transportnog sloja i dostavlja ih transportnom sloju na odredišnom računaru. Mrežni sloj enkapsulira segment sa transport-

nog sloja u svoja zaglavlja, bez potreba da vodi računa o sadržaju i formatu tog segmenta. Za mrežni sloj to je niz bajta koje treba dostaviti.

Na segment transportnog sloja se dodaju zaglavlja mrežnog sloja i tako se dobiva mrežni paket, jedinica podataka koja se šalje na mrežnom sloju. Uloga mrežnog sloja je da pronađe putanju kroz niz mrežnih čvorova od izvorišnog do odredišnog čvora i da po toj putanji dostavi paket. Za razliku od transportnog sloja koji se izvršava samo u izvorišnom i odredišnom čvoru, mrežni sloj se izvršava u svim mrežnim čvorovima koji vrše prosljeđivanje paketa na osnovu podataka sa mrežnog sloja. Ti uređaji se nazivaju ruteri (*router*). Ruteri vrše prespajanje (*switching*) ili komutacija, poput telefonskih centrala. Mrežni sloj šalje pakete od izvorišnog čvora do prvog rutera na putanji za koju je, mrežni sloj, utvrdio da je najbolja. Mrežni sloj tog prvog rutera šalje paket do narednog rutera na putanji. Ovaj proces se nastavlja do posljednjeg rutera na putanji koji šalje paket do odredišnog čvora. Mrežni sloj se ne bavi pitanjem kako će paket biti dostavljen od jednog do drugog rutera/čvora. On ga predaje podatkovnom sloju sa adresom narednog čvora/rutera. Podatkovni sloj se bavi ovim pitanje. Ovo je vrlo bitno jer odvaja pitanje pronalaska putanje od vrste veze (žičana, bežična) i protokola (Ethernet, WiFi, ...) koji se koristi na nekoj vezi, kao dijelu cjelokupne putanje.

Slanje paketa na mrežnom sloju ostvaruje se pomoću dvije osnovne funkcije mrežnog sloja: rutiranje i prosljeđivanje. Rutiranje je pronalazak putanje kroz mrežu. Prosljeđivanje je proces prebacivanja paketa sa ulaz na odgovarajući izlaz u skladu sa putanjom koju je odredilo rutiranje. Rezultat rutiranja su pravila prosljeđivanja upisana u rutere na osnovu kojih oni vrše prosljeđivanje. Pravila rutiranje i prosljeđivanje se uglavnom rade na osnovu adresa na mrežnom sloju.

Mrežni sloj treba da riješi pitanje adresiranja. Za globalni sistem kao što je Internet ovo adresiranje mora biti globalno. Za velike sisteme poput Interneta vrlo je bitna skalabilnost protokola. Rast mreže ne bi trebao da negativno utiče na rad mrežnog sloja. IP protokol koji je dominantan mrežni protokol rješava oba ova pitanja. Kako je IP jedini protokol mrežnog sloja koji se ovdje razmatra na njegovom primjeru će biti predstavljena sva pitanja kojim se mrežni sloj bavi.

2.4.1 IP protokol

IP protokol definiše konvencije adresiranja na mrežnom sloju, format paketa odnosno njegova zaglavlja i način rukovanja sa paketom. Postoje dvije verzije IP protokola koje se koriste, IPv4 i IPv6. Kako je ovo poglavlje o tradicionalnim računarskim mrežama, obrađena je tradicionalna IP verzija 4. Veliki dio principa rada koji važe za verziju 4 važe i za verziju 6. U verziji 6 adrese su duže i format zaglavlja je različit. Tokom objašnjavanja IPv4 protokola ukazano je na dijelove koji se razlikuju kod IPv6. IPv4 je definisan u RFC 791 [42], a IPv6 u RFC 8200 [56].

IPv4 (u nastavku samo IP) adrese su dugačke 32 bita. Uobičajeni način zapisivanja je da se svaki bajt IP adrese zapiše kao decimalna vrijednost i da se ove vrijednosti odvoje tačkom. Vrijednost svakog od ovih članova IP adrese (okteta) može biti od 0 do 255. Ukupan mogući broj IP adresa je 2^{32}, što je nešto više od četiri milijarde. Neke od adresa imaju posebne namjene, ali velika većina se koristi za adresiranje mrežnih čvorova.

Ove adrese su globalne i trebale bi biti globalno jedinstvene. Jedinstvenost adresa se, slično kao kod poštanskih adresa, osigurava hijerarhijskom podjelom. Centralno tijelo za raspodjelu IP adresa je ICANN (Internet Corporation for Assigned Names and Numbers). ICANN IP adrese raspoređuje po regionalnim Internet registrima (RIR), kojih je pet i otprilike pokrivaju teritoriju najvećih kontinenata. RIR su RIPE (Evropa), AFRINIC (Afrika), APNIC (Azija), ARIN (Sjeverna Amerika) i LACNIC (Južna Amerika). Regionalni registri, IP adrese koje su dobili od ICANN, dalje raspoređuju po lokalnim Internet registrima (LIR). LIR su uglavnom davaoci usluge pristupa Internetu (ISP). Oni dodjeljuju IP adrese svojim korisnicima. LIR obično dobivaju adrese u blokovima uzastopnih adresa, pri čemu je broj adresa stepen broja dva. Veličina bloka zavisi od njihovih potreba i broja dostupnih adresa u RIR. Većina LIR je tražila i dobivala blokove adresa u više navrata, tako da uglavnom imaju više blokova IP adresa. Raspodjela IP adresa po blokovima je javna. Na toj informaciji se zasniva određivanje fizičke lokacije web korisnika od strane web servera. Iz IP adrese pošiljaoca zahtjeva saznaju ko je njegov ISP (LIR), te na osnovu javne informacije o lokaciji ISP zaključuju da i korisnik mora biti u blizini (odnosno bar u istoj državi).

IP adrese dodjeljuju se čvorovima, preciznije njihovim mrežnim interfejsima, koji su uglavnom izvedeni kao NIC (*network interface card*). Čvor ima onoliko IP adresa koliko ima NIC. IP adresa može biti dodijeljena ručno od strane administratora čvora ili automatski od strane drugog mrežnog čvora koji obavlja funkciju dodjele adresa upotrebom DHCP protokola. Nezavisno od načina dobivanja IP adrese ona mora biti iz skupa adresa mreže kojoj pripada. Ako čvor pripada mreži nekog korisnika ISP koji mu je dodijelio blok od 256 IP adresa na korištenje, adresa čvora mora biti jedna od tih 256, uzastopnih, IP adresa koju ne koristi ni jedan drugi čvor iz te mreže.

Od svih čvorova jedne mreže, ako je ta mreža povezana sa drugim mrežama, bar jedan čvor ima ulogu povezivanja. Taj čvor se naziva ruter (*router*). Ruteri imaju onoliko mrežnih interfejsa koliko mreža povezuju. Iz tog razloga krajnji čvorovi, obično imaju samo jedan mrežni interfejs, jer pripadaju samo jednoj mreži. Svaka od IP adresa mrežnih interfejsa rutera pripada bloku IP adresa mreže kojoj interfejs pripada. Čvorovi unutar mreže mogu imati paket za slanje čija IP adresa može biti proizvoljna. Ono što je za mrežni sloj bitno je da li ta odredišna IP adresa pripada bloku IP adresa te mreže ili ne. Ako pripada, šalje se ka odredišnom čvoru direktno, bez posredstva rutera. Ako ne pripada, paket se mora dostaviti do rutera koji onda odlučuje kuda dalje ga treba proslijediti. Dostavljanje između čvorova iste mreže, koje uključuje

i dostavljanje ruteru, odvija se korištenjem podatkovnog sloja. Ovaj proces opisan je, kasnije, u sklopu opisa podatkovnog sloja.

Da bi čvor znao koje adrese pripadaju njegovoj mreži on uz IP adresu mora imati i podmrežnu masku (*subnet mask*). Podmrežna maska je iste dužine kao i IP adresa, 32 bita. Sastoji se od niza jedinica, nakon kog slijedi niz nula. Broj jedinica u podmrežnoj masci definiše koliko prvih bita IP adrese je isto za sve čvorove u mreži. Da bi odredio da li je odredišni čvor u njegovoj mreži, pošiljalac poredi svoju IP adresu sa odredišnom IP adresom, koristeći podmrežnu masku. Ako su IP adresa čvora i odredišta identične u prvih n bita, gdje je n broj jedinica u podmrežnoj masci, onda je odredišni čvor u istoj mreži. Ako nisu, onda je odredišni čvor u drugoj mreži.

Na primjer, ako je IP adresa čvora 100.100.100.100 i njegova podmrežna maska 255.255.255.0 (24 jedinice i osam nula), onda je IP adresa 100.100.100.50 u njegovoj mreži, a adresa 100.100.50.100 nije. Podmrežna maska 255.255.255.0 označava da mreži nekog čvora pripadaju sve IP adrese kojim su prva tri okteta IP adresa ista kao kod IP adrese tog čvora. U prethodnom slučaju, mreži čvora sa IP adresom 100.100.100.100 pripadaju sve IP adrese od 100.100.100.0 do 100.100.100.255. Da je podmrežna maska 255.255.0.0 (16 jedinica i 16 nula) onda bi i adresa 100.100.50.100 pripadala istoj mreži. U tom slučaju mreži bi pripadale sve adrese od 100.100.0.0 do 100.100.255.255. Adresa gornjeg čvora sa jednom i drugom podmrežnom maskom se često skraćeno piše 100.100.100.100/24, odnosno 100.100.100.100/16. Brojevi 24 i 16 označavaju broj jedinica u podmrežnoj masci. Adrese gornje mreže bi se za ova dva slučaja pisale 100.100.100.0/24, odnosno 100.100.0.0/16. Adrese mreža koriste se u tabelama prosljeđivanja u ruterima. Ruteri i njihov rad su obrađeni kasnije u tekstu.

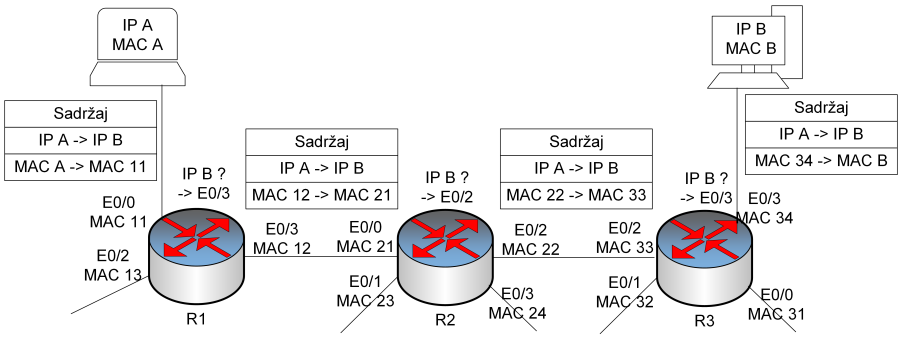
Čvor na osnovu svoje IP adrese i podmrežne maske može odrediti da li je neka odredišna P adresa u njegovoj mreži ili nije. Ako nije onda paket treba dostaviti čvoru u svojoj mreži, ruteru, koji je zadužen za prosljeđivanje paketa za druge mreže. Da bi čvor mogao dostaviti paket ruteru mora znati njegovu adresu. Adresa čvora preko kog se šalju paketi van mreže dio je mrežne konfiguracije čvora, skupa sa IP adresom i podmrežnom maskom. Ovaj čvor se naziva zadani izlaz (*default gateway*).

Da bi bio kompletiran skup mrežnih parametara čvora potrebno je još da ima i IP adresu DNS servera. Tu adresu koristi DNS klijent na čvoru, *resolver*, da kontaktira server imena kome dostavlja upite. Ova četiri mrežna parametra, IP adresa, podmrežna maska, IP adresa *default gateway* i IP adresa DNS servera, se mogu podesiti ručno ili dobiti putem DHCP.

Kada čvor dostavi paket, čija je IP adresa vam njegove mreže, ruteru, IP ruter na osnovu IP adrese odredišta odlučuje šta da radi sa paketom. Ruter ima tabelu prosljeđivanja u kojoj se nalaze blokovi IP adresa do kojih ruter zna prosljeđivati pakete. Za svaki blok adresa u tabeli prosljeđivanja navedeno je na koji interfejs rutera i na koju narednu IP adresu (*next hop*) treba proslijediti paket čija je odredišna IP adresa iz tog bloka. Ova tabela prosljeđivanja rezultat je procesa rutiranja. Ovaj proces je kasnije objašnjen. Ono što je

bitno je da ruter može znati koji je naredni ruter kom treba proslijediti paket. Taj naredni ruter nalazi se u istoj mreži (isti skup IP adresa) kao i interfejs po kom se paket šalje. Bitno je napomenuti da ruter ne mijenja odredišnu IP adresu paketa, već je samo koristi da bi, iz tabele prosljeđivanja, došao do IP adrese slijedećeg čvora kom treba dostaviti paket na dalje prosljeđivanje. Za dostavljanje paketa do narednog rutera koristi se usluga podatkovnog sloja. Mrežni paket, koji se šalje od rutera do rutera, se ne mijenja, nego se samo mijenja zaglavlje podatkovnog sloja na osnovu kog se dostavlja sljedećem ruteru. Paket na ovaj način putuje od rutera do rutera, po putanji koju su izračunali protokoli rutiranja. Na kraju putanje dolazi do rutera čiji jedan interfejs pripada istoj mreži kao i odredišni čvor (njegova IP adresa). Taj krajnji ruter dostavlja paket odredišnom čvoru koristeći podatkovni sloj mreže u kojoj se oni nalaze. Na ovaj način mrežni sloj obavlja svoju uslugu dostavljanja segmenta transportnog sloja od početnog do krajnjeg mrežnog čvora.

Putanja IP paketa od početnog do konačnog mrežnog čvora prikazana je na slici 2.6.



Slika 2.6: Put IP paketa

Potrebno je još objasniti format IP zaglavlja koji podržava ovaj rad, kao i rutiranje koje formira tabele prosljeđivanja.

IP zaglavlje, kao i zaglavlja na svim slojevima, ima adrese, odredišnu i izvorišnu, U IP zaglavlju to su IP adrese. Za prosljeđivanje paketa se koristi samo odredišna IP adresa. Izvorišna IP adresa služi prijemnom čvoru da zna od koga je dobio paket i da na osnovu adrese može poslati odgovor. Ova polja postoje i u IPv6, ali su dužine ovih adresa 128 bita.

Iako je rečeno da se IP paket ne mijenja na svom putu postoji jedno, tačnije dva polja koja se mijenjaju na svakom ruteru. Zbog moguće greške u konfiguraciji i pogrešno izračunatih putanja paket bi mogao doći u situaciju da kruži kroz mrežu i nikad ne stigne na odredište. Da bi se to spriječilo u zaglavlju IP paketa postoji polje koje služio kao svojevrstan odbrojavač. Vrijednost tog polja svaki ruter, kroz koji paket prođe, smanjuje za jedan.

Ako vrijednost padne na nulu paket se odbacuje. Početna vrijednost ovog polja zavisi od OS i za Linux 2.4 kernel iznosi 255, a za Windows 10 je 128. Ovo polje IP zaglavlja se naziva TTL (*time to live*). Ovo polje postoji i u IPv6, ali se naziva *hop limit*.

Slično kao i kod zaglavlja transportnog sloja i IP zaglavlje ima polje koje služi za detekciju grešaka. Kod IP ovo polje otkriva greške samo u zaglavlju IP paketa, pa se naziva *checksum* zaglavlja. Ovo je drugo polje koje se mijenja na svakom ruteru, kao posljedica promjene TTL mijenja se i vrijednost *checksum*. IP ne ispravlja greške, nego odbacuje pakete sa neispravnim *checksum*. Iz toga se može zaključiti da IP ne pruža pouzdan prenos paketa i nema mehanizama za to. Za pouzdan prenos, ako je potreban, mora se pobrinuti transportni sloj, na način kako to TCP radi. Ovo polje je izbačeno iz IPv6 zaglavlja.

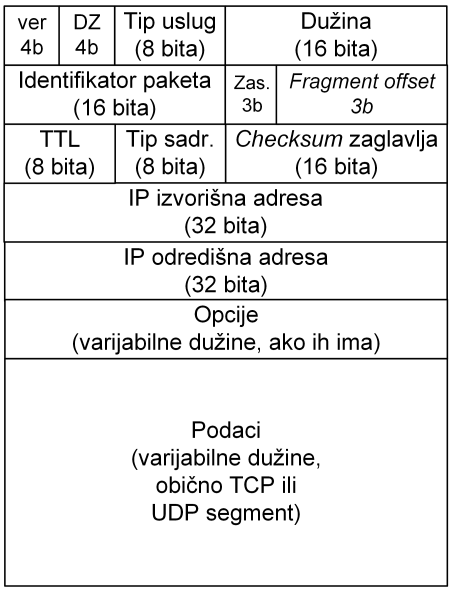
I IP zaglavlje nosi informaciju o dužini, i to u dva polja. Jedno je za ukupnu dužinu paketa, a drugo za dužinu zaglavlja, jer i IP zaglavlje ima opcionalna polja, koja se rijetko koriste. U IPv6 postoji samo jedno polje za dužinu sadržaja paketa.

Postoji posebna grupa polja u zaglavlju IP paketa vezana za fragmentaciju. Ranije je navedeno da TCP ograničava veličinu segmenta na MSS. Vrijednost MSS se računa prilikom uspostavljanja konekcije na osnovu MSS vrijednosti koju dostavi primalac. Primalac ovu vrijednost računa na osnovu MTU svog interfejsa po kom je uspostavljena konekcija. Moguće je da negdje na putu paketa postoji MTU koji je manji i koji ne može prihvatiti pakete čija veličina je zasnovana na MSS uspostavljenom na početku. IP rješava ovaj problem na licu mjesta, odnosno na ruteru koji je dobio paket koji je veći od onoga koji je moguće poslati po izlaznoj vezi. Taj ruter podijeli taj paket na potreban broj dijelova. Paketi se ponovo sastavljaju u jedan na odredišnom čvoru. Da bi odredišni čvor mogao sastaviti paket iz dijelova, ruter koji ga je rastavio upisuje nove vrijednosti u tri polja u zaglavlje. Jedno polje je identifikator paketa koje ima istu vrijednost za sve dijelove istog paketa. Drugo je oznaka (jedan bit) da li se radi o posljednjem dijelu paketu ili ne. Treće je udaljenost dijela paketa od početka originalnog paketa (*offset*). Potrebno je napomenuti da svi paketi imaju ova polja. Kod onih koji nisu fragmentirani označeno je da su posljednji, a udaljenost od početka im je nula. Ovih polja nema u IPv6. IPv6 ne podržava fragmentaciju. IPv6 pošiljalac mora poslati paket veličine koju mogu prihvatiti sve veza na putanji.

Pošto postoji više verzija IP, prvo polje IP zaglavlja je verzija IP paketa. Polje zaglavlja koje se u tradicionalnim mrežama uglavnom nije koristilo je "tip usluge". Polje bi trebalo dati informaciju ruteru na osnovu koje može raditi prioritetizaciju paketa. Posljednje preostalo neobjašnjeno polje je ono koje ukazuje šta je sadržaj IP paketa. Polje govori kome mrežni sloj treba isporučiti paket. Da li je to TCP ili UDP ili nešto drugo. Sva tri polja sa nešto drugačijim nazivima postoje i u IPv6.

Format IP paketa prikazan je na slici 2.7.

U skupu IP adresa postoji poseban podskup takozvanih privatnih IP adresa. Ove adrese su namijenjene za internu upotrebu unutar zatvorenih



Slika 2.7: Format IP paketa

sistema. Privatne IP adrese nisu globalno jedinstvene. Neku privatnu IP adresu može imati više različitih čvorova, u različitim mrežama na različitim, međusobno nepovezanim lokacijama. Iz ovoga je očigledno da čvorovi na javnom Internetu ne mogu imati privatne IP adrese. Blokovi privatnih IP adresa su:

- 10.0.0.0 - 10.255.255.255 (10/8)
- 172.16.0.0 - 172.31.255.255 (172.16/12)
- 192.168.0.0 - 192.168.255.255 (192.168/16)

Privatne IP adrese detaljnije su opisane u RFC 1918 [65].

Iako je broj od preko četiri milijarde IP adresa izgledao kao dovoljan, vrlo brzo se ispostavilo da nije. Inicijalno su LIR, poput univerziteta, dobivali velike blokove (/8) IP adresa. Kada je Internet postao popularan, povećao se broj čvorova koji su se povezivali na Internet i ponestalo je javnih IP adresa. Privatne IP adrese su omogućavale upotrebu IP adresa u zatvorenim sredinama, ali čvorovi sa tim adresama nisu mogli direktno pristupati javnom Internetu.

Rješenje koje je pomenuto i u RFC 1918, je postojanje posrednika (*proxy*). Posrednik omogućava čvorovima sa privatnim IP adresama da šalju pakete čvorovima sa javim IP adresama i da od njih dobivaju odgovore. Posrednik koji omogućava ovo prevođenje adresa naziva se, prigodno, NAT (*Network Address Translator*). NAT je funkcionalnost koja je izvedena na mrežnom uređaju koji ima bar dva mrežna interfejsa. Najčešće se ova funkcionalnost dodaje na rutere. NAT prihvata pakete koji dolaze po njegovom, unutrašnjem

interfejsu i prosljeđuje ih dalje sa IP adresom svog vanjskog interfejsa. Ako se u mreži vezanoj na unutrašnji interfejs nalaze privatne IP adrese, a na vanjskom je javna IP adresa, NAT omogućava da se sa privatnih IP adresa šalju paketi na javni Internet, jer im je izvorišna IP adresa javna. Da bi odgovor, koji će stići na vanjsku, javnu, IP adresu NAT uređaja, mogao biti proslijeđen do izvornog unutrašnjeg čvora, potrebno je da NAT zapamti preslikavanje IP adresa koje je uradio. Pošto se više unutrašnjih IP adresa preslikava na jednu vanjsku, potrebna je dodatne informacija po kojoj će se ova preslikavanja razlikovati. NAT ne mijenja samo izvorišnu IP adresu paketa, već i izvorišni port. U svojoj tabeli NAT preslikavanja zapiše originalnu izvorišnu IP adresu i port paketa i novi izvorišni port. Odgovor na paket imaće kao odredišni port onaj koji je NAT postavio kao izvorišni. Na osnovu ovog porta će se izvršiti uvid u NAT tabelu i na paketu odredišna IP adresa i port zamijeniti sa originalnim. Na taj način odgovor može stići čvoru koji je poslao originalni paket. NAT je objašnjen u RFC 3022 [40].

NAT je omogućio većem broju čvorova da imaju IP adrese i komuniciraju sa Internetom nego što bi to bilo moguće da svi moraju imati javne IP adrese. NAT to radi narušavajući odvojenost mrežnih slojeva, jer radi i na mrežnom i transportnom sloju. NAT takođe mijenja paket, premda mrežni uređaji to ne bi trebali raditi. Korist koju NAT donosi bila je dovoljno velika da bi se ovi njegovi nedostaci mogli prihvatiti. Danas je NAT sveprisutan i Internet ne bi porastao do ove veličine bez NAT-a. NAT donosi još neke prednosti kao što je nezavisnost unutrašnjega adresiranja od javne IP adrese koju dodjeljuje ISP. Promjena ISP ne utiče na unutrašnje adresiranje. Takođe, NAT sakriva unutrašnje čvorove od javnog Interneta jer oni nisu direktno dostupni po svojim privatnim IP adresama. Ipak, skup javnih IPv4 adresa dostupnih za podjelu, uz sve optimizacije, je u Evropi već iscrpljen krajem 2019. godine [38]. IPv6, sa mnogo većim skupom adresa, čini NAT manje potrebnim, ali ga ne ukida.

Ruteri prosljeđuju pakete na osnovu unosa u tabelama prosljeđivanja koje su nastale kao rezultat procesa rutiranja. Proces rutiranja omogućava ruterima da u saradnji sa drugim ruterima pronađu najbolje putanje od sebe do svih drugih mreže sa kojim su ti ruteri povezani. Najbolja putanja je ona koja ima najnižu cijenu. Cijena se računa kao skup cijena svih pojedinih veza do odredišne mreže. Cijena veze se, uglavnom, administrativno postavlja na ruteru za svaki od njegovih interfejsa sa tom vezom. Cijena može zavisiti od različitih faktora poput propusnosti, kašnjenja ili nečeg drugog. Rutiranje se ne bavi računanjem cijena pojedinih veza već ove cijene koristi za pronalazak ukupne putanja sa najnižom cijenom. Rezultat rutiranja, tabela prosljeđivanja, ne sadrži cijelu putanju već samo naredni ruter na najboljoj putanji ka odredišnoj mreži.

Računanje najniže cijene je poznat problem iz teorije grafova. Ruteri su čvorovi, a veze su ivice grafa. Obično se koristi jedan od dva algoritma za

pronalazak putanje. Jedan se naziva vektor udaljenosti (*distance vector*), a drugi stanje veze (*link state*).

Algoritam vektora udaljenosti radi tako što svaki čvor od čvorova sa kojim je povezan dobije tabele sa njihovim cijenama do svih mreža. Čvor koji je dobio ove informacije, za sve mreže izračuna ukupnu cijenu koja se sastoji od zbira njegove cijene do susjednog čvora i cijene koju je taj čvor poslao do svake od mreža. Za svaku mreža čvor izabere putanju preko susjednog čvora sa ukupno najnižom cijenom i zapiše informaciju da do te mreže pakete šalje prema tom susjedu. Ovaj algoritam se naziva Bellman-Ford. Kod ovog algoritma računanje i izbor najbolje putanje su jednostavni, ali za veliki broj rutera tabele sa cijenama koje šalju svojim komšijama mogu biti velike. Potrebno je primijetiti da u početnom trenutku, ako su svi ruteri upaljeni odjednom, ruteri imaju samo informacije o svojim cijenama do susjednih rutera, postavljene od strane administratora, koje im šalju. Kad ruter dobije nove informacije, on ažurira svoju tabelu cijena i ako se promijenila šalje je svim svojim komšijama. Nakon određenog broja razmjena, koji zavisi od veličine mreže, više neće biti promjena i tabele prosljeđivanja će biti formirane. Protokol rutiranja koji koristi ovaj algoritam je RIP.

Algoritam stanja veze šalje samo cijene svojih veza do susjednih čvorova, postavljene od strane administratora, ali ih šalje svim čvorovima. Kad čvor dobije cijene veza nekog drugog čvora on ih prosljeđuje do svojih komšija. Na ovaj način svi čvorovi dobiju informacije o svim vezama svih čvorova. Na osnovu ovih informacija svaki od čvorova izvršava Dijkstra algoritam za traženje najniže cijene od sebe do svih ostalih čvorova u mreži. Ovo računanje je komplikovanije od onog koje radi Bellman-Ford, ali je, obično, količina informacija koja se šalje po mreži (samo cijene direktnih veza) manja. Od početnog trenutka potrebno je da proteče neko vrijeme dok sve cijene veza stignu dio svih čvorova, ali je to kraće od vremena potrebnom da algoritam vektora distanci dođe do konačnih cijena. Protokol rutiranja koji koristi algoritam stanja veza je OSPF.

Oba algoritma će naći iste putanje sa najnižom cijenom. Zapravo, ako postoji više putanja sa istom najnižom cijenom moguće je da algoritmi pronađu različite od ovih putanja. U svakom slučaju neće jedan algoritam pronaći bolju putanju od drugog.

Bitno je naglasiti da se računanje putanja odvija posebno u svakom od rutera. Od drugih rutera samo dobivaju informacije potrebne za ovo računanje. Ta činjenica je pogodna za skaliranje. Dodavanje novih rutera povećava broj mreža i veličinu tabela rutiranja, ali ne tako mnogo težinu računanja. Novi ruteri preuzimaju računanje svojih putanja. Tabele prosljeđivanja i dalje sadrže samo susjedne rutere kao naredna odredišta paketa.

Navedeno skaliranje ipak ima ograničenje. Na javnom Internetu koji ima stotine miliona čvorova postoje milioni mreža. Pohranjivanje unosa za sve ove mreža u tablama prosljeđivanja u svim ruterima nije pogodno. Takve tabele bi zauzimale mnogo prostora, njihovo pretraživanje bi bilo sporo i zahtjevno za CPU. Ruteri koji bi ovo podržavali bili bi skuplji i sporiji. Iz ovog razloga

javni Internet nije jedna oblast u kojoj se nalaze sve mreže i ruteri već je podijeljen.

Osnovna jedinica podjele je autonomni sistem (AS). AS je skup mreža, i rutera, pod kontrolom jedne organizacije. Svi ruteri unutar AS koriste isti protokol rutiranja sa istim značenjem cijena veza. Sistemi su autonomni jer svaki može koristiti protokol rutiranja i značenje cijena veza po svom izboru, nezavisno od toga šta su drugi sistemi izabrali. Protokoli rutiranja koji se koriste unutar AS su pomenuti RIP, OSPF, kao i drugi poput IS-IS. Ovi protokoli se nazivaju internim ili inter-AS.

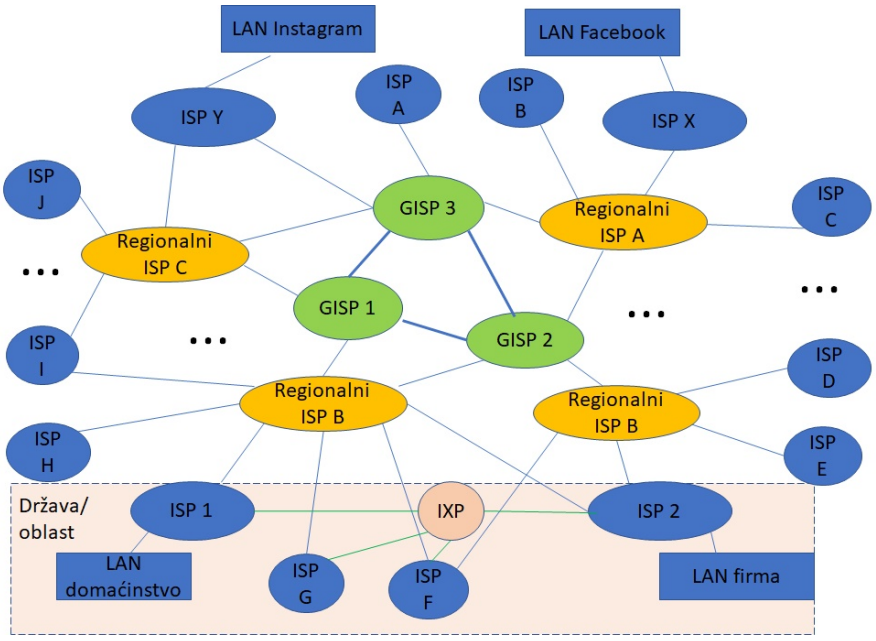
Kada čvor iz jednog AS treba komunicirati sa čvorom u drugom AS potrebno je pronaći putanju koja vodi kroz više AS koji povezuju AS izvornog i AS odredišnog čvora. Pronalazak te putanje radi se pomoću eksternih ili intra-AS ruting protokola. Ovi protokoli pronalaze niz AS koji vodi od izvorišnog AS do odredišnog AS. Može se reći da ovi protokoli apstrahuju AS, koji se mogu sastojati od velikog broja rutera i mreža, i posmatraju ih slično kako interni posmatraju rutere. Ovi protokoli ne mogu pronaći putanju sa najnižom cijenom, jer cijena putanje zavisi od cijena prolaska paketa kroz svaki AS na putu. Kako se cijene putanja unutar različitih AS računaju na različite načine njihovo sabiranje ili poređenje nema smisla. U praksi se koristi jedan eksterni protokol rutiranja, koji je *de facto* standard. Taj protokol je BGP.

BGP omogućava rubnim BGP ruterima AS da sa rubnim BGP ruterima susjednih AS razmjene informacije do kojih mreža znaju proslijediti pakete i preko kojih AS. Na osnovu ove razmjene BGP ruteri pronalaze putanje do svih javno dostupnih mreža na Internetu. Ako postoji više putanja, u principu, BGP bira onu sa manje AS, osim ako postoji administrativno ograničenje (politika) koja čini tu putanju neprihvatljivom. Ako postoji više putanja sa istim brojem AS bira se ona kod koje je rubni ruter susjednog BGP preko kog kreće putanja, bliži po internom protokolu rutiranja. Slično kao i kod internih algoritama rutiranja rezultat rada BGP su adrese sljedećih rutera kojim treba proslijediti paket, a ne cijela putanja. Čak i uz ovako skaliran sistem uz agregaciju IP adresa koja omogućava da je potrebno rutirati samo do LIR-ova broj unosa u BGP tabelama prosljeđivanja je početkom 2020. bio veći od 800.000 [10]. Broj AS se približavao 100.000 [34].

Na kraju je još preostalo da se kratko opiše kako zapravo Internet izgleda, odnosno kako su međusobno povezane mreže koje ga čine. Krajnji čvorovi, bilo da se radi o korisnicima koji konzumiraju Internet usluge, ili o serverima koji ih pružaju, vezu sa ostatkom Interneta ostvaruju putem svojih davalaca usluge pristupa Internetu (ISP). Veći korisnici mogu imati i više od jednog ISP, radi redundantnosti. Ovi ISP se obično nazivaju pristupnim ISP. Pristupni ISP uglavnom nisu međusobno direktno povezani, osim ako nisu u geografskoj blizini. Povezivanje pristupnih ISP uglavnom se obavlja preko većih ISP čiji korisnici su pristupni ISP. Jednu grupu čine regionalni ISP. Ovi regionalni ISP povezuju pristupne ISP u nekom geografskom regionu. Povezivanje regionalnih ISP se obavlja preko globalnih ISP. Globalni ISP pokrivaju globalno tržište i pružaju usluge povezivanja regionalnim ISP. Globalni ISP se nazi-

vaju ISP prvog nivoa (*tier-1*). Postoji desetak ovih ISP i oni su međusobno povezani. Opisani način povezivanja nije jedini niti obavezan. Pristupni ISP mogu biti povezani direktno sa globalnim ISP. Skup ISP na nekom području može biti povezan preko IXP (Internet eXchange point). Pored ISP na Internetu se pojavljuju i ponuđači sadržaja poput Google. Ovi ponuđači sadržaja žele da njihovi korisnici imaju što bolji pristup i zbog toga prave svoju mrežu koja omogućava da se pristupni ISP direktno povezuju sa njima. Tako da je globalni Internet isprepletena mreža mreža, koja dobro funkioniše, dobrim dijelom zahvaljujući IP protokolu.

Na slici 2.8 je prikazana pojednostavljena šema glavnih elemenata u povezivanju skupa mreža koje čine Internet.



Slika 2.8: Organizacija Interneta

Kao uvod u opis rada podatkovnog sloja kratko ponavljanje veze mrežnog i podatkovnog sloja. Paket mrežnog sloja dostavlja se podatkovnom sloju od kog se očekuje da ga dostavi do slijedećeg mrežnog čvora na putu do konačnog odredišta. Mrežni sloj ne brine o protokolu i načinu dostave paketa narednom mrežnom čvoru. Jedino brine o tome da paket ima sva potrebna zaglavlja, u skladu sa korištenim mrežnim protokolom, Podatkovni sloj, za mrežni, apstrahuje izvedbu veze između mrežnih čvorova između kojih se paket dostavlja.

2.5 Podatkovni (*data link*) sloj

Podatkovni sloj prihvata pakete mrežnog sloja i dostavlja ih mrežnom sloju na slijedećem mrežnom čvoru na putu paketa do odredišta. Podatkovni sloj enkapsulira paket sa mrežnog sloja u svoja zaglavlja i *trailer*, bez potreba da vodi računa o sadržaju i formatu tog paketa. Za podatkovni sloj to je niz bajta koje treba dostaviti.

Podatkovni sloj, za razliku od ostalih slojeva, na paket (bajte koje šalje) dodaje polja zaglavlja na početku, ali i polje na kraju (*trailer*). Iz ovog razloga se jedinica podataka koja se šalje na podatkovnom sloju naziva okvir (*frame*). Polje koje se dodaje na kraju služi za provjeru grešaka. Otkrivanje i obrada grešaka je jedna od uloga podatkovnog sloja.

Kao i na svakom drugom sloju, na podatkovnom sloju u poljima zaglavlja su adrese primaoca i pošiljaoca. Pažljivi čitalac bi mogao postaviti pitanje potrebe za adresiranjem na podatkovnom sloju kada mrežni uređaji između kojih se šalje već imaju jedinstvene adrese na mrežnom sloju. Adrese koje su upisane u paket na mrežnom sloju su adrese početnog i krajnjeg čvora. Ako bi se umjesto adresa krajnjeg čvora upisala adresa narednog mrežnog čvora kom podatkovni sloj treba isporučiti paket, izgubila bi se informacija o konačnom odredištu i mrežni čvor koji je dobio paket ne bi znao koda dalje da ga šalje, i zapravo bi mislio da je za njega. Očigledno je potrebno da postoji informacija i o tome koje je konačno odredište i o tome koji je slijedeći mrežni čvor na putu. Informacija o konačnom odredištu, koju mrežni sloj koristi za donošenje odluke o prosljeđivanju, ostaje neizmijenjena u paketu mrežnog sloja. Informacija o narednom odredištu zapisuje se u zaglavlje podatkovnog sloja u formatu adresa koje koristi protokol podatkovnog sloja.

Adrese, pošiljaoca i primaoca, na podatkovnom sloju su dodijeljene hardveru NIC (interfejsa) od strane proizvođača i ne mijenjaju se. Ove adrese nazivaju se MAC adrese. Svaki proizvođač mrežnih kartica dobija dio MAC adresnog prostora. Unutar tog prostora proizvođač se stara da svaki uređaj dobije jedinstven ostatak adrese. Na taj način se ostvaruje globalna jedinstvenost adresa. Ova adrese nije potrebno unositi u neku konfiguraciju uređaja. Ovo je različito od adresa na mrežnom sloju koje su zavisile od mreže u kojoj se interfejs nalazi.

Izvorišna adresa na podatkovnom sloju je MAC adresa interfejsa sa kog se šalje. Postavlja se pitanje kako podatkovni sloj dođe do odredišne adresa. Podsjećanje da odredišne adrese na transportnom i mrežnom sloju potiču od aplikacije koja je napravila poruku. Ta adresa iz aplikacije ne uključuje adrese na podatkovnom sloju, što je i logično jer su to lokalne adrese koje se koriste samo za dostavu na dijelu puta. Iz opisa rada mrežnog sloja vidljivo je da mrežni sloj određuje adresu, na mrežnom sloju, slijedećeg čvora kom treba isporučiti paket. Na osnovu ove adrese podatkovni sloj može pronaći adresu tog čvora na podatkovnom sloju. Za ove namjene koristi se protokol ARP (*address resolution protocol*).

ARP pronalazi MAC adrese koje odgovaraju IP adresama. Čvor kom je potrebno ovo preslikavanje pošalje ARP upit svim, *broadcast*, čvorovima u mreži (lokalnoj, povezanoj bez rutera). U upitu je IP adresa i traži se da čvor čija je to IP adresa odgovori sa svojom MAC adresom. Taj čvor odgovara sa svoje MAC adrese na MAC adresu pitaoca. Radi uštede vremena i mrežnog saobraćaja odgovor se čuva na čvoru koji je postavio pitanje u ARP tabeli. Kada se neki unos u ARP tabeli ne koristi, zadani period vremena, briše se iz tabele, radi uštede prostora.

Kada čvor pošiljalac dobije MAC adresu narednog čvora može poslati okvir. Kada okvir stigne na svoje odredište, na osnovu MAC adrese, mrežni čvor koji ga dobije skida polja podatkovnog sloja i izdvaja mrežni paket. Ako je paket stigao na konačno odredište, što se zna na osnovu toga što je adresa na mrežnom sloju adresa čvora koji je dobio paket, putovanje paketa je završeno i prosljeđuje se transportnom sloju. Ako je paket stigao do nekog čvora na putu, rutera, na osnovu odredišne adrese na mrežnom sloju i tabele prosljeđivanja mrežni sloj rutera određuje interfejs po kom treba poslati paket i IP adresu narednog čvora kom treba poslati paket. Na osnovu ovih podataka podatkovni sloj pravi okvir sa MAC adresom svog interfejsa sa koga šalje kao izvorišnom, i MAC adresom koja odgovara IP adresi (dobivena ARP-om) narednog čvora kom se paket šalje, kao odredišnom. Svaki ruter na putu paketa mijenja zaglavlje i *trailer* podatkovnog sloja, dok mrežni paket ostaje neizmijenjen.

Podatkovni sloj formira okvir koji kao niz bita šalje po mediju koji se koristi za slanje. Ako je ovaj mediji dijeljen između više čvorova, što vrlo često jeste, potrebno je, na neki način, kontrolisati koji od čvorova u kom trenutku koristi medij. Ovo je još jedna od uloga podatkovnog sloja koja se naziva kontrola pristupa mediju (MAC - *Medium Access Control*). Različiti protokoli mrežnog sloja rješavaju ovo pitanje na različite načine.

Dva najčešće korištena protokola mrežnog sloja, 802.3 Ethernet za žičane i 802.11 WiFi za bežične mreže, su obrađena u nastavku.

2.5.1 802.3 Ethernet

Ethernet je dominantan žičani protokol u računarskim mrežama. Ethernet standarde definiše IEEE 802.3 radna grupa. Iako je prvi Ethernet standard koji se odnosio na 10 Mb/s brzine prenosa usvojen 1983. godine, Ethernet se i danas razvija i koristi, pa se najnoviji standardi odnose na brzine od 400 Gb/s. Očigledno da je Ethernet prilagodljiv i da može pratiti potrebe za povećanje brzine. Prva verzija imala je konkurenciju drugih standarda, ali je uspjela ostvariti dominaciju. Bila je dovoljno dobra uz najnižu cijenu. Ovaj kriteriji se pokazao presudnim u izboru tehnologija, ne samo u računarskim mrežama već bilo gdje se tehnologija ekonomski primjenjuje u praktične svrhe.

Ethernet koristi 48 bitne MAC adrese. Prva 24 bita definišu proizvođača, a druga 24 uređaj tog proizvođača. Ethernet podržava *unicast*, *broadcast* i *multicast* adresiranje i slanje okvira.

Pored polja za MAC adrese odredišta i izvorišta, Ethernet zaglavlje ima još jedno polje. U to polje se zapisuje informacija o tome šta je sadržaj koji Ethernet okvir prenosi. Polje je veličine dva bajta i u njega se upisuje brojčana vrijednost na osnovu koje se može utvrditi tip sadržaja. Najčešći sadržaj je paket mrežnog sloja koji se označava heksadecimalnim brojem 0800 za IPv4 ili heksadecimalnim brojem 86DD za IPv6. Oznaka za okvire koji prenose pomenute ARP pakete je 0806 heksadecimalno. Veličina polja omogućava definisanje velikog broja različitih sadržaja koji se mogu prenijeti u Ethernet okviru. Kako je rečeno podatkovni sloj, ovdje Ethernet, prenosi sadržaj bez potrebe za njegovim tumačenjem.

Veličina sadržaja koje Ethernet može prenijeti je od 46 do 1500 bajta. Ovih 1500 bajta je ranije pomenuti MTU (*maximum transfer unit*) na osnovu kog se računa TCP MSS.

Nakon sadržaja, u Ethernet okviru se nalazi polje za provjeru ispravnosti okvira (FCS). U ovom polju se nalazi 32 bitna CRC vrijednost ostatka okvira. Ethernet primalac odbacuje okvire koji imaju neispravnu vrijednost u FCS, jer na osnovu toga zaključuje da je došlo do greške u prenosu okvira. Ne šalje nikakve potvrde, ni pozitivne ni negativne, tako da pošiljalac ne zna da li je okvir ispravno primljen. Pouzdanost se mora ostvariti na nekom od viših slojeva, što je uglavnom transportni, TCP. U praksi, broj bitskih grešaka kod Ethernet veza je jako mali, reda veličine 10^{-10}, pa je potreba za ponovnim slanjem izuzetno rijetka.

Greške koje Ethernet otkriva i otklanja su greške nastale usljed istovremenog slanja više pošiljalaca na mreži. Ova pojava se naziva kolizija. U počecima Ethernet-a svi pošiljaoci unutar jedne mreže koristili su isti mediji za slanje i kolizije su bile moguće i česte. U savremenim izvedbama mreža ova pojava je onemogućena. Radi kompletnosti i objašnjenja principa u nastavku je kratko opisan Ethernet pristup ovom pitanju.

Otklanjanje grešaka nastalih usljed kolizije se ostvaruje putem Ethernet kontrole pristupa mediju (MAC). Za ovu namjenu koristi se CSMA/CD (*Carrier-sense multiple access with collision detection*) metod. Kod ovog metoda pošiljalac šalje okvir kad utvrdi da niko drugi ne koristi medij. Svi čvorovi mogu slati i primati, pa svi mogu znati da li neko nešto šalje. Kad pošiljalac pošalje okvir on osluškuje da li se na mediju pojavio još neki signal koji se mješa sa njegovim. Ako nije onda nije došlo do kolizije i smatra da je okvir uredno poslan. Ako na mediju čuje signal drugačiji od onog koji je poslao, znači da je signal pomiješan sa signalom drugog pošiljaoca. Došlo je do kolizije i primalac nije mogao ispravno primiti poslani okvir. Da bi obavijestio primaoca o grešci šalje predefinisani signal koji svim čvorovima sa pristupom mediju ukazuje na koliziju i potrebu da prestanu slati. Nakon otkrivanja kolizije, svi čvorovi koji su je izazvali čekaju slučajno vrijeme prije ponovnog slanja. Ideja je da neće izabrati isti trenutak slanja. Ako ponovo dođe do kolizije vremenski interval iz kog se generišu slučajne vrijednosti trenutka slijedećeg pokušaja slanja se duplicira. Ovaj proces se nastavlja dok se okvir ne pošalje bez greške ili dok se ne pokuša maksimalan broj puta, što je 16 za Ethernet. Ovaj pris-

tup nije najefikasniji mogući, ali se pokazao dovoljno dobrim u vrijeme kada se koristio u mrežama koje nisu bile previše opterećenje i u kojim je većinu okvira slao manji broj čvorova.

Pitanje fizičkog povezivanja čvorova u mreži do sada nije razmatrano. Prećutno se podrazumijevalo da postoje fizičke veze koje omogućavaju slanje paketa između od svakog do svakog čvora u mreži. Na podatkovnom sloju to pitanje mora biti razriješeno. Rješenje je isto kao i rješenje za telefonske mreže. Umjesto da je svaki čvor povezan sa svakim čvorom, svi čvorovi su povezani na uređaj koji služi za prespajanje. Takva organizacija omogućava da se svi čvorovi povezuju uz manji broj fizičkih veza, žica, koje bi bile skupe i teške za održavanje. U prvim izvedbama Ethernet centralnog čvora za povezivanje njegova uloga je bila da svaki okvir koji dobije po jednom od svojih ulaza proslijedi na sve svoje izlaze, uz osiguravanje dovoljne jačine signala na svakom od izlaza. Ovaj uređaj naziva se *hub*.

Hub je jednostavan, ali cijelu prenosnu mrežu koju povezuje čvorove pretvara u jedan dijeljeni medij. Time se ograničava da u jednom trenutku samo jedan čvor može slati. Kako u mreži postoji veliki broj veza ka centralnom čvoru, a u jednom trenutku se mogu koristiti samo dvije, taj dizajn ne koristi dobro dostupnu propusnost mreže. To bi bilo kao da se na jednoj telefonskoj centrali može obavljati samo jedan razgovor u jednom trenutku.

Rješenje na koje se brzo prešlo radi na istom principu kao i telefonska centrala, pa mu je i engleski naziv vrlo sličan. Telefonska centrala se na engleskom naziva *switchboard*, a mrežni čvor za povezivanje *switch* (engleski naziv se udomaćio i kod nas). Mrežni *switch*, poput telefonske centrale, okvir koji dobije po jednom ulazu prosljeđuje na samo jedan izlaz na kom se nalazi čvor kome je okvir namijenjen. Na ovaj način omogućeno je da u jednom trenutku više parova čvorova mogu slati i primati poruke. Cijela mreža više nije dijeljeni medij i više nema kolizija.

Po ovoj funkcionalnosti prosljeđivanja okvira sa jednog ulaza na jedan izlaz u zavisnosti od njegovog odredišta, *switch* je sličan ruteru. Jedna razlika je što Ethernet *switch* prosljeđuje okvire na izlaze na osnovu njihove odredišne MAC adrese, a ruter prosljeđuje pakete na osnovu njihove odredišne IP adrese. Druga razlika je način na koji se konstruiše tabela prosljeđivanja. Ethernet *switch* sam popunjava svoju tabelu prosljeđivanja. Nije ga potrebno konfigurisati. U toj tabeli prosljeđivanja se nalaze MAC adrese i odgovarajući izlazi interfejs (koji se obično naziva *port*[1]) *switch*-a na koje treba poslati okvire za te adrese. Kada *switch* dobije paket na nekom *port*-u (jedan fizički interfejs služi kao ulaz i izlaz) od nekog čvora on zaključuje da po tom portu treba slati pakete za taj čvor. U tabelu prosljeđivanja upisuje izvorišnu MAC adresu okvira i port po kom je došao. Iz okvira koji *switch* dobije za prosljeđivanje se pročita odredišna MAC adresa, potraži u tabeli prosljeđivanja i proslijedi na odgovarajući port. Ako u tabeli prosljeđivanja *switch*-a nema odredišne

[1] Port uređaja odnosi se na fizički interfejs i nema veze sa portom na transportnom sloju, kao adresom aplikacije.

MAC adrese okvira, taj okvir se šalje na sve portove, osim onog po kom je došao. Na taj način se osigurava da okvir neće biti odbačen. Kad svi čvorovi u mreži pošalju okvir, tabela prosljeđivanja *switch*-a biće kompletna. Za *switch* se kaže da je samoučeći. Ako se čvor premjesti na drugi port, tabela se ažurira sa prvim slanjem okvira od tog čvora.

Radi podsjećanja na ulogu podatkovnog sloja treba napomenuti da okviri koji idu van mreže imaju kao odredišnu MAC adresu rutera. Iz ugla podatkovnog sloja ruter je konačno odredište na koje treba poslati okvir. Mrežni sloj na ruteru će se pobrinuti za njegovo dalje slanje. Centralni čvor za povezivanja, *hub* ili *switch*, ne mora biti jedan. Kad je broj krajnjih čvorova koje treba povezati veći od broja portova koristi se više uređaja za povezivanje koji su međusobno povezani. Princip rada se ne mijenja.

Svi čvorovi povezani pomoću jednog ili više *switch*-eva, bez posredstva rutera čine jednu mrežu. Povećanjem broja čvorova i *switch*-eva u mreži pojavila se potreba za organizovanjem čvorova u grupe. Ovo je moguće postići povezivanjem čvorova koji pripadaju jednoj grupi na odvojene *switch*-eve od čvorova koji pripadaju drugoj grupi. Ovo zahtjeva posebno ožičavanje i možda nabavku novih *switch*-eva. Alternativa, koja se danas intenzivno koristi su virtuelni LAN-ovi (VLAN). VLAN se konfiguriše na *switch*-u, koji to podržava. Ova konfiguracija omogućava da se skupovi portova *switch*-a pridruže različitim VLAN. Čvorovi koji su povezani na portove koji pripadaju istom VLAN komuniciraju normalno, ali ne mogu komunicirati sa čvorovima koji su povezani na portove koji pripadaju drugom VLAN. Isti skup VLAN može biti konfigurisan na više povezanih *switch*-eva. Kao i kod jednog *switch*-a, čvorovi koji su povezani na portove svih *switch*-eva koji pripadaju istom VLAN komuniciraju kao čvorovi u istoj mreži, a čvorovi koji su povezani na drugi VLAN su kao čvorovi u drugoj mreži. Čvorovi iz dva različita VLAN mogu razmjenjivati pakete samo preko rutera, jer pripadaju različitim mrežama. VLAN omogućava stvaranje više logičkih, virtuelnih. mreža preko jedne fizičke infrastrukture.

Kada se koristi više povezanih *switch*-eva potrebno je omogućiti razlikovanje okvira koji se razmjenjuju između *switch*-eva po tome kom VLAN pripadaju. Za ovo se koriste posebni portovi i posebne oznake okvira. Portovi koji se koriste za povezivanje konfigurišu se kao zajednički (*trunk*), koji pripadaju svim VLAN. Okviri koji razmjenjuju po ovim vezama imju dodatnu oznaku VLAN u zaglavlju. Ova oznaka je definisana IEEE 802.1Q standardom. Ubacuje se u zaglavlje Ethernet okvira između polja za izvorišnu MAC adresu i polja sa informacijom o tipu sadržaja Ethernet paketa. IEEE 802.1Q oznaka je dužine četiri bajta. U prva dva bajta je heksadecimalni brojem 8100. Pošto je ovo polje na mjestu gdje je oznaka sadržaja okvira iz njega se može vidjeti da se radi o 802.1Q okviru. Naredna tri bita su oznaka prioriteta koji se mogu koristiti za različit tretman okvira. Naredni jedan bit se nekad koristio da ukaže na tip MAC adrese, gdje je vrijednost 0 označavala Ethernet okvir. Danas se koristi u kombinaciji sa prethodna tri bita da definiše da li se okvir može odbaciti u slučaju mrežnog zagušenja. Preostalih 12 bita se koriste za identifikaciju VLAN. VLAN, 802.1Q oznake se mogu i slagati (*stack*). U okvir koji

već ima VLAN oznaku može se dodati nova oznaka ispred postojeće. Ovo slaganje omogućava agregaciju okvira iz više različitih VLAN i njihov zajednički tretman na dijelu putanje. Ta mogućnost se danas koristi od strane ISP ili davalaca *triple-play* usluge za odvajanje saobraćaja po klasama ili korisnicima radi pružanje odgovarajućeg QoS.

Izgled Ethernet okvira i Ethernet okvira sa jednom i dvije VLAN oznake prikazan je na slici 2.9.

Odredišni MAC (6 bajta)	Izvorišni MAC (6 bajta)	Tip (2 bajta)	Sadržaj (46 – 1500 bajta)	CRC (4 bajta)

802.1Q oznaka

Odredišni MAC (6 bajta)	Izvorišni MAC (6 bajta)	x8100 (2 bajta)	Pri+F (4 bita)	VLAN ID (12 bita)	Tip (2 bajta)	Sadržaj (46 – 1500 bajta)	CRC (4 bajta)

Odredišni MAC (6 bajta)	Izvorišni MAC (6 bajta)	802.1Q oznaka (4 bajta)	802.1Q oznaka (4 bajta)	Tip (2 bajta)	Sadržaj (46 – 1500 bajta)	CRC (4 bajta)

Slika 2.9: Ethernet okvir, okvir sa jednom i sa dvije 802.1Q oznake

2.5.2 802.11 WiFi

Wi-Fi je dominantan protokol za bežično povezivanje lokalnih računarskih mreža. Zasnovan je na na porodici IEEE 802.11 standarda. Prva verzija standarda usvojena je 1997. godine, ali se, slično kao i Ethernet, standard stalno ažurira sa novim verzijama u skladu sa razvojem potreba i tehnologije. Novije verzije su omogućavala prenos većim brzinama. Najnovija verzija 802.11ax, koja se naziva Wi-Fi 6, ima teoretski maksimalnu brzinu prenosa podataka od 9,6 GB/s. Radi jednostavnijeg razlikovanja novijih i starijih verzija Wi-Fi alijansa, organizacija koja promoviše i certificira WiFi uređaje, uvela je 2018. godine numeraciju. Veći broj uz WiFi znači novija verzija. Oznake IEEE 802.11, kojih je bilo dosta (ali WiFi brojeve su dobile a, b. g, n, ac i ax), su mogle biti zbunjujuće za krajnje korisnike i kupce opreme.

WiFi koristi dvije noseće frekvencija 2,4 i 5 GHz. Različite verzije WiFi koriste različite širine kanala. Novije verzije koriste veću širinu kanala čime, između ostalog, ostvaruju i veće brzine prenosa. WiFi 6 koristi četiri različite širine kanala, 20, 40, 80 i 160 MHz. Obje noseće frekvencije nalaze se u takozvanom ISM (*industrial, scientific and medical*) dijelu frekventnog spektra. Ovo je dio spektra za koji se ne dobivaju posebne dozvole i svako ga može koristiti uz ograničenje snage predajnika. Očigledno je da WiFi može biti, i jeste, izložen smetnjama od drugih predajnika koji rade na tim frekvencijama. Ipak, WiFi je prilagodljiv uslovima na mreži i pokazao se kao dovoljno dobro rješenje za stvarnu upotrebu. Manje je pouzdan od Ethernet-a, ali jednostavniji za uspostavljanje jer nema razvlačenja žica.

WiFi koristi iste 48 bitne MAC adrese kao i Ethernet, te podržava *unicast*, *broadcast* i *multicast* adresiranje i slanje okvira. WiFi MTU je 2304 bajta.

WiFi radi u složenijim uslovima nego Ethernet, pa je protokol nešto složeniji. Zbog ovoga su i WiFi okviri složeniji. Kao i Ethernet, WiFi okviri imaju polja za odredišnu i izvorišnu MAC adresu, kao i polje za provjeru ispravnosti okvira (FCS). FCS je isti za sve IEEE 802 protokole. Prvo polje WiFi okvira naziva se *Frame Control*. Ovo polje definiše o kakvom okviru se radi i od vrijednosti u njegovim podpoljima zavisi tumačenje ostalih polja. Bitno je reći da postoje tri vrste okvira, podatkovni, kontrolni i upravljački. Detalji svakog od okvira mogu se naći u standardu.

Jedna od velikih razlika u odnosu na Ethernet, koja je dovela do potrebe za dodatnim tipovima okvira, je nemogućnost pouzdanog otkrivanja kolizije. Kod Ethernet-a svi čvorovi čiji okviri mogu doći u koliziju mogu čuti jedan drugog, te tako otkriti koliziju. Kod WiFi je moguće da dva pošiljaoca izazovu koliziju na nekom primaocu, a da toga nisu svjesni. Razlog za ovo je slabljenje elektromagnetnih talasa koji se prostiru vazduhom. Signali svakog od pošiljaoca mogu se sudariti na primaocu, ali oslabiti dok dođu do drugog pošiljaoca toliko da ih ne može detektovati. Ovo može biti zbog fizičke udaljenosti ili prepreke između pošiljalaca. Ova pojava se naziva problem skrivenog terminala. Pored ovoga, WiFi slanje ide direktno između čvorova, odnosno nema centralnog čvora preko koga ide razmjena, pa su svi čvorovi u istom kolizionom domenu.

WiFi koristi drugačiju kontrolu pristupa mediju (MAC) od Ethernet-a, koja se naziva CSMA/CA (*Carrier-sense multiple access with collision avoidance*). Kako kolizije nije moguće otkriti, nastoji ih se izbjeći. To se postiže rezervacijom medija za slanje. Pošiljalac koji želi poslati okvir primaocu, prije slanja podataka, šalje poseban kontrolni okvir. Kontrolni okviri su manji jer ne prenose podatke. Naravno i kontrolni okviri se šalju kad pošiljalac "čuje" da je medij slobodan. U ovom kontrolnom okviru pošiljalac navodi kome želi da šalje podatke, u MAC adresi primaoca, i koliko dugo očekuje da će to slanje trajati, u polju trajanje. Ovaj okvir se naziva RTS (*request to send*). Ako dođe do kolizije primalac neće odgovoriti na ovaj okvir, pa će ga pošiljalac opet poslati. Očekivani odgovor je okvir CTS (*clear to send*). MAC primalac ovog okvira je čvor koji je poslao RTS. U ovom okviru navodi se očekivano trajanje slanja podataka, zasnovano na vremenu iz RTS okvira. Namjena ovog okvira je da sve pošiljaoce koji bi mogli izazvati koliziju na prijemnom čvoru obavijesti da pričekaju sa slanjem svojih okvira dok se ovaj dogovoreni prenos podataka ne završi. Pretpostavka je da su čvorovi do kojih dopre ovaj signal oni koji bi mogli izazvati koliziju.

WiFi mreže imaju dva načina rada, infrastrukturni i *ad-hoc*. Kod infrastrukturnog načina rada postoji infrastruktura u obliku jednog ili više posebnih čvorova preko kojih ide sva razmjena podataka. Ti čvorovi se nazivaju pristupne tačke (AP). AP ima identifikaciju bežične mreže, SSID, koju emituje. Čvorovi koje želi postati dio te mreže javljaju se AP. AP ih mora prihvatiti, da bi postali dio mreže, što može zahtijevati neki proces prijavljivanja.

Čvorovi okvire za druge čvorove šalju AP, a AP šalje te okvire ka odredišnom čvoru. čak i ako su ta dva čvora jedan pored drugog. Na ovaj način AP upravlja saobraćajem u mreži. Fizička veličina mreže zavisi od dometa AP. AP su uglavnom povezane na Ethernet i ožičenu infrastrukturu.

Drugi način rada, *ad-hoc*, nema infrastrukturne čvorove, AP. U ovom slučaju čvorovi nisu samo pošiljaoci ili primaoci, već mogu biti i posrednici. Pošto nema potrebe za infrastrukturom ove mreže se lako uspostavljaju. Veličina mreže zavisi od broja čvorova i dometa svakog od njih, a može se protezati preko više čvorova. *Ad-hoc* mreže se sve više koriste tamo gdje nije moguće ili ekonomično praviti infrastrukturu. Jedan od primjera su senzorske mreže.

WiFi prilagođava brzinu slanja intenzitetu bitskih grešaka u komunikaciji. Glavni razlog za bitske greške je slabljenje odnosa signal-šum (S/N). Ako je broj bitskih grešaka prevelik, mijenja se način kodiranja. Prelazi se na kodiranje sa manje stanja, koje je manje osjetljivo na varijacije prijemnog signala izazvane smanjenim S/N. Zbog ove promjene kodiranja dolazi do smanjenja brzina slanja. Naravno, povećanje S/N dovodi do povećanja brzine na isti način. Promjena S/N uzrokovana je udaljenošću predajnika i primaoca, brzinom njihovog kretanja ili drugim elektromagnetnim talasima. U praksi je brzina prenosa uglavnom manja od one koja se navodi u specifikacijama standarda. Ta brzina je maksimalna moguća u idealnim uslovima.

2.6 Nedostaci tradicionalnih računarskih mreža

Tradicionalne računarske mreže izvedene na prethodno opisan način koristeći navedene protokole i danas, nekih 40-ak godina od nastanka, se masovno koriste i dobro obavljaju svoju funkciju. Zapravo obavljaju je toliko dobro da se usluge drugih sistema prenosa podataka, poput telefonskih mreža ili televizijskog emitovanja, prebacuju na računarske mreže. Ti sistemi su radili na drugoj infrastrukturi i sa drugim protokolima. To je bila namjenska infrastruktura i namjenski protokoli koji su obezbjeđivali kvalitet usluge (QoS) primjeren usluzi koju su pružali, uglavnom konverzacija ili slanje multimedijalnih podataka u realnom vremenu. Računarske mreže nisu bile projektovane za prenos takvih informacija kao što su slike i zvuk, već prije svega podatke. Pokazalo se da su računarske mreže i protokoli dovoljno prilagodljivi da uspješno mogu prenositi i multimedijalne sadržaje. Vrlo bitan preduslov za ovo je postojeća infrastruktura globalnog Interneta koja je bila dovoljno raširena, pouzdana i dovoljne propusnosti. Pored toga pokazalo se da je pružanje ovih usluga preko računarske mreže finansijski povoljnije. Kako je ranije rečeno, tehnologija koja je dovoljno dobra, a ima najnižu cijenu pobjeđuje. Tako je bilo i ovdje. Računarske mreže i globalni Internet su preuzele ulogu telefonskog sistema, TV emitera, video klubova, novina i svih drugih vidova razmjene informacija. Ova pojava naziva se mrežna konvergencija. Upotreba jedne mreže

za sve vrste komunikacija naziva se objedinjene komunikacije. To je dobro radilo ali je stvorilo pritisak na postojeću Internet infrastrukturu i protokole.

HTTP, kao dominantni aplikativni protokol na Internetu, podnio je veliki teret i preuzeo na sebe pružanje mnogih usluga za koje nije bio projektovan. Došlo je do situacije u kojoj tradicionalni HTTP više nije mogao obavljati ovu ulogu sa potrebnom efikasnošću. Sigurnost HTTP se postiže upotrebom TLS. Ovaj dodatni sloj, iako neophodan za sigurnost, uvodi dodatno kašnjenje u uspostavljanje konekcija. Integraciju ova dva sloja bilo je potrebno podići na novi nivo. Navedeno je dovelo do najvećih promjena u HTTP od njegovog nastanka, koje se još odvijaju. Tim promjenama posvećeno je naredno poglavlje 3.

Dodavanje mogućnosti komunikacije raznim svakodnevnim uređajima poput mjerača temperature ili grijača proširilo je računarske mreže da obuhvate i ove stvari i napravilo IoT (*Internet of Things*). Mnoge od tih stvari nemaju resurse za izvedbu cijelog skupa protokola uobičajenog za komunikacije u računarskim mrežama. Te uvezane stvari se mogu nalaziti na udaljenim lokacijama do kojih se ne mogu provući žice i do kojih ne dopiru bežični signali postojećih AP. Na tim lokacijama možda nema ni napajanja, pa stvari rade na baterije i od presudnog značaja je da troše minimalnu količinu energije na komunikacije. I ovdje se pokazalo da postojeći protokoli tradicionalnih računarskih mreža nisu adekvatni i da su potrebni novi da bi IoT mogao nastaviti rasti i funkcionisati. Pitanjima IoT umrežavanja posvećeno je poglavlje 4.

Razvoj društvenih mreža i *cloud computing*-a doveo je do promjene organizacije računarskih mreža u kojim sad postoje veliki podatkovni centri iz kojih se pruža većina usluga. Ti centri obrađuju mnogo veće količine podataka nego ikad prije. Imaju više procesorske snage i prostora za pohranu koncentrisane na relativno malom prostoru jedne prostorije ili danas zgrade. Uvezivanje svih tih resursa u mrežu putem koje efikasno razmjenjuju podatke je zadatak za koje tradicionalne računarske mreže više nisu pogodne. Ovom pitanju je posvećeno poglavlje 5.

U sve ovo treba dodati i pojavu mobilnih telefona koji su preuzeli mnoge uloge korisničkih računara, prerasli ih po broju i konstantno razmjenjuju podatke. Sve ove pojave uticale su na prikupljanje slanje i obradu, ranije nezamislivih, količina podataka, pa je stvoren i termin Veliki podaci (*Big Data*) da opiše tu pojavu. Uz svu prilagodljivost i skalabilnost tradicionalnih računarskih mreža promjene zahtjeva bile su prevelike da bi mreže mogle pružati očekivani QoS i QoE (*Quality of Experience*). Potrebni su novi generalni pristupi od kojih je softversko umrežavanje (SDN) onaj od kog se najviše očekuje. SDN je dodatno obrađen u poglavlju 6.

* * *

Ovim je završen pregled tradicionalnih računarskih mreža. Kroz pregled je ukazano kako su principi komuniciranja uklopljeni u rad računarskih mreža.

Podjela problema uvezivanja na manje podprobleme, slojeve, omogućila je skalabilnost i današnji Internet. Pregled je bio relativno kratak da bi se ostavilo prostora za savremena pitanja pomenuta iznad. Za više detalja o tradicionalnim računarskim mrežama preporučuju se dvije odlične knjige [27][29] sa udžbeničkim pristupom koje već imaju veći broj izdanja, kao i novija knjiga [54] koja ima nešto drugačiji, ali vrlo koristan pristup objašnjavanju principa na kojim rada računarske mreže, kako tradicionalne tako i savremene.

3

HTTP

HTTP je dominantan aplikativni protokol na Internetu. Iako je počeo kao protokol za pristup web stranicama, zbog svojih karakteristika i prilagodljivosti omogućava mnogo više. Većina savremenih aplikacija prilagođena je da danas radi preko HTTP-a. Dobar primjer je *streaming* video sadržaja, koji na prvi pogled ne bi trebao nikako biti rađen preko HTTP. Pametnim korištenjem HTTP protokola omogućeno je da se po njemu šalje video koji je pri tome i prilagodljivog kvaliteta uslovima na mreži. Prema Sandvine statistikama iz septembra 2019. HTTP *media stream* je vodeći saobraćaj na Internetu [57].

HTTP je dugo, od 1999. godine, bio u verziji 1.1. Tek, 2015. godine, usvojena je verzija HTTP/2. Iako je ta verzija aktivna još uvijek postoji veliki broj web lokacija koje je ne podržavaju, tako da savremeni web preglednici podržavaju obje verzije. Ovo je prva velika promjena HTTP nakon 16 godina. Izgleda da kad se ta promjena desila, krenule su i druge veće i značajnije promjene. Verzija HTTP/3 je već u razvoju, te čak dostupna, u vrijeme pisanja na *canary* verziji Google Chrome web preglednika i CloudFlare web lokacijama. Ova nova verzija ne radi preko TCP kao dosadašnje, već preko UDP, odnosno QUIC, novog transportnog protokola. Takođe je obavezna upotreba zaštićene TLS konekcije. TLS je takođe 2018. nakon 10 godina prešao sa verzije 1.2 na verziju 1.3. Ova verzija unijela je najveće promjene TLS do sada.

Očigledno je došlo vrijeme intenzivnih promjena u najkorištenijim protokolima za prenos podataka. Ovo poglavlje posvećeno je tim promjenama i objašnjenju novih protokola, onih koji su važeći i onih koji dolaze.

3.1 Razlozi za promjenu HTTP

HTTP protokol je dizajniran da bude jednostavan. Ta jednostavnost je pomogla njegovom širenju i uticala da se danas toliko koristi. Sa druge strane, jednostavnost je počela da negativno utiče na performanse. Za prve web stranice koje su bile jednostavni HTML dokumenti, bio je dovoljan jedan HTTP zahtjev i jedan HTTP odgovor da bi web preglednik dobio, i prikazao, cijelu

stranicu. Tada je sa odgovorom na svaki zahtjev zatvarana TCP konekcija po kojoj su dostavljeni HTTP zahtjev i odgovor. Kad su web stranice počele da imaju više elemenata, poput slika, za koje su bili potrebni dodatni HTTP zahtjevi i odgovori, bilo je potrebno nešto promijeniti. Za svaki zahtjev trebalo je uspostavljati novu TCP konekciju što je, nepotrebno, trošilo vrijeme. Uvedene su, takozvane, perzistentne HTTP konekcije, u HTTP 1.1. Sada se TCP konekcija nije prekidala nakon HTTP odgovora već se držala otvorenom dok ne istekne vrijeme čekanja na zahtjev, nakon kog se zaključivalo da zahtjeva više nema. Ovo je ubrzalo HTTP, ali je povećalo zahtjeve na servere, koji su sada držali konekcije duže otvorenim.

Uz perzistentne konekcije pojavio se još jedan pristup za ubrzavanje učitavanja web stranica, HTTP *pipelining*. Web klijent šalje više HTTP zahtjeva ka serveru bez čekanja na pojedinačne odgovore. Ovo omogućava serveru da istovremeno obrađuje više zahtjeva od istog klijenta. Time se uglavnom ubrzava odgovaranje servera i učitavanje stranica, pogotovo kod konekcija sa velikim RTT. Ipak postoji ograničenje, da server mora slati odgovore na zahtjeve onim redom kako su zahtjevi pristigli. Ako serveru treba više vremena da odgovori na neki zahtjev, svi odgovori na kasnije zahtjeve makar i bili spremni, moraju čekati. ovo se naziva *head-of-line blocking*.

Pokazalo se da ni perzistentne konekcije ni HTTP *pipelinig* nisu dovoljni, jer su web stranice sadržavale sve više resursa (osim slika tu su i CSS, JavaScript, ...) koje je trebalo dobaviti, jedan po jedan, putem HTTP zahtjeva. Dodatno ubrzavanje učitavanja web stranica bilo je moguće ostvariti paralelnim učitavanjem više resursa istovremeno. Kako HTTP 1.1 to ne podržava proizvođači web preglednika su smislili drugo rješenje. Web preglednici otvaraju više TCP konekcija ka web serveru istovremeno i po svakoj od konekcija preuzimaju dio objekata potrebnih za prikazivanje web stranice. Ovim se ubrzava učitavanje stranica, ali se povećavaju zahtjevi na resurse klijenta i servera, te povećava kompleksnost upravljanja sa više konekcija. Savremeni web preglednici su našli ravnotežu između ubrzavanja i povećanja opterećenja u otvaranju do šest TCP konekcija prema jednoj web lokaciji istovremeno.

Autori web stranica i aplikacija uvodili su još neke izmjene sa ciljem ubrzavanja web stranica koje poslužuju. Domen *sharding* je pristup u kom web server poslužuje resurse potrebne za formiranje web stranice sa više različitih domenskih imena. Imena su uglavnom slična poput www.nesto.ba, www1.nesto.ba, www2.nesto.ba. ..., i pod kontrolom istog davaoca usluge, a moguće i na istoj IP adresi. Ovim se web preglednicima omogućava da otvore po šest (broj iz prethodne stavke) konekcija sa svakim od web servere koji poslužuju objekte za ove domene. Time se dodatno ubrzava učitavanje web stranica, ali po cijenu uvođenja novih domenskih imena, DNS upita i TCP konekcija. Još jedna tehnike ubrzavanja je povezivanje više JavaScript i CSS datoteka u jednu i/ili brisanje "space" znakova koji nisu neophodni, radi bržeg prenosa. Slično ovom je i pravljenje "image sprites", što je više slika objedinjeno u jednu datoteku. One se kasnije mogu pojedinačno prikazivati upotre-

bom CSS. Koristi se i umetanje sadržaja CSS i JavaScript datoteka i, base64 kodiranih, slika direktno u HTLM datoteku web stranice.

Svi ovi "trikovi" ukazivali su na potrebu da se HTTP izmjeni na način da podržava ono što autori web preglednika i stranica rade da bi ubrzali prikazivanje web stranica preko HTTP 1.1.

3.2 HTTP/2

HTTP/2 je postao zvaničan standard usvajanjem RFC 7540 [32], 2015. godine. Razvoj standarda trajao je oko tri godine, od prvog nacrta. U međuvremenu je važeći standard HTTP 1.1 ažuriran, posljednji put 2014. u nizu RFC od 7230 do 7235.

Fokus izmjena u odnosu na prethodnu verziju bio je na poboljšavanju performansi. Iskorišteno je iskustvo optimizacije HTTP 1.1 da bi se otkrila uska grla i predložila poboljšanja. Kao rezultat izmjena HTTP/2 je postigao i manje opterećenje za mrežu jer se isti efekat postiže sa manje TCP konekcija nego kod HTTP 1.1. Svi savremeni web preglednici i web serveri podržavaju HTTP/2. Međutim broj web lokacija koje podržavaju HTTP/2 je još uvijek, u oktobru 2019., ispod 50% [64]. Ipak, pošto velika većina najposjećenijih web lokacija podržava HTTP/2, procenat HTTP/2 saobraćaja je odavno prešao 70% HTTP saobraćaja.

HTTP/2 je razvijen na osnovu ideja koje su bile ugrađene u Google eksperimentalni protokol SPDY. Ovaj protokol, iako nije bio zvaničan standard, bio je podržan od strane savremenih web preglednika, do usvajanja HTTP/2 standarda.

HTTP/2 podržava kompatibilnost unazad za aplikacije. Semantika i osnovni koncepti, poput HTTP metoda, polja zaglavlja, statusnih kodova i URI-a, nisu se promijenili. Ono što HTTP/2 mijenja je način formatiranja i prenosa podataka kojim upravljaju web klijent i server. Ova promjena je sakrivena od aplikacija i one se ne moraju mijenjati.

Konkretne promjene vezane za ubrzavanje koje je HTTP/2 donio u odnosu na HTTP 1.1 su:

- binarni format
- multipleksiranje zahtjeva i odgovora
- prioritetizacija
- kompresija zaglavlja
- serverski *push*

Svaka od ovih izmjena objašnjena je u nastavku.

3.2.1 Binarni format

HTTP/2 binarno kodira poruke. HTTP poruke. Zahtjevi i odgovori, više nisu ASCII tekst u više linija koji i ljudi mogu čitati. Binarno kodiranje je efikasnije, a čitljivost poruka za ljude nije od presudnog značaja. Ovo kodiranje (i

dekodiranje) rade HTTP/2 klijent i server. Aplikacije nisu svjesne ovog kodiranja i ne moraju se mijenjati. Međutim i HTTP server i HTTP klijent moraju podržavati ovo kodiranje da bi se mogli razumjeti. HTTP 1.1 klijent ne može raditi sa HTTP/2 serverom i obratno. Binarnim kodiranjem smanjuje se i veličina zaglavlja. Prelazak na binarno kodiranje bio je neophodan da bi se omogućile ostale promjene

3.2.2 Multipleksiranje zahtjeva i odgovora

Moguće je poslati više HTTP zahtjeva po jednoj HTTP konekciji, bez čekanja na odgovore, što je bilo moguće i u HTTP 1.1. Međutim, kod HTTP/2 odgovori se mogu slati redom koji je drugačiji od reda prijema zahtjeva. Ovim je prestala potreba za uspostavljanje višestrukih TCP konekcija. Sada klijenti mogu bez čekanja, po jednoj TCP, konekciji poslati zahtjeve za sve objekte koji su im potrebni za prikazivanje web stranice. Uklonjena je i pojava *head-of-line blocking* na nivou HTTP transakcija.[1] Također više nema potrebe za ostalim tehnikama ubrzavanja pomenutim u 3.1 koje su koristile web lokacije.

Multipleksiranje je omogućeno drugačijom organizacijom slanja podataka između HTTP klijenta i servera. Umjesto slanja HTTP zahtjeva i odgovora, sada je osnovna jedinica slanja okvir (*frame*). Okviri su, kako je navedeno, binarno kodirani. Svaki okvir nosi neki specifičan tip podataka, recimo HTTP zaglavlje, HTTP sadržaj, ... Postoji 10 tipova okvira od kojih su najvažnija dva prethodno pomenuta *Headers* i *Data Frame*. Svaki okvir ime identifikaciju toka (*stream*) kom pripada. Jedan ili više okvira čini poruku (*message*), kompletan niz okvira, koji predstavlja HTTP zahtjev ili odgovor. Niz bidirekcionih poruka čini jedan tok. Svaki tok ima jedinstveni identifikator. Tok u HTTP/2 je zapravo niz HTTP zahtjeva i odgovora koji se u HTTP 1.1 slao po jednoj TCP konekciji. Svi tokovi se, u HTTP/2, šalju unutar jedne TCP konekcije.

3.2.3 Prioritetizacija

Da bi osigurao pravovremenu isporuku HTTP odgovora u potrebnom redoslijedu (kada su oni ubačeni u okvire, poruke, pa tokove), HTTP/2 omogućava prioritetizaciju tokova. Prioritetizacija se radi na osnovu težine i zavisnosti, koje su definisane za svaki tok. Težina može imati vrijednost od 1 do 256. Zavisnost nekog toka je broj toka od kog ovaj tok zavisi. HTTP/2 klijent može, putem težina i zavisnosti koje dodjeljuje svakom od tokova, omogućiti serveru da utvrdi kojim redoslijedom bi klijent želio dobivati odgovore. Zavisnost definiše koji tokovi se obrađuju prije. Tokove od kojih zavise drugi tokovi server bi trebao obrađivati i slati odgovore ranije. Odnos težina tokova koji

[1] HTTP zahtjevi i odgovori se prenose unutar TCP konekcije koja se prenosi po IP paketima. IP paket koji kasni zadržava isporuku paketa koji dolaze nakon njega u TCP konekciji, što je opet *head-of-line blocking*, ali na nivou paketa.

zavise od istog toka, ili su nezavisni, određuje odnos resursa koje treba da dobiju za obradu (CPU, memorija, ...) i slanje (mrežna propusnost). Oznake prioriteta su samo indikacija želja klijenta, koje server može ispoštovati, ako je u mogućnosti, ali ne mora.

3.2.4 Kompresija zaglavlja

Svaki HTTP zahtjev i odgovor se sastoji od zaglavlja i tijela (sadržaja). Kako je HTTP bez stanja (*stateless*), svaki zahtjev i odgovor mora biti kompletan i omogućiti drugoj strani da ga razumije, bez pamćenja prethodnih poruka. Radi bržeg prenosa, još u HTTP verziji 1.1, uvedena je kompresija tijela poruka (zahtjeva i odgovora). Iako zaglavlja nisu velika i uglavnom su manja od tijela odgovora, ipak svako je veličine nekoliko stotina bajta. Na velikom broju zahtjeva i odgovora ukupan broj bajta koje je potrebno prenijeti nije zanemariva. HTTP/2 vrši kompresiju korištenjem HPACK algoritma. Ovaj algoritam je detaljno opisan u RFC 7541 [52]. Principijelno kompresija se ostvaruje na dva načina.

Prilikom slanja polja zaglavlja ona su kodirana Huffmanovim kodom. Time se smanjuje njihova veličina.

Većina polja zaglavlja, zajedno sa njihovim vrijednostima, se ponavlja između uzastopnih zahtjeva/odgovora. Zaglavlja koja se ponavljaju ne šalju se ponovo. HTTP/2 klijent i server čuvaju indeksirane liste zaglavlja koja su prethodno dobili. Za ta zaglavlja pošiljalac može samo poslati njihov indeks, što je dodatna ušteda u odnosu na slanje kompresovane vrijednosti.

3.2.5 Server *push*

Kod HTTP/2, server ima mogućnost da klijentu dostavi neke objekte bez potrebe da ih klijent zatraži. Na osnovu zahtjeva klijenta za nekim objektom, server nakon slanja tog objekta, šalje klijentu odmah i druge za koje zaključi da će mu biti potrebni. Očigledan primjer za ovo je kad klijent sa prvim zahtjevom traži objekat u kom se nalazi HTML kod web stranice. U kodu stranice su referencirani različiti objekti, poput CSS datoteke i slika, koji su potrebni klijentu za prikazivanje stranice. Server klijentu šalje traženu datoteku sa HTML kodom, ali mu odmah zatim šalje i CSS datoteku i slike, bez čekanja na zahtjev od klijenta. Na ovaj način se štede mrežni resursi potrebni za slanje zahtjeva i ubrzava proces prikazivanje stranice jer se ne čeka na zahtjeve da bi se poslali potrebni objekti.

Iako server može raditi *push*, ovaj proces je pod kontrolom klijenta. Klijent može serveru reći da ne radi *push* ili odbaciti objekte dostavljene na ovaj način.

Zbog prethodnih izmjena HTTP/2 konekcije su perzistentne. Više nema potrebe za više TCP konekcija između istog klijenta i servera. Ovim se i

prosječni životni vijek TCP konekcija koje koristi HTTP produžio. TCP je optimiziran za duže konekcije, pa se na ovaj način poboljšala iskorištenost u odnosu na stari HTTP 1.1 sa više kraćih HTTP konekcija.

HTTP/2 ne zahtjeva upotrebu TLS. U koliko se koristi TLS mora se koristiti verzija 1.2 ili novija. U praksi izvedbe svih savremenih web preglednika podržavaju HTTP/2 samo preko TLS. Na taj je način *de facto* obavezna upotreba HTTP/2 samo po sigurnim konekcijama.

3.3 TLS 1.3

TLS (Transport Layer Security) je preovladavajući protokol za zaštitu HTTP saobraćaja. Uzimajući u obzir dominantnu ulogu weba u Internet trgovini i ekonomiji može se reći da je TLS najvažniji protokol za sigurnost e-ekonomije. TLS protokol umetnut je između aplikativnog i transportnog sloja. Prilikom upotrebe TLS aplikacije poruke prosljeđuju TLS-u umjesto TCP-u, a TLS nakon obrade poruka koju obavlja prosljeđuje TLS segmete TCP-u. Iz navedenog se vidi da se TLS može koristiti, i koristi se, i za osiguravanje drugih aplikativnih protokola. Ovdje se koristi u kontekstu HTTP jer, kako je prethodno rečeno, TLS postaje praktično neodvojiv dio HTTP/2.

TLS je nastao iz Netscape SSL (Secure Socket Layer) protokola. IETF je 1999. usvojio prvu verziju TLS 1.0. Uslijedile su verzije 1.1 2006. i 1.2 2008. Ove verzije nisu značajnije mijenjale rad TLS nego su uglavnom otklanjale pronađene sigurnosne nedostatke prethodnih verzija. Najnovija verzija TLS 1.3 usvojena je 2018. kroz RFC 8446 [53]. Ova verzija unijela je najveće promjene do sada. Ove promjene i logika iza njih opisani su u nastavku.

3.3.1 Pregled rada TLS 1.2 i nedostaci

Upotrebom TLS protokola ostvaruje se povjerljivost i integritet razmijenjenih poruka, te potvrđuje identitet servera. Sve navedeno postiže se upotrebom kriptografije. Konkretno kod TLS koristi se, takozvana hibridna kriptografija. Kod hibridne kriptografije koristi se asimetrična kriptografija za sigurnu razmjenu sesijskog ključa koji će se koristiti za simetrično šifriranje poruka koje se razmjenjuju. Time se ostvaruje povjerljivost. Integritet i autentičnost pošiljaoca poruka se ostvaruje upotrebom koda za autentičnost poruka (MAC). U principu to je *hash*-iranje sa ključem. Identitet servera potvrđuje se na osnovu certifikata koji server dostavlja, provjerom koju obavlja klijent koristeći listu certifikacijskih ustanova čije javne ključeve posjeduje.

Kako postoje različiti algoritmi za razmjenu ključeva, simetrično šifriranje i MAC sa različitim parametrima, neophodno je da se TLS klijent i server dogovore o njima. Pored toga server mora potvrditi identitet i ključ za simetrično šifriranje se mora dogovoriti. Ovaj proces se odvija u nekoliko razmjena poruka između TLS klijenta i servera. U pojednostavljenom obliku proces uspostavljanja TLS 1.2 konekcije sastoji se od slijedećih koraka:

1. Klijent serveru šalje pozdravnu poruku (`ClientHello`) sa prijedlogom algoritama/parametara za njihovu komunikaciju i slučajni broj (*nonce*), jedinstven za tu konekciju;
2. Server odgovara svojom pozdravnom porukom (`ServerHello`) u kojoj je odabir predloženih algoritama/parametara i njegov slučajni broj (*nonce*), te digitalni certifikat u kom se nalazi njegovo DNS ime;
3. Klijent dostavlja serveru novi slučajni broj (*premaster secret*), ali ga, radi očuvanja njegove tajnosti, šifrira serverovim javnim ključem (E_{SJK})
   - *premaster secret* (PMS), skupa sa klijentskim i serverskim *nonce* koristi se za generisanje istog skupa ključeva kod servera i klijenta koji se koriste u nastavku TLS sesije;
4. Server potvrđuje uspostavljanje parametara konekcije i ključeva

Nakon uspostavljanja konekcije počinje razmjena aplikativnih podataka, poput HTTP poruka, po sigurnoj TLS konekciji.

Koraci uspostavljanje TLS 1.2 konekcije prikazani su na slici 3.1.

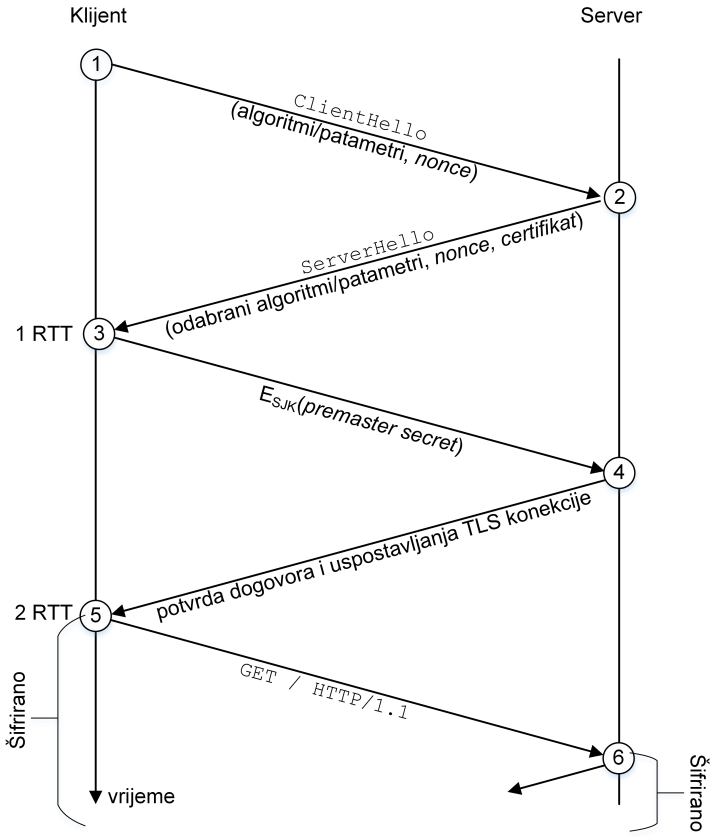
Iz pregleda proces uspostavljanja TLS konekcije može se vidjeti da on dodaje dvije razmjene poruka između klijenta i servera prije nego što krene razmjena HTTP (koji je fokus ovog poglavlja) poruka. To uvodi dodatno kašnjenje od dva RTT, što je oko dva puta po 100 ms. U potpodglavlju o HTTP/2 rečeno je da je jedan od ciljeva novog protokola bio ubrzavanje konekcije. Slično je i sa promjenom TLS čiji je jedan od ciljeva bio skraćivanje procesa uspostavljanja TLS konekcije. Time se smanjuje kašnjenje.

Drugi, po redoslijedu nabrajanja, ali ne i važnosti, cilj je povećavanje sigurnosti. TLS 1.2 je bio maksimalno siguran koliko je to bilo moguće ostvariti bez ozbiljnijih promjena TLS protokola. Uspješni napadi na TLS najčešće su bili napadi na proces rukovanja u kojim su strane u komunikaciji (klijent i server) bili prevareni od strane napadača da u procesu dogovora sigurnosnih algoritama/parametara izaberu starije i slabije, za koje postoje sigurnosni propusti. Teorija kriptografije i formalne provjere sigurnosti protokola napredovale su od 1990-tih kada je osmišljen TLS. Bilo je vrijeme da se primjene za pravljenje nove verzije TLS. Bilo je i propusta u izvedbi protokola u raznim softverima, ali to nije bio propust protokola.

3.3.2 TLS 1.3 izmjene

TLS verzija 1.3 napravila je najveće izmjene TLS do sada. Smanjen je broj koraka prilikom uspostavljanja konekcije. Umanjen je dio uspostavljanja konekcije koji se odvija nešifrirano. Uklonjen je veliki broj starih protokola/parametara koji su imali sigurnosne propuste. U nastavku je kratak opis kako su ove izmjene napravljene.

Prvi i vrlo važan korak koji je trebalo učiniti je uklanjanje svih algoritama šifriranja i načina koji nisu sigurni. Time se povećava sigurnost jer se smanjuje, ako ne i eliminiše, mogućnost da se TLS klijent i server prevare da izaberu neki



Slika 3.1: TLS 1.2 uspostavljanje konekcije

od manje sigurnih algoritama/parametara. Sa smanjenjem izbora smanjila se potreba za pregovaranjem oko izbora koje je bilo prva razmjena između klijenta i servera kod ranijih verzija TLS.

Dogovor oko ključa

Izmjena vezana za uklanjanje algoritama koji su se pokazali nesigurni izvršena je u fazi dogovora oko ključa za simetrično šifriranje. U opisu koraka uspostavljanja TLS 1.2 konekcije navedeno je da klijent generiše slučajan broj (PMS) koji dostavlja serveru šifriran serverovim javnim ključem. Nedostatak ove razmjene je što potencijalno nije sigurna u budućnosti. Ako se sazna privatni ključ servera, iz snimljenog mrežnog saobraćaja, može se doći do PMS te dešifrovati sve ikad razmijenjene poruke.

Da bi se ovo spriječilo, TLS 1.3. koristi Diffie-Hellman razmjenu ključeva. Kod ove razmjene ključeva svaka strana, i klijenti server, generiše svoj par

ključeva, privatni i javni. Veza između privatnog i javnog ključa definisana je matematičkom relacijom i parametrima koji su dogovoreni između klijenta i servera. Ovi parametri i matematička relacija nisu tajni. Ipak se iz javnog ključa ne može doći do privatnog. Klijent i server pošalju onom drugom svoj javni ključ. Svaka od strana, uvrštavajući javni ključ koji je dobila od druge strane i svoj privatni ključ u matematičku relaciju zasnovanu na istim dogovorenim parametrima, dobiva vrijednost koja je ista i na klijentu i na serveru Ova vrijednost se koristi kao PMS.

Ako klijent i server generišu različite parove ključeva za svaku konekciju ovakva razmjena ključeva se naziva *ephemeral*. Takva razmjena ključeva ima osobinu *forward* sigurnosti, odnosno sigurna je u budućnosti. Onaj ko bi došao do privatnog ključa klijenta ili servera mogao bi dešifrovati samo poruke iz konekcije u kojoj je korišten taj ključ. TLS 1.3 propisuje upotrebu samo *ephemeral* Diffie-Hellman razmjene ključeva. Pored toga, pošto se pokazalo da neki parametri Diffie-Hellman razmjene mogu dovesti do nesigurnosti, ograničen je izbor parametara na samo nekoliko koji su pouzdano sigurni.

Bitno je napomenuti da je sada razdvojen postupak dogovora oko ključa od potvrđivanja identiteta servera. Kod ranijih verzija TLS server je potvrđivao svoj identitet time što je mogao dešifrirati PMS koji je bio šifriran njegovim, trajnim, javnim ključem. Ovdje se PMS generiše upotrebom parova privremenih ključeva i ovi ključevi se ne koriste za potvrđivanje identiteta servera. Način potvrđivanje identiteta objašnjen je ispod.

Potvrđivanje identiteta servera i zaštita dogovora oko ključeva

Da bi se otežali napadi na proces dogovora, TLS 1.3 počinje šifriranje ovog procesa ranije nego TLS 1.2. Čim dobije javni ključ klijenta (u `ClientHello` poruci) i pošalje svoj javni ključ (u `ServerHello` poruci), server ima ključ koji dijeli sa klijentom i odmah ga počinje koristiti za šifriranje. Slijedeća poruka koju server dostavlja nakon `ServerHello`, bez čekanja na klijenta je šifrirana. Ta poruka se koristi za potvrđivanje identiteta servera. Server, kao i kod kod TLS 1.2 dostavlja certifikat sa javnim ključem potpisan od strane certifikacijske ustanove. Svoj identitet server potvrđuje dokazom da ima pristup privatnom ključu koji odgovara javnom ključu iz certifikata. Pristup privatnom ključu server dokazuje potpisivanjem, sa tim privatnim ključem, *hash*-a poruka razmijenjenih tokom dogovora oko ključa. Pored potvrđivanja identiteta na ovaj način se štiti integritet poruka dogovaranja jer klijent provjerom potpisa potvrđuje i da ove poruke nisu izmijenjene.

Algoritmi za šifriranje, integritet i autentičnost poruka

Skup algoritama za simetrično šifriranje znatno je smanjen. Uklonjeni su svi algoritmi koji su postojali radi kompatibilnosti unazad. Jedini preostali su AEAD (*Authenticated Encryption with Associated Data*) algoritmi. Ovi algoritmi omogućavaju istovremeno šifriranje i potvrđivanje autentičnosti. Na

taj način prestala je potreba za MAC algoritmima za zaštitu autentičnosti poruka. Uklanjanjem nesigurnih algoritama onemogućeni su napadi zasnovani na izboru takvih algoritama. Smanjenjem skupa šifratora olakšan je, i ubrzan, dogovor.

Koraci

Proces uspostave TLS 1.3 konekcija, u skladu sa opisanim izmjenama, sastoji se od slijedećih koraka:

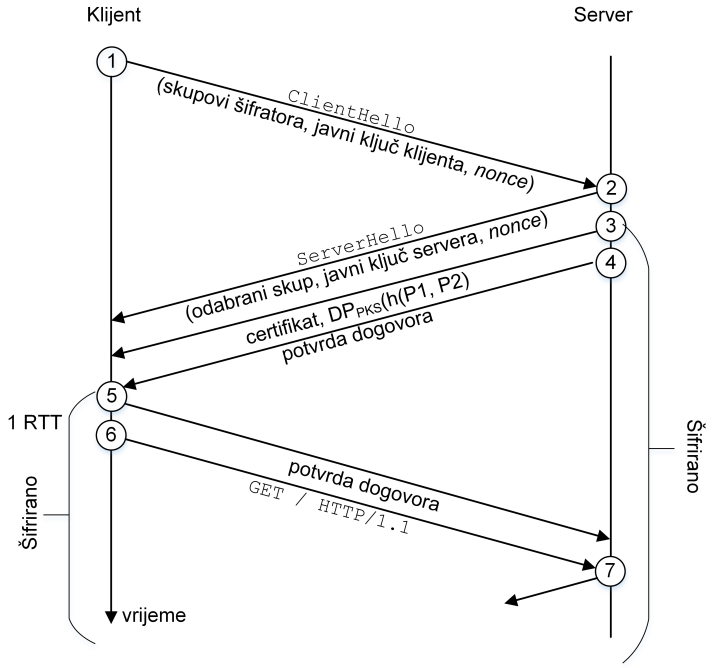
1. Klijent serveru šalje pozdravnu poruku (`ClientHello`) sa prijedlogom skupova šifratora koje klijent podržava (koji je sada prilično mali), javni ključ klijenta (jedinstven za tu sesiju) za razmjene ključeva koje klijent podržava i slučajni broj (*nonce*), jedinstven za tu konekciju;
2. Server odgovara svojom pozdravnom porukom (`ServerHello`) u kojoj je odabrani skup šifratora, javni ključ servera (jedinstven za tu sesiju) za razmjenu ključeva koju je server odabrao i njegov slučajni broj (*nonce*);
3. Server, bez čekanja na klijenta šalje narednu poruku, koja je šifrirana sa ključem koji su klijent i server izračunali na osnovu prve dvije poruke. U toj poruci je digitalni certifikat servera u kom se nalazi njegovo DNS ime, digitalno potpisan *hash*-a prethodne dvije poruke sa privatnim ključem koji odgovara javnom ključu iz certifikata ($(DP_{PKS}(h(P1, P2))$;
4. Server zatim, opet bez čekanja na klijenta, šalje poruku da je završio sa dogovaranjem parametara.
5. Klijent šalje poruku da je završio sa dogovaranjem parametara.
6. Klijent zatim, bez čekanja na server, počinje slanje aplikativnih podataka šifriranih sa izabranim AEAD algoritmom koji osigurava povjerljivost i integritet (Potrebno je ukazati na to da ovo slanje podataka počinje oko jedan RTT nakon početka konekcije, čime je vrijeme do početka slanja podataka duplo skraćeno u odnosu na TLS 1.2).

Koraci uspostavljanje TLS 1.3 konekcije prikazani su na slici 3.2.

Ponovno uspostavljanje konekcije bez čekanja (0-RTT)

Radi dodatnog ubrzavanja, TLS 1.3 omogućava slanje podatka već u prvoj poruci klijenta ka serveru, ako su ranije imali uspostavljenu TLS 1.3 konekciju. Za to se koristi takozvani PSK (*pre-shared key*) način rada. Da bi se to omogućilo, klijent i server, tokom trajanja prve konekcije između njih, uspostave tajnu informaciju koja se naziva *resumption master secret*. Prilikom uspostavljanja nove konekcije klijent može iskoristiti ovu tajnu informaciju za šifriranje podataka koje onda može slati već u prvoj poruci ka serveru.

Postoje određene smetnje uvođenju novih protokola na Internetu, pogotovo onih nisu u potpunosti kompatibilni unazad. Iako se TLS protokol



Slika 3.2: TLS 1.3 uspostavljanje konekcije

izvršava u krajnjim tačkama konekcije, klijent i server, paketi između njih putuju kroz različite mrežne uređaje. Neki od uređaja su ruteri i *switch*-evi, koji su neophodni za rad. Drugi uređaji kroz koje saobraćaj prolazi su dodani radi sigurnosti (*firewall*, IDS, IPS, ...), poboljšanja performansi (*proxy*, *cache*, ...) nadzora i evidentiranja saobraćaja ili za druge namjene (NAT, ...). Ovi drugi uređaji se nazivaju zajedničkim imenom *middlebox*-ovi. Softver u ovim uređajima se rjeđe ažurira od softvera na krajnjim uređajima. Iz tog razloga može nepravilno protumačiti ili izmijeniti polja u paketima sa verzijama protokola koje su novije od onih za koje je on programiran. Ova pojave je poznata u računarskim mrežama jer otežava i usporava uvođenje novih protokola i naziva se okoštavanje (*ossification*).

Da bi se smanjila mogućnost smetnji TLS 1.3 saobraćaju od strane *middlebox*-ova, TLS 1.3 paketi su napravljeni da izgledaju kao paketi ponovnog uspostavljanja TLS 1.2. sesije. Takva izvedba olakšala je uvođenje TLS 1.3 u širu upotrebu. Sa druge strane protokol je postao komplikovaniji nego što je neophodno za komunikaciju krajnjih uređaja. Slična situacije je bila i sa novim transportnim protokolom QUIC koji je opisan u nastavku.

3.4 QUIC

HTTP/2 je smanjio potreban broj konekcija i ubrzao učitavanje web stranica. TLS 1.3 je smanjio broj koraka prilikom uspostavljanja TLS sesije. Za dalja ubrzanja bile su potrebne izmjene na transportnom sloju. Sve verzije HTTP, sa ili bez TLS, koriste TCP kao transportni protokol. Prije slanja podataka TCP provodi uspostavlja konekciju kroz tri poruke (*3-way handshake*). Ove tri poruke uvode kašnjenje od 1,5 RTT. Sa druge strane UDP, nekonekcioni transportni protokol, šalje podatke odmah, bez uspostavljanja konekcije (i bez kašnjenja). Nedostatak UDP je nepouzdanost koja se ogleda u nedostatku garancije isporuke segmenata i isporuke po redu. HTTP očekuje pouzdanu isporuku segmenata po redu. Da bi se ubrzalo uspostavljanje konekcija, ali i podržala pouzdanost prenosa bio je potreban novi transportni protokol.

QUIC je, slično kao i SPDY, krenuo kao Google protokol 2012. godine. Nakon četiri godine upotrebe i usavršavanja, Google je 2016. godine dostavio QUIC IETF-u na razmatranje i standardizaciju. Od tada je proces dorade prošao kroz brojne izmjene u odnosu na Google verziju i još uvijek je u statusu nacrta. U vrijeme pisanja, februar 2020. godine, aktuelna verzija je 27 [25]. Protokol se razvija uporedo sa novom verzijom HTTP broj 3, koji je prvi aplikativni protokol koji koristi QUIC i koji je opisan u narednom potpoglavlju. Već postoji više izvedbi QUIC protokola koje se koriste.

Pored ubrzavanja konekcija, smanjivanjem broja koraka prilikom uspostavljanja konekcije, QUIC donosi još neke promjene. Neke promjene su rezultat iskustava iz razvoja HTTP/2, poput rada sa *stream*-ovima i otklanjanja HOL zadržavanja koje TCP unosi u HTTP/2. Druge su izazvane potrebom za pouzdan prenos, poput korištenje FEC (*Forward Error Correction*), kontrole toka i zagušenja. Neke unose nove mogućnosti kao što je mogućnost očuvanja konekcije prilikom promjene mreža. Neke su bitne za sigurnost ali i prolazak kroz *middlebox*-ove, kao što je obavezna upotreba TLS, minimalno 1.3. Sve pomenute promjene su opisane u nastavku.

Brže uspostavljanje konekcije

QUIC smanjuje broj poruka koje razmjenjuju klijenti i server prilikom uspostavljanja konekcije. Za uspostavu konekcije uz upotrebu TLS 1.3 preko TCP potreban je bar jedan RTT za uspostavljanje TCP konekcije i bar još jedan za uspostavljanje TLS 1.3 konekcije. QUIC preskače uspostavljanje TCP konekcije i odmah prelazi na TLS. Uspostavljanje QUIC konekcija uključuje uspostavljanje i TLS konekcije. Na ovaj način se prva HTTP poruka od klijenta ka serveru može poslati već nakon jednog RTT. Pošto QUIC podržava TLS 1.3 brzo uspostavljanje konekcije između klijenta i servera koji su se ranije povezivali, za takve slučajeve HTTP poruka se šalje odmah, bez ikakvog čekanja. Kada se ovo uporedi sa čekanjem radi uspostavljanja konekcije na slanje prve HTTP poruke po TLS 1.2 preko TCP, koje je trajalo 1,5 RTT za TCP i dva RTT za TLS, dobije se značajno ubrzanje.

Podrška za *stream*-ove i otklanjanje HOL

Za razliku od TCP, QUIC podržava *stream*-ove. U jednom QUIC paketu može biti multipleksirano više *stream*-ova. HTTP/2 *stream*-ovi se preslikavaju u QUIC *stream*-ove. *Stream* predstavlja niz bajta koji se dostavljaju po redu. QUIC *stream* može biti dvosmjeran ili jednosmjeran. QUIC garantuje dostavu bajta po redu unutar jednog *stream*-a, ali ne daje nikakve garancije na redoslijed dostave različitih *stream*-ova.

Kašnjenje ili gubitak jednog TCP segmenta izazivalo je čekanje svih *stream*-ova koji su bili u tom segmentu. To se naziva TCP HOL (*Head of Line*) blokiranje. Iako se slična stvar može dogoditi i sa QUIC, QUIC izvedbe treba da pakuju što manje različitih *stream*-ova u jedan paket. Na taj način kašnjenje ili gubitak QUIC paketa utiče samo na *stream*-ove iz tog paketa. Ostali *stream*-ovi nisu pogođeni ovim događajem.

QUIC ima ugrađenu kontrolu toka po *stream*-ovima, kao i ukupnu po konekciji.

Pouzdanost i kontrola zagušenja

QUIC ima podršku za otkrivanje gubitaka i kontrolu zagušenja. Ove funkcionalnosti su opisane u posebnom IETF dokumentu. Taj dokument je kao i osnovni QUIC dokument je još u fazi nacrta. Namjena otkrivanja gubitaka i kontrole zagušenja je slična kao i kod TCP, ali je izvedba nešto drugačija.

Brojevi QUIC paketa se monotono povećavaju, bez obzira na to da li se u njima nalaze podaci koji se šalju prvi put ili ponovo. Okviri (*frames*) koji prenose *stream* podatke imaju informaciju o udaljenosti od početka (*offset*) *stream* na osnovu kojih se osigurava isporuka bajta unutar *stream* po redu. Ovim je pojednostavljen proces. Više nema potreba za utvrđivanjem da li je neki paket poslan prvi put ili ponovo, kao kod TCP. Brojevi paketa definišu poredak u vremenu, a *stream offset* poredak unutar *stream*.

Kontrola zagušenja u QUIC zasnovana je na TCP NewReno. ali je QUIC signalizacija, vezana za kontrolu zagušenja, dovoljno opšta da je moguće koristiti i druge algoritme. Cubic algoritam se često koristi. Takođe dvije strane u QUIC komunikaciji mogu koristiti različite algoritme kontrole zagušenja. Prethodno pomenuti način numerisanja paketa olakšava razlikovanje izgubljenih i paketa koji nisu isporučeni po redu. Time se može preciznije računati i RTT. Na taj način se poboljšava i kontrola zagušenja.

Migracija konekcija

TCP konekcija je definisana uređenom četvorkom: Izvorišna IP adresa, izvorišni port, odredišna IP adresa i odredišni port. Promjena nekog od ovih parametara prekida konekciju. Ova činjenica predstavlja smetnju u savremenim komunikacijama. Ako TCP klijent promijeni mrežu tako što tokom kretanja pređe sa bežične Wi-Fi mreže na 4G mrežu njegova IP adresa će se promijeniti i TCP konekcija će se prekinuti.

QUIC ima drugačiji pristup identifikaciji konekcija. Taj pristup omogućava da konekcije opstanu i u slučaju promjene adrese (IP i/ili porta) neke od strana u komunikaciji. Konekcije kod QUIC se identifikuju sa 64 bitnim konekcijskim ID. Ovaj ID omogućava da se konekcija migrira na drugu IP adresu i/ili port bez prekidanja.

Promjene porta mogu se desiti i kod statičnih krajnjih tačaka. NAT uređaji, pogotovo za UDP konekcije koje QUIC koristi, rade NAT *rebinding*, promjenu izlaznog NAT porta ili čak IP adrese za neki tok (konekciju).

Upotreba TLS i prolaz kroz *middlebox*-ove

Kod upotrebe TLS i TCP aplikacije koriste TLS za zaštitu povjerljivosti i integriteta paketa, te potvrđivanje autentičnosti, a TLS koristi TCP za transport. QUIC direktno aplikacijama pruža ove usluge. Usluga se ostvaruje pogodnim kombinovanjem TLS i QUIC. TLS se koristi za za dogovor oko ključeva i parametara, potvrđivanje identiteta i šifriranje. QUIC se koristi za pouzdanost, isporuku po redu i za slanje aplikativnih podataka. Već je rečeno da QUIC koristiti minimalno verziju 1.3 TLS.

Pored zaštite povjerljivosti integriteta poruka, njihovo šifriranje olakšava prolaz kroz *middlebox*-ove. QUIC se prenosi preko UDP. Na taj način nema potrebe da se *middlebox*-ovi mijenjaju da bi podržali novi transportni protokol. Dovoljno je da podržavaju UDP. Šifriranjem sadržaja QUIC paketa *middlebox*-ovi ne vide njihov sadržaj i nisu u prilici da pogrešno protumače neke opcije i na osnovu njih izmijene ili odbace paket.

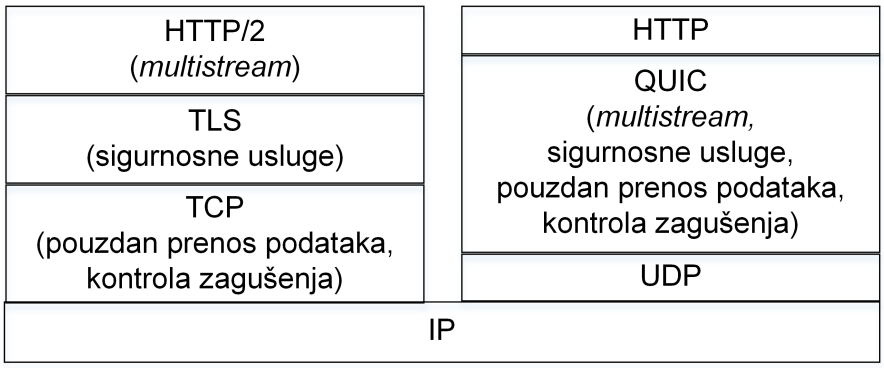
Postoji dodatna korist u tome što QUIC koristi UDP kao transportni protokol. Transportni protokol izvedeni su u operativnim sistemima. To čini njihove promjene težim i sporijim. QUIC nije takva vrsta transportnog protokola, već koristi UDP kao transportni protokol izveden u operativnom sistemu. Sam QUIC je, poput aplikativnih protokola, izveden u takozvanom korisničkom (*user*) prostoru. To omogućava njegovo brže i lakše mijenjanje i veći broj izvedbi različitih autora/proizvođača softvera.

Planirano je da QUIC bude transportni protokol za različite aplikativne protokole. Trenutni fokus je na podršci za HTTP. Nakon prihvatanja QUIC kao IETF RFC standarda posvetiće se više pažnje podršci drugih aplikativnih protokola.

Odnos skupa protokola HTTP/2, TLS i TCP u odnosu na HTTP, QUIC i UDP, te uloge svakog od protokola iz skupa prikazani su na slici 3.3.

3.5 HTTP/3

HTTP/3 je najnovija verzija HTTP protokola. Kao i QUIC, još uvijek je u fazi nacrta, ali na sigurnom putu da postane RFC standard. Razvoj HTTP/3



Slika 3.3: Položaj i uloga QUIC u slojevima protokola

i QUIC ide paralelno pa je i verzija nacrta HTTP/3 u vrijeme pisanja ista, 27 [33]. Osnovna razlika u odnosu na HTTP/2 je prelazak sa TCP transportnog protokola na QUIC po UDP. U prvim verzijama, 2016.godine, naziv protokola bio je "HTTP po QUIC".

Ovo je prirodan nastavak procesa izmjena HTTP sa, prvenstvenim, ciljem ubrzavanja rada. HTTP/3 je preuzeo većinu osobina koje su ugrađene u HTTP/2. Zadržana je kompatibilnost unazad za aplikacije. Semantika i osnovni koncepti, poput HTTP metoda, polja zaglavlja, statusnih kodova i URI-a, nisu se promijenili. Kao i HTTP/2 koristi binarni format, multipleksiranje zahtjeva i odgovora upotrebom *stream*-ova koji se mogu prioritetizirati, te omogućava serverski *push*. Kako QUIC ima podršku za *stream*-ove integracija je mnogo bolja.

Ono što je različito je kompresija zaglavlja. Kompresija zaglavlja kod HTTP/2 zasnovana je, dijelom, na smanjenju potrebe za slanjem polja zaglavlja koja se nisu promijenila. QUIC osigurava isporuku bajta po redu unutar pojedinih *stream*-ova, ali ne garantuje redoslijed isporuke među *stream*-ovima. Sada pošiljalac ne može biti siguran u redoslijed pristizanja zaglavlja, pa se ne može oslanjati na referenciranje ranije poslanih polja zaglavlja. Algoritam kompresije HPACK koji je koristio HTTP/2 zamijenjen je u HTTP/3 sa novim koji se zove QPACK. QPACK koristi posebne jednosmjerne *stream*-ove za održavanje tabele stanja zaglavlja. Na taj način se zadržava mogućnost kompresija zaglavlja, ali prilagođena prenosu po QUIC.

Postoje i još neke sitnije razlike u odnosu na HTTP/2 kod rada sa *stram*-ovima vezane za kontrolu toka i prioritetizaciju i prebacivanje dijela odgovornosti na QUIC. Ove razlike nisu presudne, a njihovi detalji se još mogu promijeniti do konačne verzije. Iz tog razloga nisu detaljnije obrađene.

3.5.1 Otvorena pitanja

Prelazak na UDP je velika promjena. Internet infrastruktura je trenutno optimizirana za TCP. Veliki broj *firewall* dopušta UDP saobraćaj samo po portu 53 koji se koristi za DNS. Biće potrebno dopustiti i UDP port 443, podrazumijevan za HTTP/3, odnosno QUIC, ili drugi UDP port koji HTTP/3 server bude koristio. UDP bazirani aplikativni protokoli i serveri koji ih implementiraju znali su imati poteškoća sa amplifikacijskim napadima. Napad je zasnovan na nesrazmjeri između poruka od klijenta, koje mogu biti jako male, i odgovora od servera, koji može bit mnogo veći, ali usmjeren ka drugom klijentu. Na ovaj način napadač može višestruko povećati saobraćaj ka žrtvi i time, lakše, izvršiti napad uskraćivanje usluge (DoS). Da bi se ovo spriječilo QUIC ima ograničenje na minimalnu veličinu inicijalnog paketa od 1200 bajta. To je dovoljno veliko da onemogući bitno povećavanje odgovora. Zbog prilagođavanja operativnih sistema TCP, UDP radi nešto sporije i koristi više CPU nego TCP. To će se vjerovatno izmijeniti u novim verzijama OS koje budu ugrađivale podršku za QUIC.

Da bi protokol ušao u upotrebu potrebno je da postoji podrška klijenata i servera. HTTP/3 je, u novembru 2019., podržan u probnim verzijama Google Chrome i Mozilla Firefox, te u nekoliko verzija web servera, uključujući i *nginx* putem dodatnih biblioteka i CloudFlare *quiche* server. Ubrzavanja koja HTTP/3 donosi vjerovatno će stimulisati njegovo brže uvođenje, pogotovo na lokacijama sa više prometa. Takođe je potrebno da klijent i server saznaju da druga strana podržava HTTP/3. To se danas, i vjerovatno do dominacije HTTP/3, rješava tako što konekcija počinje kao HTTP/2 preko TCP, ali server prilikom prvog odgovora dostavlja polje zaglavlja `Alt-Svc` sa vrijednošću `h3=":broj_udp_porta"`. Nakon toga klijent, ako podržava HTTP/3, uspostavlja HTTP/3 i QUIC konekciju na navedeni port.

HTTP/3 više ne podržava slanje nešifriranih poruka. Zapravo QUIC radi koristeći TLS. Koristiće se samo šema HTTPS://, a ne HTTP://. To je sasvim u skladu sa tendencije zaštite kompletnog web saobraćaja. Ipak to predstavlja poteškoću za web lokacije koje moraju instalirati certifikate, te otežava nadzor i kontrolu saobraćaja, što ima i dobre i loše strane.

* * *

Ovim je završen pregled intenzivnih izmjena koje se dešavaju sa HTTP i TLS protokolima posljednjih godina. Ove promjene pokazuju potrebe za izmjene tradicionalnih protokola, ali i njihovu prilagodljivost, kao i prilagodljivost cjelokupne Internet infrastrukture.

4

IoT umrežavanje

Internet of Things (IoT) predstavlja očiglednu težnju uvezivanja svega u mrežu koja omogućava razmjenu informacija. Stvari (*thigs*) u IoT uključuju sve fizičke stvari[1]. Internet u IoT predstavlja globalnu mrežu i ukazuje na mogućnost globalnog uvezivanja stvari. Pored toga riječ Internet u IoT odnosi se i na tehnologije uvezivanja koje se koriste za povezivanje na sadašnjem Internetu.

Kako je ovo knjiga iz oblasti računarskih mreža IoT će biti sagledan iz ugla uvezivanja, odnosno izazova koje uvezivanje stvari donosi. Stari Internet, koji ne uključuje stvari, uvezivao je prvo klasične računare. Ti računari su bili serveri i personalni računari, kako fiksni (*desktop*) tako i mobilni (*laptop*). Kasnije su uključeni i drugi uređaji poput mobilnih telefona, tableta i sličnih. Zajednička karakteristika svih ovih uređaja je da su zasnovani na računarskoj arhitekturi koja uključuje centralnu procesorsku jedinicu, radnu i trajnu memoriju, ekran i dio koji omogućava slanje i prijem podataka. IoT uključuje sve ove uređaje, ali i nove stvari koje su nešto drugačije. Osnovna razlika između ovih uređaja i proširenog skupa stvari iz IoT je u resursima dostupnim za obradu, pohranu i razmjenu podataka. Nove stvari su mnogo ograničenije po svakom od ovih resursa. Pored toga nove stvari mogu biti ograničene i sa količinom energije za rad, jer često nemaju stalno napajanje ili imaju baterije koje su ili malog kapaciteta ili treba da traju godinama. Ova ograničenja postavljaju nove zahtjeve na uvezivanje ovih uređaja sa ostatkom Interneta. Na osnovu ovih zahtjeva razvijeni su novi pristupi i protokoli za uvezivanje IoT. Ti protokoli i njihovo uklapanje sa postojećim preovladavajućim Internet protokolima su tema ovog poglavlja.

[1] Pojam *Internet of Everything* je širi, jer pored fizičkih stvari uključuje i ljude, procese i podatke.

4.1 IoT elementi i funkcionisanje

Kako je već rečeno, osnovi element IoT sistema su *stvari* koje imaju mogućnost razmjene podataka. Nema teoretske prepreke da se ove stvari međusobno povezuju i čine IoT sistem. Ako su stvari savremeni računari ili mobilni uređaji onda je računarska mreža koju uvezivanjem tvore IoT sistem. Međutim, to nije poenta IoT. IoT je orijentisan na interakciju računarskog svijeta sa fizičkim. Ove stvari koje imaju interakciju sa fizičkim svijetom su uglavnom obavljale jednostavne funkcije i imale su ograničene mogućnosti pohrane, obrade i razmjene informacija. Kao primjeri ovakvih stvari se najčešće navode senzori i aktuatori. Senzori prikupljaju informacije o nekoj fizičkoj veličini iz svoje okoline. Aktuatori utiču na neku fizičku veličinu u svojoj okolini. IoT senzori šalju prikupljene informaciju u IoT sistem. IoT aktuatori djeluju na osnovu naredbi koje dobiju iz IoT sistema. Direktno povezivanje ovakvih uređaja nije korisno. Oni nisu u mogućnosti obraditi informacije i na osnovu obrade poduzeti akcije. I tradicionalni sistemi upravljanja, koji se sastoje od senzora i aktuatora, imaju dio sistema koji se bavi prikupljanjem podataka, njihovom obradom i generisanjem komandi. Očigledno je da i IoT sistemi moraju imati komponentu sa istom namjenom.

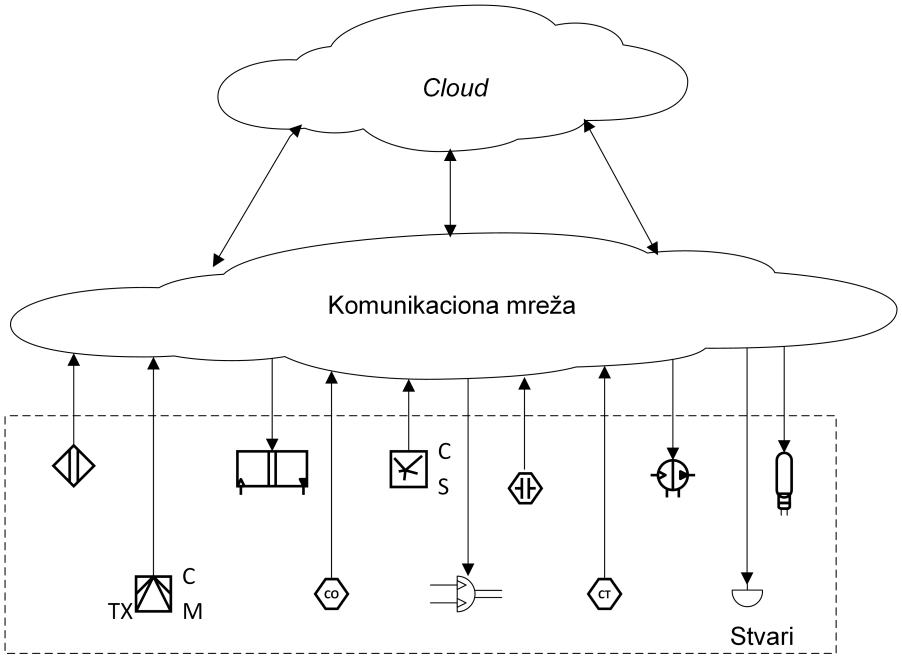
Nadalje, IoT sistemi bi trebali predstavljati napredak u odnosu na tradicionalne sistema. Jedan od aspekata tog napretka su savremene metode obrade podataka. Ove metode omogućavaju ostvarenje tradicionalne uloge koja podrazumijeva da se organizovani i smisaoni podaci predstavljaju informacije, čijom se obradom može steći znanje. Danas popularna metoda za ovu namjenu je mašinsko učenje. Pored ovoga, količina podataka koju veći IoT sistemi prikupljaju, svrstava ih u takozvane *big data* sisteme, koji zahtijevaju posebnu obradu podataka. Ta količina podataka omogućava i rudarenje podataka (*data mining*). Ukratko prikupljanje i obrada podataka u IoT sistemima otvara mnogo mogućnosti. Te mogućnosti nisu tema ovog poglavlja i o njima se više može pročitati u knjigama posvećenim IoT [30][28][24]. Ono što je ovdje bitan zaključak je da je za ostvarivanje planirane uloge IoT neophodno da postoji jedan ili više elemenata čija je uloga prikupljanje i obrada podataka.

Elementi IoT sistema sa ovim zadatkom mogu biti računari sa dovoljno kapaciteta za potrebno prikupljanje i obradu. Ovi računari se mogu nalaziti blizu IoT stvari, ali i negdje na udaljenoj lokaciji sa kojom se povezuje preko globalnog Interneta. U opštem slučaju može se smatrati da se nalaze na proizvoljnoj Internetskoj lokaciji. U skladu sa savremenim trendom pružanja usluge obrade informacija putem *cloud*, ova komponenta IoT sistema se uglavnom posmatra kao da se nalazi u *cloud*.

Stvari i *cloud* sa kojim se razmjenjuju informacije potrebno je povezati Povezuju se putem mreže koja može biti tradicionalna računarska mreža koja se koristi za povezivanje korisničkih uređaja, poput personalnih računara i mobilnih uređaja, sa Internet serverima. Međutim veoma često je takva mreža neadekvatna za IoT potrebe. Tim pitanjem će se baviti ostatak poglavlja,

nakon što se nešto detaljnije prikažu stvarni elementi IoT sistema i njihova povezanost.

Osnovni elementi IoT: stvari, mreža i *cloud* i njihova povezanost prikazani su na slici 4.1.



Slika 4.1: IoT - Osnovni elementi i veze

Iako slika 4.1 prikazuje osnovne funkcionalne elemente IoT sistema, praktične izvedbe obično imaju i neke dodatne (među)elemente. Dva su osnovna razloga za to. Jedan je mogućnost uvezivanja, a drugi potreba za agregacijom podataka.

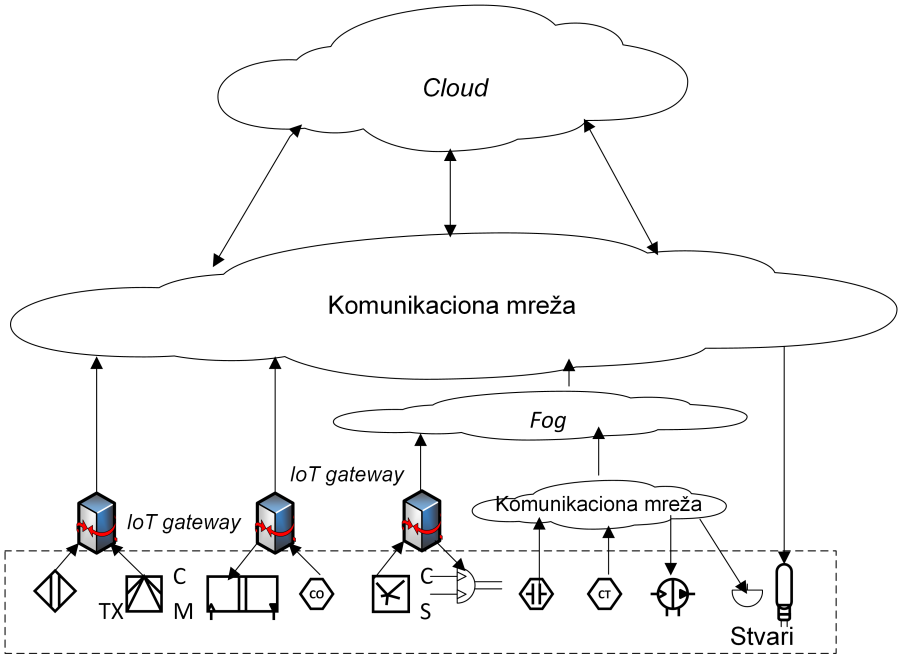
Jednostavne stvari, poput senzora, često nemaju resurse koji bi im omogućili da svoje podatke dostavljaju do *cloud*. Za direktno povezivanje sa *cloud* potreban je puni TCP/IP skup protokola. Senzori i aktuatori mogu koristiti komunikacione protokole koji su ili naslijeđeni ili prilagođeni njihovim, ograničenim, resursima. Ovi protokoli mogu biti nekompatibilni sa standardnim TCP/IP skupom protokola. U tom slučaju potreban je uređaj koji omogućava povezivanje stvari sa mrežnom infrastrukturom putem koje se može povezati sa *cloud*. Takav uređaj se uglavnom naziva *gateway*. Njegova uloga je da omogući prosljeđivanje podataka koji su pristigli po jednom interfejsu po protokolu koji taj interfejs podržava na drugi interfejs po drugom protokolu. IoT *gateway*

najčešće omogućava uvezivanje stvari koje podržavaju različite, njima odgovarajuće, protokole na standardnu računarsku mrežu i preko nje na globalni Internet. Ovi posebni protokoli koje IoT stvari podržavaju će biti posebno razmatrani. Jedan IoT *gateway* uglavnom povezuje više IoT stvari. U IoT sistemu može postojati više IoT *gateway* za više različitih protokola koje različite IoT stvari podržavaju. Neke izvedbe IoT *gateway* imaju podršku za više različitih protokola pa mogu povezivati više grupa različitih uređaja. Važno pitanje koje se ovdje javlja je propusnost IoT *gateway*. On ne smije biti usko grlo te treba voditi računa o broju uređaja koji se na njega povezuju. To pitanje je dobar uvod u drugi razlog za uvođenje dodatnih IoT (među)elemenata, a to je agregacija.

IoT stvari mogu slati sve podatke koje prikupljaju u *cloud*. Sa porastom broja stvari raste i količina podataka koja se dostavlja. Ovo može predstavljati opterećenje na računarsku mrežu koja povezuje IoT stvari sa *cloud* kao i za *cloud* infrastrukturu koja obrađuje te podatke. Sa druge strane, upitno je da li su svi ti podaci potrebni i korisni u *cloud*. Agregacija podataka pojavljuje se kao moguće rješenje. Obrada izvornih podataka koje šalju IoT stvari može se obavljati negdje bliže tim stvarima. Rezultat te obrade su agregirani podaci, na način i po kriteriju koji odgovara konkretnoj namjeni za koju se IoT sistem koristi. Prema *cloud* se šalju agregirani podaci čime se štede mrežni i *cloud* resursi. Ova obrada može biti i složenija i može uključivati analizu i mašinsko učenje, a ne samo agregaciju podataka. Jedno logično mjesto za ovu agregaciju je IoT *gateway*. To je uređaj na kom se svakako prikupljaju podaci, pa bi se tu mogla uraditi i neka inicijalna obrada. Naravno ovo podrazumijeva da IoT *gateway* mora imati dovoljno resursa za planiranu obradu podataka, a ne samo za prosljeđivanje. Ovakav pristup agregaciji, gdje se ona obavlja vrlo blizu izvora IoT podataka naziva se *edge computing* [22]. Nešto širi pojam koji obuhvata različita mjesta i načine obrade IoT podataka prije njihovog slanja na *cloud* naziva se *fog computing* [12]. Obrada IoT podataka bliže njihovom izvoru donosi još neke prednosti. Podaci se obrađuju sa manjim kašnjenjem u odnosu na obradu u *cloud*. Uređaji koji obrađuju podatke mogu biti svjesni lokacije izvora. Uređaji mogu biti prilagođeni vrsti i formatu podataka koji se obrađuju, pa time efikasniji. Savremeni IoT sistemi uključuju sve ove elemente, kako je prikazano na slici 4.2.

4.2 Posebnosti IoT umrežavanja

Umrežavanjem na tradicionalnom Internetu dominira ograničeni skup protokola: HTTP(S)/TCP/IP/Ethernet-WiFi. Za žičano uvezivanje na podatkovnom, i fizičkom, sloju Ethernet je izašao kao jasni pobjednik, nad svim drugim proteklih koji su se pojavili kao alternativa. Relativna jednostavnost Ethernet i dovoljno dobro podržavanja prethodnih verzija omogućili su Ethernet-u da zauzme dominantnu poziciju na tržištu i zadrži je sve do danas. WiFi, odnosno



Slika 4.2: IoT - Prošireni skup elemenata i veze

802.11, protokol ima sličan položaj u bežičnom lokalnom povezivanju uređaja u računarsku mrežu, kao i Ethernet. IP je praktično jedini mrežni protokol koji se koristi na Internetu, i to pretežno verzija IPv4. TCP je uzeo primat među transportnim protokolima.[2] A HTTP dominira na aplikativnom sloju.

Ovaj dominantni skup protokola podrazumijeva čvorove/uređaje koji ga podržavaju. Za jednostavne stvari iz IoT, koje imaju ograničene resurse za obradu i pohranu podataka i ograničenu energiju za rad, to može predstavljati poteškoću. Pored ovoga, koncepti povezivanja na savremenom Internetu, koji uglavnom podrazumijevaju *unicast* saobraćaj između dva krajnja čvora, nisu uvijek pogodni za IoT sisteme. U nastavku su objašnjene posebnosti IoT sistema u odnosu na standardne računarske mreže, koje stvaraju potrebu za novim pristupima umrežavanju i novim protokolima.

4.2.1 Potrošnja energije

Jednostavne stvari iz IoT se često napajaju iz baterija, jer na lokaciji na kojoj se nalaze nema napajanje iz električne mreže. Ove stvari mogu biti i na teško dostupnim mjestima, pa je planirano da rade duži vremenski period, koji se

[2] U poglavlju o HTTP pokazano je da se dešavaju promjene po ovom pitanju, što i jeste poenta drugog dijela knjige.

mjeri godinama, bez održavanja. Za ovakve uređaje je vrlo bitno koliko energije troše za slanje i prijem podataka. Standardni protokoli iz računarskih mreža nisu optimizirani po ovom kriteriju. Za ove uređaje su pogodniji drugi protokoli koji su pravljeni sa ciljem da troše što manje energije. Bitno je napomenuti da je za jednostavne uređaje često najveći potrošač energije komunikacija.

4.2.2 Procesorska moć

Izvedba standardnih mrežnih protokola, a posebno sigurnosnih, podrazumijeva određenu mogućnost procesiranja uređaja koji ih podržavaju. Ta mogućnost procesiranja može biti mnogo veća od one koju jednostavne IoT stvari, poput senzora i aktuatora, imaju. Ovakvi uređaji imaju resurse za obradu podataka (procesor, radnu i trajnu memoriju) optimizirane za njihovu osnovnu namjenu. Trošenje tih resursa za komunikaciju može ići nauštrb njihove osnovne funkcionalnosti. Povećavanje ukupnih procesorskih resursa ovih uređaja podiže njihovu cijenu, povećava potrošnju energije, a moguće i njihovu veličinu što može biti neprihvatljivo za praktične izvedbe. Potrebni su jednostavniji komunikacioni protokoli koje ovi uređaji mogu podržati.

IETF RFC 7228 [13] definiše tri klase uređaja 0, 1 i 2. Podjela je napravljena po količini radne memorije za obradu podataka (RAM) i trajne memorije za pohranu programskog koda koji se izvršava (*flash*). Klasa 0 su uređaji sa manje od 1 KB RAM i manje od 100 KB *flash* memorije. To su uglavnom jednostavni senzori koji mogu komunicirati samo preko posrednika. Klasu 2 čine uređaji koji mogu podržati većinu Internet protokola, ali im je korisno da to budu što manje zahtjevne verzije. Ovi uređaji imaju više od 50 KB RAM i više od 250 KB *flash* memorije. U klasi 1 su uređaji sa mogućnostima između klasa 0 i 2. Oni imaju resurse za komunikaciju, ali nedovoljne za izvedbu standardnih Internet protokola.

4.2.3 Broj uređaja

U IoT sistemima uglavnom postoji veliki broj krajnjih uređaja koji šalju podatke na isto odredište. Ovaj broj je bar za jedan red veličine veći nego kod savremenih računarskih mreža. Takvo povećanje broja uređaja otvara dva bitna pitanja. Jedno je vezano za tip konekcije, a drugo za adresiranje.

Tip konekcije

Dominantan način dostave podataka u tradicionalnim računarskim mrežama je jedan na jedan, između dvije tačke. Bilo da je jedan pošiljalac, a jedan primalac, bilo da se radi o situaciji gdje obje strane u komunikaciji šalju podatke. U IoT sistemima može postojati veliki broj senzora čiji podaci treba da dospiju do jednog ili više primalaca. Veći broj aktuatora može primati

naredbe od jednog ili više čvorova. Ako se za svako slanje ostvaruje posebna, jedan na jedan, konekcija to može stvarati više mrežnog saobraćaja nego što je neophodno. U ovakvim sistemima pogodno bi bilo imati mogućnost različitih načina dostave podataka, poput više na jedan, jedan na više ili više na više.

Adresiranje

Uobičajeni način adresiranja čvorova u mreži je korištenjem IP adresa. IPv4 adrese su uglavnom potrošene i prije dolaska IoT. NAT koji se, godinama, koristi kao "privremeno" rješenje ima svojih nedostataka koji posebno dolaze do izražaja kod IoT uređaja koji možda trebaju dvosmjernu komunikaciju, identifikaciju pojedinih uređaja i/ili su mobilni.

4.2.4 Brzina prenosa podataka

Različiti IoT sistemi imaju različite potrebe za brzinu prenosa podataka, količinu podataka u jedinici vremena. Savremeni Internet protokoli podržavaju velike brzine prenosa koje ne moraju biti potrebne svim IoT sistemima. Protokoli koji omogućavaju prenos manjim brzinama mogu biti dostatni. Ovakvi protokoli obično imaju i manje zahtjeva na energiju i procesorsku moć.

4.2.5 Domet

Udaljenosti između uređaja u IoT sistemima mogu biti različite. Te udaljenosti, u nekim slučajevima, mogu biti jako male ili jako velike u odnosu na uobičajene udaljenosti između uređaja u računarskim mrežama, koje su za lokalne mreže obično desetine metara. Za vrlo male udaljenosti postoje odgovarajući protokoli, kao i za one vrlo velike, pa je potrebno izabrati najpogodnije za namjenu.

4.2.6 Noseće frekvencije

Frekvencija koju uređaj koristi za prenos informacija utiče na domet i brzinu prenosa. Frekvencija koja se može koristiti zavisi od ograničenja lokacije na kojoj se uređaji nalaze. Upotreba radio-frekventnog spektra je regulisana od strane država i međunarodnih organizacija. Za upotrebu većine frekvencija potrebne su licence koje se uglavnom plaćaju. Međunarodna telekomunikacijska unija (ITU) je definisala dio spektra za čiju upotrebu nisu potrebne licence. Taj dio spektra naziva se ISM zbog namjene za industrijske, naučne i medicinske primjene. Iako se ISM spektar može koristiti bez licence postoje državne i regionalne regulative koje definišu ograničenja na jačinu signala, širinu kanala, procenat korištenja (*duty cycle*) i još neke parametre. Pošto za ISM spektar nema licenci onda nema ni garancija za dostupnost ili zaštitu komunikacija na ovim frekvencijama.

4.2.7 Topologija

Način povezivanja IoT uređaja koji razmjenjuju informacije može biti različit od uobičajenog u računarskim mrežama. Preovladavajući klijent/server model u suštini pravi konekciju između dva čvora, uglavnom preko drugih komunikacionih čvorova, radi razmjene informacija. Uređaji u IoT sistemu sa slike 4.2 mogu imati jednu ili više različitih uloga, poput klijenta, servera, posrednika. Zato oni mogu biti povezani različitim topologijama.

4.3 Potreba za IoT specifičnim protokolima

Posebnosti IoT umrežavanja dovele su do upotrebe protokola izvan spomenutog dominantnog skupa protokola HTTP(S)/TCP/IP/Ethernet-WiFi. Pošto je ova oblast još uvijek u razvoju, ne postoji standardni skup IoT protokola, već postoji više različitih protokola koji obavljaju slične funkcije. U budućnosti će vjerovatno neki od protokola postati dominantni, ali do tada je potrebno imati uvid u veći broj. Od postojećih protokola u nastavku će biti predstavljeni oni koji se najviše koriste ili, po mišljenju autora, imaju najsvjetliju budućnost. Predstavljanje protokola, kako je to uobičajeno u literaturi iz računarskih mreža, urađeno je po mrežnim slojevima. Neki od navedenih protokola ne pripadaju samo jednom sloju. Takvi protokoli su prikazani u okviru mrežnog sloja na kom je njihova uloga bitna za zadovoljavanje potreba IoT sistema.

4.4 Podatkovni i fizički sloj

Protokoli na ovom sloju rješavaju pitanje povezivanja IoT stvari sa ostatkom IoT sistema. Neke od tehnologija koje se koriste definišu i slojeve iznad podatkovnog. Ako ti slojevi ne uključuju IP, kao jedini protokol mrežnog sloja koji se koristi na Internetu, onda je za povezivanje stvari sa globalnom mrežom potreban *gateway*-om koji onda prosljeđuje podatke preko mreže u *cloud*. To je jedna vrsta podjele komunikacionih protokola za IoT. Druga podjela je po dometu. Po ovoj podjeli postoje WPAN (Wireless Personal Area Networks), WLAN (Wireless LAN) i WAN (Wide Area Network) tehnologije.

Najčešće korištene WPAN tehnologije su one zasnovane na standardima IEEE 802.15 radne grupe za WPAN, poput 802.15.1 Bluetooth i 802.15.4, na kom je zasnovan Zigbee. Postoje i druge WPAN tehnologije koje su u potpunosti vlasničke poput Z-wave. U nastavku su predstavljene neke od ovih tehnologija.

4.4.1 Bluetooth

Bluetooth je nastao sa ciljem da zamijeni RS-232 kablove za povezivanje perifernih uređaja sa računarima. Kasnije se ideja proširila i na povezivanje

periferala mobilnih uređaja. IEEE je 2002. standardizovao Bluetooth verziju 1.1 standardom 802.15.1-2002. Standardom 802.15.1-2005. pokrivena je Bluetooth verzija 1.2. Verzije Bluetooth nakon toga više ne standardizuje IEEE. Standardom upravlja Bluetooth SIG (*Special Interest Group*) sa preko 35.000 članova od kojih su najaktivniji, i najuticajniji, veliki proizvođači hardvera i softvera. Aktuelna verzija Bluetooth u vrijeme pisanja je 5.2 [3] koja je objavljena 31.12.2019. godine. U tom dokumentu može se naći više detalja o Bluetooth.

Bluetooth nije namijenjen prvenstveno i isključivo za IoT. Danas se koristi za razne namjene. Jedna od vrlo popularnih je slanje zvuka između različitih uređaja poput mobilnih telefona ili televizora i zvučnika/mikrofona. Ova namjena je slanje velike količine podataka u realnom vremenu što je vrlo različito od potreba IoT. U verziji 4.0 Bluetooth je uveo način rada sa malom potrošnjom energije, BLE (Bluetooth *Low Energy*). To je učinilo da ova tehnologija postane mnogo interesantnija za upotrebu u IoT sistemima. BLE troši od 2 do 100 puta manje energije od drugog načina rada koji se naziva BR/EDR (Bluetooth *Basic Rate/Enhanced Data Rate*). Konkretan iznos potrošnje zavisi od hardvera i načina upotrebe. Način upotrebe zavisi od aplikacije i odnosi se na to koliko je uređaj aktivan, a koliko nije (*sleep mode*) i kojom snagom šalje. BLE smanjuje potrošnju energije, prije svega, tako što uređaji većinu vremena nisu aktivni i aktiviraju se samo kad je potrebno. Kada se desi događaj zbog kog se BLE uređaj budi, šalje se kratka poruka centralnom uređaju, koji uglavnom ima bolje napajanje. To je sasvim pogodno za IoT uređaje. Očekuje se da uređaji koji koriste BLE mogu raditi više godina sa baterijama veličine dugmeta.

Manja potrošnja energije kod BLE plaćena je nešto nižim brzinama prenosa koje se kreću od 125 kb/s do 2 Mb/s. "Klasični" Bluetooth postiže brzine od 3 Mb/s. Domet BLE je sličan kao i klasičnog Bluetooth i od verzije 5.0 iznosi stotine metara. BLE koristi 40 kanala širine 2 MHz. Za efikasnu upotrebu nelicenciranog spektra izbor kanala se vrši upotrebom FHSS (*Frequency-Hopping Spread Spectrum*).

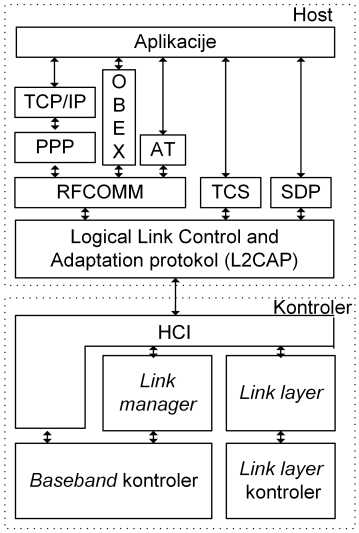
Kod klasičnog (BR/EDT) Bluetooth formiraju se mreže koje se sastoje od jednog *master* i do sedam *slave* čvorova. Razlog za ovo ograničenje su tro-bitne adrese koje se koriste. Ova mreža se naziva *piconet*. Svi čvorovi u ovakvom *piconet* u jednom trenutku koriste isti kanal za komunikaciju. Mreža može biti proširena povezivanjem sa drugim *piconet*. Jedan čvor nalazi se u dva *piconet* i omogućava prosljeđivanje između njih. Taj čvor može biti *master* u jednom, a *slave* u drugom *piconet* ili *slave* o oba.

BLE koristi 24 bitne adrese, pa jedan *master* može biti povezan sa preko 16 miliona *slave* uređaja. *Master* sa svakim *slave* pravi poseban *piconet* koji radi na različitom kanalu. Uređaji pored *master* i *slave* mogu biti i u *standby* stanju. Zapravo većina BLE uređaja većinu vremena provodi u ovom stanju. Ove osobine su pogodne za IoT sisteme jer podržavaju povezivanje velikog broj uređaja, od kojih je samo manji broj aktivan u nekom trenutku.

Bluetooth imaju jedinstvene 48-bitne adrese, koje su poput Ethernet MAC adresa. Prva 24 bita su dodijeljena proizvođaču, a preostala 24 generiše proizvođač. Svaki uređaj istog proizvođača treba imati različitu adresu. Pored ovih hardverskih adresa BLE može koristiti i slučajne adrese, što se koristi za povećavanje sigurnosti. Postoje tri načina generisanja ovakvih adresa: slučajna statična, privatna razlučiva (*resolvable*) i privatna nerazlučiva. Slučajna statična adresa je ili uprogramirana u uređaj ili se mijenja sa svakim paljenjem (priključenjem na napajanje) uređaja. Privatne nerazlučive adrese se mijenjaju za svaku konekciju. Privatne razlučive adrese se generišu slučajno ali zavise od IRK (*Identity Resolving Key*). To je vrijednost koju uređaji dogovaraju prilikom povezivanja (uparivanja).

Bluetooth podržava tri različite topologije. Jedna je povezivanje dva čvora (*point to point*), druga je povezivanje jednog čvora sa svima (*broadcast*) i treća je povezivanje svih čvorova (*mash*). Različite topologije su pogodne za različite namjene. BLE podržava sve tri.

Bluetooth ne definiše samo podatkovni sloj nego protokole za sve slojeve. Ovi slojevi nisu direktno povezani sa ISO OSI slojevima. Skup (*stack*) Bluetooth protokola podijeljen je na tri sloja: *host*, kontroler i radio hardver. Radio hardver brine se za prijem i slanje signala. Kontroler je izvedba donjeg dijela podatkovnog sloja i brine se za kontrolu pristupa mediju i grešaka. *Host* sloj pruža usluge aplikacijama i obavlja ulogu transportnog i mrežnog sloja, ako i gornjeg dijela podatkovnog sloja. Ovaj sloj ima više različitih protokola. *Host* i kontroler komuniciraju preko HCI (*Host Controler Interface*). Na slici 4.3 su prikazani obavezni protokoli kod Bluetooth.



Slika 4.3: Bluetooth skup protokola

Za praktičnu upotrebu je bitna činjenica da je Bluetooth podrška ugrađena u većinu savremenih mobilnih telefona, tako da oni, u nekim scenarijima upotrebe, mogu igrati ulogu *gateway* bez potrebe za nabavku i konfiguraciju posebnog uređaja.

Bluetooth je opisan sa aspekta koji su navedeni da su posebni za IoT umrežavanje. Više detalja o protokolu može se pored zvanične specifikacije naći i u novijoj knjizi posvećenoj BLE [5]

4.4.2 IEEE 802.15.4

IEEE 802.15.4 je IEEE standard [1] za uređaje vrlo male potrošnje koji imaju nisku cijenu i šalju podatke malim brzinama. Ovakve mreže se nazivaju i LR-WPAN (*Low-Rate Wireless Personal Area Networks*). Standard definiše fizički sloj i MAC (*Media Access Control*) podsloj podatkovnog sloja. On predstavlja osnovu za više protokola koji onda definišu slojeve iznad. Neki od tih protokola su Zigbee, 6LoWPAN, Wireless HART, ISA100.11a, Thread preko 6LoWPAN. Neki od njih su dodatno opisani nakon opisa 802.15.4 protokola.

IEEE 802.15.4 omogućava komunikaciju na udaljenostima do 10 metara[3] i po brzinama do 250 kb/s. Zbog svojih osobina uglavnom se koriti u automatizaciji domaćinstava i zgrada, u mrežama u vozilima i industriji, u igračkama i nekim mjeračima potrošnje. Većina ovih sistema se može smatrati IoT sistemima.

Bitna karakteristika 802.15.4 za IoT je da nudi mogućnost smanjivanja potrošnje izborom manje brzine prenosa korištenjem nekog od više mogućih izvedbi fizičkog sloja. Protokol je fleksibilan i podržava uređaje koji mogu biti vrlo jednostavni, a time imaju i manje troškove proizvodnje i održavanja.

LR-WPAN mreže podržavaju komunikacione sisteme u kojim se paketi obično šalju povremeno u kratkim intervalima na manju udaljenost. Iz tog razloga je veličina okvira u ovakvim mrežama mala. Slanje ovakvih okvira kraće traje i smanjuju se šanse za smetnje u radio spektru, s obzirom da se koristi slobodni dio spektra.

Osnovna komponenta 802.15.4 mreže je uređaj. Dva ili više uređaja koji komuniciraju na istom komunikacionom kanalu čine jedan WPAN. Pogodnost 802.15.4 za IoT ogleda se i u tipovima uređaja koje podržava: uređaji ograničene (RFD - *Reduced Function Device*) i pune (FFD - *Full Function Device*) funkcionalnosti. Uređaji ograničene funkcionalnosti obavljaju jednostavne funkcije i ne šalju velike količine podataka. Shodno tome, ovakvi uređaji mogu biti napravljeni sa vrlo malo resursa i raditi sa malom potrošnjom energije. U WPAN mreži oni mogu biti samo krajnji čvorovi koji šalju ili primaju podatke i ne mogu raditi prosljeđivanje i/ili rutiranje za druge čvorove. To upravo odgovara jednostavnim IoT senzorima, poput mjerača temperature, ili aktuatorima, poput prekidača za svjetlo. koji mogu biti napajani iz baterije dugi vremenski period bez potrebe za zamjenom. Ovakvi uređaji se ne mogu

[3] U praksi se ostvaruju i nekoliko puta veće udaljenosti od 10 metara

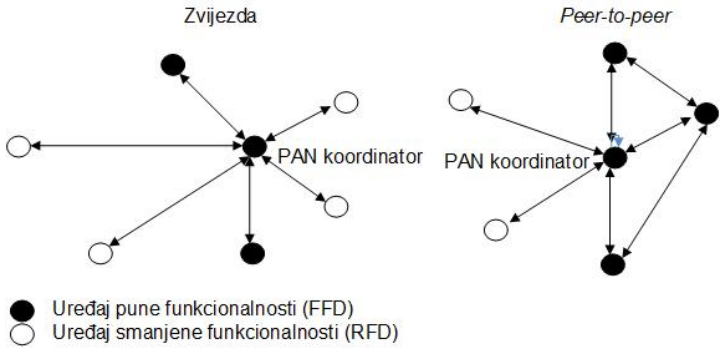
povezivati bez posredstva uređaja pune funkcionalnosti, koji onda, između ostalog, igraju ulogu *gateway*. Uređaji pune funkcionalnosti mogu imati sve funkcionalnosti koje ograničeni uređaji nemaju. Oni mogu raditi rutiranje i/ili prosljeđivanje saobraćaja u WPAN. Iz tog razloga oni se koriste za povezivanje RFD i imaju ulogu koordinatora ili PAN koordinatora,

Jedan WPAN sastoji se od bar dva uređaja koji komuniciraju na istom fizičkom kanalu. U svakom WPAN mora postojati PAN koordinator. Tu ulogu, kako je rečeno, može imati samo FFD.

U skladu sa duhom podržavanja fleksibilnosti 802.15.4 može raditi u jednoj od dvije topologije: zvijezda (*star*) ili *peer-to-peer*. U topologiji zvijezde svi uređaji povezani su direktno sa PAN koordinatorom. Sva komunikacija odvija se preko PAN koordinatora. Očigledno je, iz ove topologije, zašto PAN koordinator mora biti FFD i uglavnom imati slano napajanje. Uloga koju preuzima PAN koordinator omogućava da ostali uređaji moraju manje raditi, slati i primati manje podataka i trošiti manje energije, pa mogu biti RFD, koji su jednostavniji i niže cijene proizvodnje i nižih troškova rada. Ova topologija pogodna je za IoT aplikacije koje su po svojoj prirodi organizovane sa jednim centralnim i više perifernih čvorova poput računara i periferala, automatizacije kuće ili nadzora zdravstvenih parametara osoba.

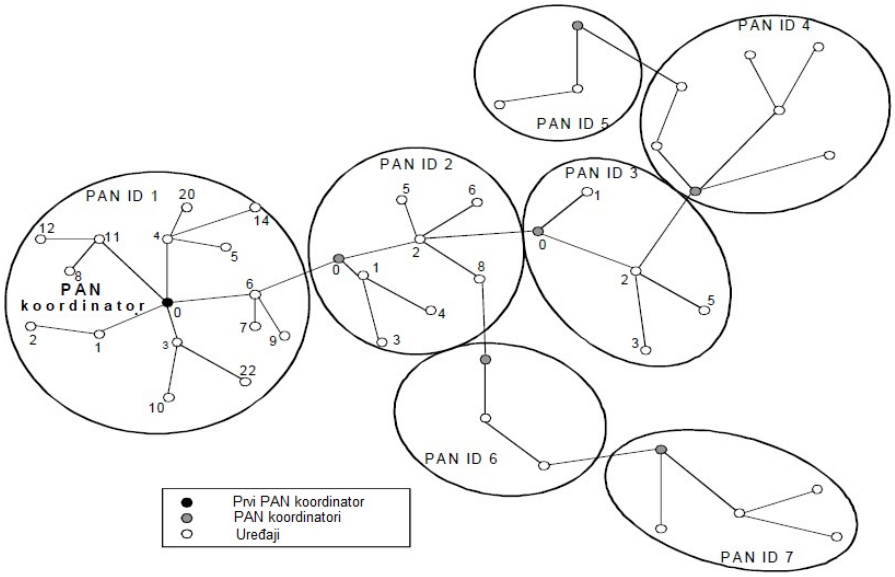
U *peer-to-peer* topologiji, svaki uređaj može komunicirati sa svakim drugim uređajem koji je u njegovom dometu. To omogućava različite načine povezivanja i ostvarivanje redundantnosti. I dalje mora biti samo jedan PAN koordinator, ali sada ne mora sav saobraćaj ići preko koordinatora. Koordinator zadržava ulogu kontrole i internog PAN adresiranja. Čvorovi koji prosljeđuju i/ili rutiraju saobraćaj moraju biti FFD. Ova topologija pogodna je za IoT aplikacije koje obično nemaju jedan centralni čvor poput nadzora industrijskih postrojenja ili skladišta i inventara.

Obje topologije su prikazane na slici 4.4.



Slika 4.4: 802.15.4 - Topologije

Mreža povezanih čvorova ne mora se sastojati od samo jednog PAN-a. PAN koordinator može delegirati ulogu PAN koordinatora za dio mreže drugom FFD čvoru. Tada se oko tih čvorova formiraju novi PAN. Novi koordinatori i dalje imaju vezu sa PAN koordinatorom koji ih je promovisao. Na ovaj način se povećava rasprostranjenost mreže, ali se kompleksnost i kašnjenje. Primjer ovakve mreže prikazan je na slici 4.5.



Slika 4.5: 802.15.4 mreža - Stablo klastera [1]

Na fizičkom sloju IEEE 802.15.4., prema prvoj verziji iz 2003., koristi tri frekventna opsega iz ISM dijela spektra.

- Opseg 2.4 GHz se koristi u cijelom svijetu. Podijeljen je na 16 kanala i omogućava prenos podataka brzinama od 250 kb/s sa OQPSK (*offset quadrature phase-shift keying*) uz DSSS (*direct-sequence spread spectrum*) modulacijom.
- Opseg 915 MHz se koristi u Sjevernoj i Južnoj Americi i Australiji. Podijeljen je na 10 kanala i omogućava prenos podataka brzinama od 40 kb/s sa BPSK (*binary phase-shift keying*) modulacijom.
- Opseg 868 MHz se koristi u Evropi, Africi i na Bliskom istoku. Ima samo jedan kanal i omogućava prenos podataka brzinama od 20 kb/s sa BPSK modulacijom.

Kasnije verzije standarda su definisale dodatne frekventne opsege za druga geografska područja, dodatne modulacije i time povećale brzine prenosa na

opsegu 868 MHz na 100 kb/s, a na opsegu 915 MHz na 250 kb/s. Ovaj standard se stalno razvija i za očekivati je i dalje promjene.

Maksimalna veličina podatkovnog dijela 802.15.4 okvira je 127 bajta. To je mnogo manje nego kod Ethernet. To je takođe mnogo manje nego što je minimalna veličina MTU (*Maximum Transfer Unit*) kod IPv6. Iz tog razloga mora doći do fragmentacije IPv6 paketa ako se prenose preko 802.15.4 okvira.

U, ovom, podatkovnom dijelu fizičkog 802.15.4 okvira prenosi se MAC okvir. MAC okvir, uobičajeno, ima zaglavlje, sadržaj (*payload*) i podnožje (*footer*) okvira. Zaglavlje je podijeljeno na dio sa atributima okvira (*Frame Control*) veličine dva bajta, redni broj veličine jedan bajt i adresna polja veličine od četiri do 20 bajta. Atributi okvira su vrsta okvira, način adresiranja i dodatna kontrolna polja/zastavice. Redni broj je jedinstveni identifikator okvira. U adresnim poljima se nalaze identifikatori PAN izvorišta i odredišta, kao i izvorišna i odredišna adresa okvira. Sadržaj zavisi od vrste okvira koji se prenosi i može biti promjenljive dužine (naravno uz ograničenje da ukupna dužina MAC okvira ne može biti veća od 127 bajta). U *footer*-u se nalazi FCS (*Frame Check Sequence*) koji omogućava provjeru integriteta okvira. U njemu se nalazi 16-bit ITU-T CRC koji se računa za zaglavlje i sadržaj okvira.

Jedan od atributa okvira je bit koji definiše da li 802.15.4 MAC okvir ima ugrađene sigurnosne elemente. Ovo polje/bit se naziva *Security Enabled*. Ako je ovaj bit postavljen, na MAC zaglavlje se, nakon adresnih polja, dodaje dodatno sigurnosno zaglavlje (*Auxiliary Security Header*). Ovim se dodatno smanjuje veličina polja za sadržaj okvira. Veličina ovog dodatno zaglavlja zavisi od sigurnosnih funkcija i izabranog nivoa zaštite koji su ugrađene u okvir. Sigurnosne funkcije su funkcije su šifriranje, radi zaštite povjerljivosti, i *hash*-iranje sa ključem (MIC - *Message Integrity Code*), radi zaštite integriteta.

Izgled 802.15.4 MAC okvira sa svim poljima prikazan je na slici 4.6. Više detalja o svim poljima okvira može se pronaći u standardu [1].

Slika 4.6: 802.15.4 - MAC okvir

Postoje četiri predefinisana tipa 802.15.4 MAC okvira. Tip se definiše u atributima okvira. Svaki od okvira ima različitu namjenu. Tipovi i namjene su:

- *Beacon* - prenos *beacon*-a od PAN koordinatora
- MAC komandni - prenos komandi između uređaja

- Podatkovni - prenos podataka
- Potvrdni (*Acknowledgement*) - potvrđuje uredan prijem okvira

Pomoću ovih okvira 802.15.4 MAC sloj obavlja svoje osnovne funkcije: kontrolu pristupa mediju i omogućavanje pouzdane komunikacije između uređaja. Kontrola pristupa mediju se postiže definisanjem kako će uređaji koristiti dodijeljene frekvencije. To uključuje i raspoređivanje i rutiranje podatkovnih okvira. Protokol 802.15.4 podržava dva načina rada: *beacon* i *non-beacon*. U prvom načinu rada koordinator periodično šalju *beacon* okvire. Sa ovim okvirima koordinatori identificiraju PAN i omogućavaju sinhronizaciju čvorova u PAN-u. *Beacon* okviri označavaju početak, takozvanog, "superokvira". Strukturu "superokvira" definiše koordinator. Taj okvir je podijeljen na 16 vremenskih odsječaka jednakog trajanja. Koordinator određuje koliko prvih vremenskih odsječaka će biti namijenjeno za kompetitivni pristup mediju, a preostali vremenski odsječci su takozvani garantovani (GTS - *guaranteed time slots*). Garantovani vremenski odsječci se koriste da omoguće pristup mediju za aplikacije koje zahtijevaju malo kašnjenje ili garantovanu propusnost. Za kontrolu pristupa mediju u kompetitivnim vremenskim odsječcima i *non-beacon* načinu rada 802.15.4 koristi CSMA/CA (*Collision Sense Multiple Access/Collision Avoidance*) algoritam, poput 802.11 Wi-Fi standarda.

Uspostavljanje WPAN počinje od koordinatora koji bira dostupan kanal za komunikaciju. Bira se kanal na kom nema smetnji što se utvrđuje skeniranjem energije po svim kanalima na frekventnom opsegu na kom uređaj koordinator radi. Ostali uređaji se uključuju u WPAN kroz proces pridruživanja (*association*). Uređaj pronalazi dostupne WPAN. Traženje WPAN može biti pasivno ili aktivno. Pasivno se koristi kod mreža koje koriste *beacon*. Uređaj osluškuje i kad primi *beacon* ima informacije o WPAN. U mrežama u kojim koordinatori ne šalju *beacon*-e periodično, uređaj aktivno traži WPAN slanjem zahtjeva za *beacon*. Kada pronađe dostupne WPAN uređaj bira onaj kom se želi priključiti. Uređaj šalje zahtjev za pridruživanje WPAN koordinatoru ili drugom FFD uređaju koji je već član WPAN. Kada dobije potvrdan odgovor na zahtjev uređaj je postao dio WPAN. Uspostavlja se odnos roditelj dijete između čvora kom je poslan zahtjev i uređaja. Na taj način se i formira topologija mreže.

Pronalaženje najbolje putanje između čvorova (rutiranje) 802.15.4 mreže nije definisano standardom. To je zadatak viših slojeva i rade ga protokoli koji se oslanjaju na 802.15.4. Jedan od ovih protokola je opisan u nastavku. Verzije 802.15.4 se stalno razvijaju, a dvije zanimljive za IoT su 802.15.4e i 802.15.4g.

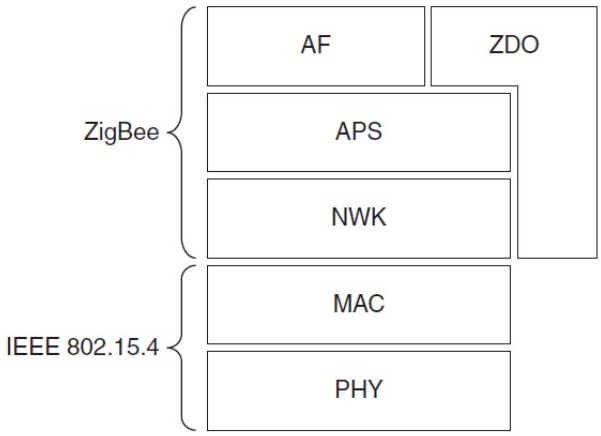
4.4.3 Zigbee

Zigbee je specifikacija skupa protokola koji se oslanjaju na IEEE 802.15.4 standard da pruži usluge fizičkog i MAC sloja ISO OSI mrežnog modela. Ova specifikacija omogućava interoperabilnost između uređaja različitih proizvođača.

Kao i 802.15.4 i Zigbee je namijenjen na povezivanje većeg broja uređaja ograničenih sposobnosti i energije na manjem prostoru veličine desetina metara. Razvojem specifikacija upravlja Zigbee Alliance, udruženje prouvođača. Ovo udruženje je za IEEE 802.15.4 nešto kao WiFi Alliance za IEEE 802.11. Zigbee je otvoreni vlasnički (*proprietary*) standard. Uređaji koji podržavaju Zigbee plaćaju naknadu za licenciranje koja garantuje poštovanje standarda i pravo na korištenje Zigbee loga. Zigbee je vjerovatno najpoznatija izvedba IEEE 802.15.4 pa se ponekad u literaturi pogrešno poistovjećuju. Zigbee je imao četiri verzije od 2004. godine, a trenutno važeća je ona iz 2007. godine [9].

Zigbee je pouzdan. Troši tako malo energije da Zigbee uređaji mogu raditi godinama sa istom AA baterijom. Zigbee je siguran i koristi 128 bitni AES šifrator, NIST standard. Domet i brzina prenosa kod Zigbee su isti kao i kod 802.15.4 jer je to podatkovni i fizički sloji koji Zigbee koristi.

Zigbee standard definiše slojeve od aplikativnog do mrežnog. Ovi slojevi imaju iste uloge kao i kod tradicionalnih računarskih mreža. Razlika u odnosu na tradicionalne mreže je što ovdje nema mogućnosti za različite protokole od onih definisanih standardom. Kompletan skup Zigbee slojeva prikazan je na slici 4.7.



Slika 4.7: Zigbee - skup protokola

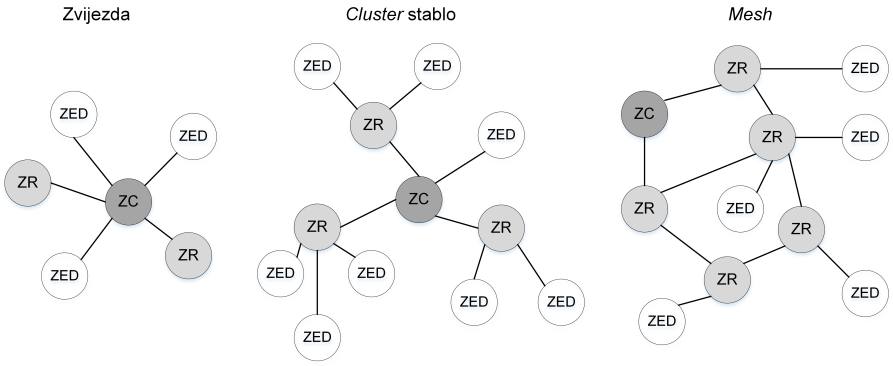
NWK je mrežni sloj i brine se za adresiranje i rutiranje. Umjesto IP adresa koristi drugačije mrežne adrese, NwkAddr. Ove adrese su dugačke 16 bita i na jedinstven način identifikuju čvor u Zigbee mreži. U drugoj Zigbee mreži, (W)PAN, mogu postojati iste NwkAdrese, ali je PAN ID te mreže drugačiji.

Zigbee mreža se može sastojati od tri tipa uređaja: Zigbee koordinator (ZC), Zigbee ruter (ZR) i Zigbee krajnji uređaj (ZED). Za svaku Zigbee mrežu

postoji samo jedan koordinator, ZC. ZC ima ulogu 802.15.4 PAN koordinatora. Kao i kod 802.15.4 ovu ulogu može obavljati samo uređaj pune funkcionalnosti, FFD. Zigbee ruteri (ZR) nisu obavezni, ali omogućavaju proširenje mreže jer preuzimaju ulogu prosljeđivanja paketa, koju bez njih obavlja samo ZC. ZR ulogu obavljaju FFD uređaji. ZED su krajnji uređaji koji mogu samo slati ili primati pakete. Oni ne mogu prosljeđivati pakete za druge čvorove. ZED su uglavnom stvari iz IoT, poput senzora ili prekidača. To mogu biti uređaji ograničene funkcionalnosti (RFD) koji imaju male mogućnosti i troše malo energije za rad.

Zigbee podržava tri osnovne topologije, prikazane na slici 4.8:

- Zvijezda - svi krajnji uređaji povezani su direktno na koordinatora. Ukupna veličina mreže je ograničena na po jedan korak od koordinatora i postoji samo jedna putanja između svakog para čvorova.
- *Cluster* stablo - na koordinatora se mogu vezati i Zigbee ruteri, a na njih krajnji uređaji. Ova mreža može biti veće stablo od jednostavne zvijezde, ali i dalje postoji samo jedna putanja između svakog para čvorova.
- *Mesh* - Zigbee ruteri se mogu međusobno povezivati, a ne samo preko koordinatora. Veličina ove mreže je, teoretski, ograničena maksimalnim brojem čvorova (2^{16}) i omogućava postojanje više različitih putanja između parova čvorova. To je dobro sa aspekta redundantnosti, ali komplikuje pronalaženje putanja i povećava potrošnju rutera



Slika 4.8: Zigbee - topologije

Pakete je moguće slati na jednu adresu (*unicast*) u mreži ili na sve (*broadcast*). Rutiranje može raditi mreža ili pošiljalac može navesti čvorove kroz koje paket treba da prođe na putu do odredišta (*source routing*). Mreža može pronalaziti putanje koristeći stablo, čiji korijen je koordinator, ili to može biti *mesh*. Za *mesh* rutiranje koristi se reaktivni algoritam zasnovan na AODV (Ad hoc On-demand Distance Vector) protokolu. Ovaj algoritam ne računa

putanje unaprijed, već po potrebi. Time se štedi energija i resursi, jer ruteri ne moraju stalno preračunavati putanje do svih čvorova i sve ih čuvati. Sa druge strane na ovaj način se usporava slanje jer se prilikom pristizanja paketa na ruter putanja izračunava, a nije unaprijed spremna.

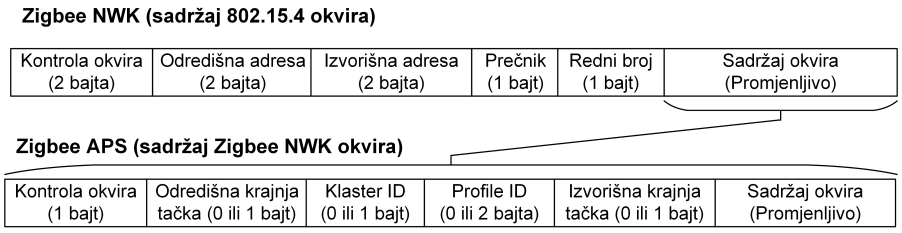
APS (*Application Support Sublayer*) je transportni sloj kod Zigbee. Slično kao i TCP ovaj sloj brine se za potvrđivanje paketa između krajnjih tačaka, koje uključuje i ponovno slanje i izbjegavanje dupliranja paketa. Adrese u APS su 8 bitne i one određuju kojoj krajnjoj tračci treba isporučiti sadržaj paketa i od koje krajnje tačke dolazi. Na jednom Zigbee čvoru može biti maksimalno 240 krajnjih tačaka (*Endpoint*). Krajnje tačke su poput Zigbee aplikacija. Odnose se na funkcije koje Zigbee uređaj može obavljati. Krajnja tačka može biti prekidač, mjerač temperature, sijalica, ... To znači da jedan Zigbee uređaj može omogućavati komunikaciju za više IoT uređaja.

Krajnje tačke pripadaju nekom od profila. Za različite namjene se koriste različiti profili. Profili mogu biti javni ili privatni (MSP - *Manufacturer-Specific Profile*). Javni profili su definisani od strane Zigbee alijanse, a privatni od strane nezavisnih proizvođača. Prvi profil koje je alijansa definisala je automatizacija domaćinstva (*Home Automation*). Ovaj profil ima broj 0104 heksadecimalno. Oznake profila su 16 bitne. Primjeri drugih javnih profila su *Industrial Plant Monitoring* (IPM), *Commercial Building Automation* (CBA).

Za profile se definišu 16 bitni klasteri koji definišu značenje. Na primjer broj 0006 heksadecimalno označava OnOff klaster. Njegova očigledna namjena je paljenje ili gašenje onoga na što se odnosi, poput svjetla, zvona, grijača. Klasteri sadrže i komande i atribute. Komande definišu akcije. Atributi čuvaju informacije o stanju klastera. Atribut OnOff za OnOff klaster pokazuje njegovo stanje. Aplikacija može tražiti da očita ili promjeni stanje. Klasteri imaju i pravac. Prekidač sa OnOff klasterom je izlazni i može se povezati sa svjetlom sa istim klasterom koje je ulazno. Ne mogu se povezivati ulazni sa ulaznim i izlazni sa izlaznim.

Javni Zigbee profili koriste Zigbee Cluster Library (ZCL). ZCL pojednostavljuje rad jer definiše 8-bitne komande specifične za klastere. Heksadecimalno 00 je komanda za gašenje (Off) za OnOff klaster. Ove komande čine sadržaj Zigbee paketa.

Izgled Zigbee paketa na različitim slojevima prikazan je na slici 4.9



Slika 4.9: Zigbee - enkapsulacija

Potrebno je još pomenuti krajnju tačku 0 koje je neophodna na svakom Zigbee uređaju. Ona je odgovorna za upravljanje uređajem. Naziva se ZDO (*Zigbee Device Object*). ZDO pronalazi neposredne susjede i usluge koje pružaju. Takođe upravlja ključevima za šifriranje i omogućava uspostavljanje sigurne konekcije između uređaja. Ima mehanizme za rad sa APS i NWK slojem, pa je tako i nacrtana na slici 4.7.

Za povezivanje Zigbee sa IP mrežama potreban je *gateway*. To ima svoje dobre i loše strane kako je objašnjeno u prvom dijelu poglavlja. Zigbee alijansa je radi olakšavanje povezivanja sa IP mrežama u aplikativnom profilu za *smart energy metering* definisala Zigbee IP koji je zasnovan na IPv6.

Pored Zigbee, postoje i druge specifikacije protokola koje se oslanjaju na 802.15.4 da pruži usluge fizičkog i podatkovnog sloja koje ovdje nisu obrađene. Oni uglavnom imaju domet definisan ograničenjima 802.15.4, a potrošnja im zavisi od izvedbe i usluga koje pružaju, te načina povezivanja čvorova i pronalaženja putanja.

4.4.4 IEEE 802.11ah

Prethodno pomenute WPAN mreže imaju relativno mali domet. Za veći domet potrebna je nešto drugačija tehnologija. WiFi 802.11 je dominantan protokol za WLAN, bežične lokalne mreže. Nedostatak ovog skupa protokola za upotrebu u IoT sistemima je relativno velika potrošnja energije. Standard je prilagođen uređajima koji imaju veće baterije koje se mogu redovno dopunjavati.

Da bi omogućio upotrebu 802.11 u IoT sistemima IEEE je napravio radnu grupu za novi standard 802.11ah. Glavni cilj ove grupe bio je smanjiti potrošnju energije potrebnu za rad. Drugi cilj bio je povećati broj čvorova koji se mogu povezati na jedan AP. Još jedna od promjena je povećan domet i bolji prolazak kroz prepreke, poput zidova. Prva verzija standarda objavljena je u decembru 2016. godine [2]. Drugi naziv koji se koristi za ovaj standard je WiFi HaLow.

Da bi se manjila potrošnja energije smanjilo se vrijeme koje je potrebno da uređaj bude povezan na mrežu. Uređaji trebaju biti aktivni u tačno određenim trenucima na vrlo kratak period vremena od nekoliko milisekundi svakih 15 sekundi. To je sasvim pogodno za IoT saobraćaj koji je povremen. Krajnji čvorovi sa AP dogovaraju vremenske trenutke kad će biti aktivni, TWT (*Target Wake Time*). Ovaj dogovor se postiže na osnovu očekivanog trajanja aktivnosti, recimo slanja podataka od krajnjeg čvora, i zahtjeva drugih krajnjih čvorova. AP raspoređuje TWT krajnjih čvorova da izbjegne ili minimizira preklapanja i potencijalnu borbu za pristup mediju. Na osnovu TWT krajnji čvorovi mogu biti neaktivni dok ne dođe njihov trenutak za komunikaciju. Dodatno, produženo je maksimalno vrijeme koje čvor može biti neaktivan (*Max Idle Period*) sa oko 16 sati u 802.11 na preko pet godina u 802.11ah. Još neke izmjene koje pomažu očuvanju energije navedene su kasnije.

802.11ah ne koristi standardne 802.11 frekvencije 2,4 i 5 Ghz, Noseća frekvencija koju koristi je iz dijela 900 MHz spektra koji ne podliježe licenciranju. Konkretne frekvencije zavise od zemlje. U Evropi je to opseg od 863 do 868 MHz. Upotreba nižih frekvencija povećala je domet i poboljšala prolaznost kroz prepreke.

Podržano je nekoliko širina kanala. Krajnji čvorovi su obavezni da imaju podršku za širine od 1 i 2 MHz. AP moraju podržavati i 4 MHz širinu kanala. Podrška za širine od 8 i 16 MHz je opcionalna. Što je širina kanala manja, domet je veći, ali je brzina prenosa podataka manja. Najmanja brzina bi trebala biti oko 150 kb/s na udaljenost od jednog kilometra, što zavisi od konkretnih okolnosti na terenu. Maksimalna teoretska brzina je 347 Mb/s sa 16 MHz širinom kanala i mnogo kraćim dometom.

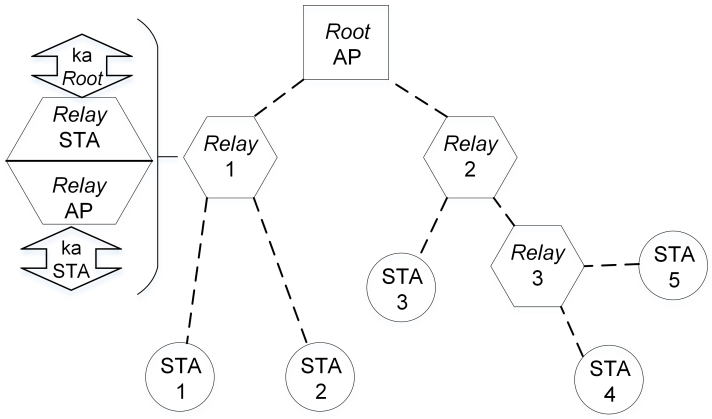
Povećavanje broja krajnjih čvorova koji se mogu povezati na jedan AP ostvareno je uvođenjem 13 bitnog AID (*association identifier*) koji omogućava povezivanje do 8191 krajnjih čvorova. Zaglavlja su smanjena u odnosu na druge verzije 802.11 da bi se smanjila količina kontrolnih informacija koje se prenose. To je bilo vrlo bitno zbog povećavanja broja čvorova. Da bi se smanjio broj kolizija koristi se RAW (*Restricted Access Window*). Krajnji čvorovi su razvrstani u grupe na osnovu AID i svaka grupa dobiva svoj vremenski interval za slanje i prijem podataka. Ovo je u skladu i sa potrošnjom energije jer svi uređaji iz neke grupe mogu otići u hibernaciju tokom perioda kada nije njihov interval aktivnosti.

Topologija 802.11ah mreža je nešto drugačija od drugih 802.11 mreža. Standardno u 802.11 postoje dvije vrste uređaja, sa aspekta topologije, krajnji čvor i pristupna tačka (AP). Kod 802.11h postoje tri vrste čvorova: krajnji čvor (STA), korijenska pristupna tačka (*Root* AP) i *Relay* čvor. Novi tip čvora *relay* ponaša se kao AP prema krajnjim čvorovima koji su povezani na njega, a kao krajnji čvor prema *Root* AP ili drugom *relay* čvoru na koji je povezan. Ovakva topologija, prikazana na slici 4.10 omogućava formiranje mnogo većih mreža, što je takođe pogodno za IoT sisteme.

Krajnji čvorovi mogu biti u jednom od tri stanja očuvanja energije: TIM (*Traffic Indication Map*), Non-TIM i neraspoređen. Krajnji čvorovi u TIM stanju osluškuju obavijesti (*beacon*) od AP u kom dobivaju informacije o podacima koji su za njih pristigli tokom perioda njihove neaktivnosti. Non-TIM krajnji čvorovi prilikom povezivanja sa AP dogovaraju vrijeme slanja kao PRAW (*Periodic Restricted Access Windows*). Neraspoređene stanice ne osluškuju saobraćaj od AP već prozivaju da bi dobili podatke za pristup mediju.

Pošto na jedan AP može biti povezan 8191 krajnji čvor TIM obavijesti mogu biti jako velike. Da bi se TIM obavijesti smanjile i dodatno smanjila potrošnja AP koristi TIM segmentaciju. TIM obavijesti odnose se samo na dio krajnjih čvorova. Krajnji čvorovi mogu izračunati kad će TIM obavijest koja se odnosi na njih biti objavljena i preći u hibernaciju do tada.

Još jedan dodatak u 802.11ah namijenjen za uštedu energije i racionalnu upotrebu medija su implicitne potvrde. Kad dva uređaja razmjenjuju podatke,



Slika 4.10: 802.11ah - topologija

slanje podataka od druge strane smatra se implicitnom potvrdom da je paket za tu stranu uredno stigao. Nema potrebe za posebnim slanjem potvrda, osim na kraju prije nego što krajnji čvor pređe u stanje neaktivnosti.

Upotreba 802.11ah je pogodna za povezivanje na druge mreže, Internet i *cloud*. Ovaj protokol podatkovnog sloja može koristiti IP mrežni sloj i kompletan skup TCP/IP protokola.

4.4.5 LoRaWAN

Do sada opisane tehnologije i protokoli podatkovnog sloja pogodni za IoT omogućavali su lokalno povezivanje na manjim udaljenostima koje se mjere stotinama metara. IoT sistemi imaju potrebu za povezivanjem i na veće udaljenosti i povezivanje u WAN. Za potrebe IoT razvijene su nove WAN tehnologije niske potrošnje energija, LPWAN (*Low Power Wide Area Network*). Jedan karakterističan predstavnik ovih tehnologija koji se sve više koristi je LoRa i LoRaWAN. Termin LoRa se odnosi na fizički, a LoRaWAN na MAC sloj.

LoRa radi u nelicenciranom dijelu spektra na frekvencijama od 433 i 868 MHz, u Evropi. Ovo je jako bitno radi poređenje sa drugim WAN tehnologijama koje su zasnovane na upotrebi mobilne telefonske mreže, odnosno licenciranog dijela spektra. Operatori mobilne telefonije plaćaju priličnu visoke cijene za korištenje svog dijela spektra, pa ga pokušavaju maksimalno racionalno koristiti. Ovo se obično prevodi u više cijene prenosa podataka po ovim, nego po nelicenciranim frekvencijama. Druga velika razlika je domet baznih stanica i, njihovog LoRa pandana, LoRaWAN *gateway*. LoRaWAN *gateway* ima domet od nekoliko kilometara u urbanim i do 15 km u ruralnim sredinama, što je mnogo više nego domet baznih stanica mobilne telefonije. Potrebno je 5 do 10 puta manje LoRaWAN *gateway* nego baznih stanica za

pokrivanje iste oblasti. Dobar primjer dometa LoRaWAN je sedam LoRaWAN *gateway* koji su bili dovoljni da se pokrije cijela površina Belgije.

Fizički LoRa sloj je vlasnički i dokumentacija nije javno dostupna. Koristi kanale širine 125 kHz. Koristi se šest ovih kanala za prenos po kojim se raspoređuje slanje. Što je slanje više raspoređeno (*spread*) manja je brzina prenosa, ali je domet veći. Brzine prenosa za LoRa su od 0,3 do 5 kb/s, a domet preko 10 km. Potrebno je primijetiti da su brzine mnogo manje nego kod WPAN i WLAN, ali i ove brzine mogu biti dovoljne za IoT potrebe, pogotovo na veće udaljenosti. LoRa je prilično robusna i otporna na smetnje. Podržava prilagodljivu brzinu, ADR (*Adaptive Data Rate*). Brzina slanja čvorova prilagođava se kvalitetu signala i njihovoj gustini. Čvorovi koji su bliže mogu slati većom brzinom, prije završiti slanje, osloboditi kanal i otići u neaktivno stanje koje štedi energiju. Udaljeniji čvorovi šalju manjim brzinama, duži vremenski period.

LoRaWAN nije jedina tehnologija koja koristi LoRa fizički sloj. Druge, konkurentske arhitekture oslanjaju se na LoRa, poput, Symphony Link i DASH7. U nastavku je obrađen samo LoRaWAN. Ovim specifikacijama, tehnologijama i certifikacijom upravlja LoRa alijansa, koja broji preko 160 članova među kojima je i kompanija vlasnik LoRa standarda Semtech, kao i IBM, Cisco i mnoge druge. Za razliku od LoRa fizičkog sloja, LoRaWAN je otvoren protokol. Važeća specifikacija, u vrijeme pisanja, je 1.1 [8].

LoRaWAN ima tri klase uređaja A, B i C i tri MAC protokola, po jedan za svaku klasu. Klasa A je optimizirana za minimalnu potrošnju energije, ali ima najveće kašnjenje. Klasa C minimizira kašnjenje, na uštrb povećane potrošnje energije. Klasa B je između klasa A i C. Klasa A namijenjena je za IoT uređaje sa ograničenim baterijskim napajanjem. Prilikom priključivanja mreži uređaja klase A definiše se kašnjenje prijema (*Receive Delay*) za svaki pojedini uređaj. Uređaj klase A šalje ono što ime, te nakon toga prelazi u neaktivno stanje dok ne istekne vrijeme kašnjenja prijema, kada se aktivira i prima podatke, ako ih ima. Ovo kašnjenje prijema može biti podešeno na veoma veliki period, u zavisnosti od namjene uređaja. Uređaj je prilikom prvog povezivanja na mrežu svrstan u klasu A, a onda može tražiti drugu klasifikaciju. Uređaji klase B oslanjaju se na obavijesti (*beacon*) koje šalje *gateway* u kojim su informacije o trenucima komunikacije za svaki od krajnjih čvorova. U periodu kad nije njegovo vrijeme za slanje i prijem, uređaj klase B je neaktivan. Uređaji klase C stalno osluškuju i to su uglavnom uređaji koji imaju stalno napajanje.

Veličina sadržaja LoRaWAN paketa je od 59 do 230 bajta za frekventni opseg 863–870 MHz, a od 19 do 250 bajta za frekventni opseg od 902–928 MHz.

Krajnji LoRaWAN čvorovi moraju se registrovati i potvrditi svoj identitet prilikom povezivanja na mrežu. Ovo može biti urađeno na dva načina: unaprijed ili dinamički prilikom pristupa. Registracija unaprijed, ARP (*Activation by Personalization*), podrazumijeva da su registracijski podaci koji uključuju i enkripcijske ključeve prekonfigurisani na uređaju i u LoRaWAN serveru. Prilikom povezivanja na mrežu nema posebne procedure jer je sve podešeno.

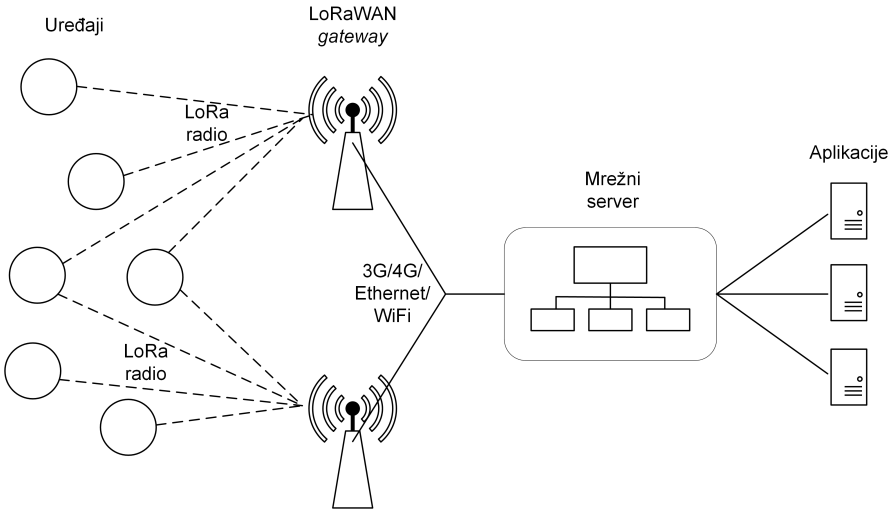
Dinamička, OTAA (*Over-the-air Activation*), aktivacija zahtjeva proceduru prilikom povezivanja na mrežu tokom koje se generišu enkripcijski ključevi. LoRaWAN šifrira podatke koje šalje koristeći 128 bitni AES algoritam.

LoRaWAN koristi nekoliko različitih načina adresiranja. Krajnji uređaji imaju globalno jedinstvene DevEUI adrese koje su im dodijelili proizvođači. To su EUI-64 adrese koje se generišu na osnovu MAC adresa i predstavljaju nešto kao adresu na podatkovnom sloju. Prilikom registracije u LoRaWAN mrežu krajnji uređaji dobivaju mrežnu adresu DevAddr. Ovo je 32 bitna adresa. Prvih sedam bita su identifikator mreže (NwkID). Preostalih 25 bita su identifikator uređaja (NwkAddr). LoRaWAN aplikacije imaju svoje AppEUI adrese. I ovo su adrese u EUI-64 formatu. Prvi dio adrese je dodijeljen pružaocu usluge, a drugi dio generiše server pružaoca usluge koji osigurava jedinstvenost adresa.

Topologija LoRaWAN je proširenje u odnosu na klasičnu zvijezdu. Više krajnjih uređaja se povezuje na jedan *gateway*, što je klasična zvijezda. Međutim jedan krajnji uređaj može biti povezan na više *gateway*, što je druga zvijezda. Zato se ova topologija naziva zvijezda od zvijezda (*star of stars*). LoRaWAN *gateway* ponaša se kao most (*bridge*) između LoRa mreže i mreže koja ga povezuje sa *cloud* LoRaWAN mrežnim serverima. Ta mreža je uglavnom neka od standardnih mrežnih tehnologija. LoRaWAN definiše samo podatkovni sloj. Upotrebu slojeva iznad ili slanje podataka po podatkovnom sloju definiše aplikacija. LoRaWAN mrežni serveri rade identifikaciju i eliminaciju duplih paketa koji su stigli po više *gateway*. Oni se brinu za rutiranje paketa do odgovarajuće aplikacije i njihovo potvrđivanje, te pružaju sigurnosne usluge.

LoRaWAN je dizajniran da omogući krajnjim uređajima da komuniciraju, preko velike udaljenosti, sa aplikacijama na Internetu. Ovo je različito od drugih razmatranih komunikacionih tehnologija koje omogućavaju komunikaciju između uređaja. Kod LoRaWAN komunikacija ide od krajnjeg uređaja do LoRaWAN mrežnog servera/usluge i aplikacije, te obratno od aplikacije/mrežnog servera do uređaja. Dio ovog prenosa, od krajnjih uređaja do LoRaWAN *gateway*, je po LoRa radio mreži. LoRaWAN *gateway* je sa mrežnim serverima povezan nekom drugom komunikacionom tehnologijom poput Ethernet-a, WiFi ili mobilne mreže. Topologija LoRaWAN prikazana je na slici 4.11.

Još jedna LPWAN tehnologija koja koristi nelicencirani ISM dio spektra je Sigfox. Sa jedne strane ovo je vrlo ograničena tehnologija po brzini slanja, veličini i broju poruka. Maksimalna brzina, veličina poruke i broj poruka za slanje od krajnjeg uređaja (*uplink*) su 100 b/s, 12 bajta i 140 poruka. Za slanje ka krajnjem uređaju (*downlink*) ove veličine su 600 b/s, osam bajta i četiri poruke. Sa druge strane domet Sigfox je veći nego kod LoRa i iznosi 50 km. Također, zbog malog opterećenja na centralni uređaj, na koji se krajnji uređaji povezuju, broj krajnjih uređaja po baznoj stanici može biti i jedan milion. Sigfox je pogodan za pokrivanje velikog područja sa velikim brojem uređaja koji šalju malo podataka.



Slika 4.11: LoRaWAN - topologija

4.4.6 NB-IoT

Davaoci usluga mobilne telefonije imaju mreže koje pokrivaju ogroman prostor, cijele države i kontinente. Te mreže su korištene za povezivanje mobilnih telefona. Potrebe ovih uređaja su da imaju stalno aktivnu vezu velike propusnosti koja im, danas, omogućava da gledaju video visoke rezolucije u realnom vremenu. Mobilna mreža je dizajnirana i usavršavana da zadovolji ove potrebe. Porastom broja IoT uređaja i povećanjem potrebe i tržišta za njihovo uvezivanje, mobilni operatori su vidjeli svoju priliku. Međutim potrebe IoT uređaja su drugačije nego mobilnih telefona. IoT uređaji šalju povremeno, male količine podataka i to može biti malom brzinom. IoT uređaji šalju više saobraćaja ka mreži, dok mobilni telefoni preuzimaju više podataka od mreže. Bilo je potrebno uklopiti ovakve uređaje u mobilnu mrežu i omogućiti im onoliko resursa koliko im je potrebno. Licence za korištenje frekvencija koje plaćaju mobilni operateri su visoke i potrebno je optimizirati upotrebu spektra. Davanje kanala koji koristi mobilni telefon IoT uređaju nije racionalno. Sa povećanjem broj IoT uređaja to neće biti ni moguće. Iako mobilni telefoni rada na baterije i potrebno je voditi računa o potrošnji energije, ipak se očekuje da baterija mobilnog telefona traje do nekoliko dana. Za IoT uređaje čija baterija treba da traje više godina vrlo je bitno koliko energije troši na komunikaciju.

Radi uklapanja IoT uređaja u mobilnu mrežu, organizacija za standardizaciju mobilnih mreža, uvodila je nove standarde pogodne za IoT. Ti standardi, poredani po redoslijedu pojavljivanja su LTE Cat-1, LTE-M i NB-IoT. Opšte promjene koje se odnose na sve ove standarde su da se umjesto punog dupleks prenos koristio polu dupleks. To je pojednostavilo komunikaciju, smanjilo po-

trošnju energije i snizilo cijenu uređaja. Od verzije LTE-M smanjena je širinu kanala sa standardnih 20 MHz za mobilne telefone na 1,4 MHz i smanjena snaga predajnika. Ovo je dovelo do daljeg pojednostavljenja, smanjene potrošnje energije i nižih cijena uređaja. Zanimljivo je da se i sa svakom novom verzijom brzina prenosa smanjivala, sa 10 Mb/s na stotine kb/s, a kašnjenje povećavalo, sa milisekundi na sekunde, što zvuči nelogično. Međutim, kad se uzme u obzir da su brzine bile mnogo veće, a kašnjenja mnogo manja, nego što je to potrebno IoT uređajima ova promjena je logična jer pomaže zadovoljavanju drugih potreba IoT sistema.

Najnoviji standard NB-IoT (*Narrow Band* IoT), naziva se i LTE Cat-NB. U ovoj verziji koristi se, u skladu sa nazivom, mnogo manja širina kanala od samo 180 kHz. Naravno ovo dalje snižava cijenu opreme i potrošnju energije. Brzina prenosa je desetine kb/s u prvoj verziji. U najnovijoj LTE Cat-NB2 je 127 kb/s ka uređaju i 159 kb/s od uređaja. Kašnjenje je reda sekundi. Ovaj standard, radi jednostavnosti, ne podržava mobilnost uređaja. To ga čini pogodnim samo za stacionarne IoT uređaje, što je još uvijek većina, ali treba imati u vidu. Suženi kanal, za razliku od ranijih standarda, ne omogućava slanje glasa ili slike, nego isključivo podataka.

Potrebno je naglasiti da je ovo prava tehnologija podatkovnog sloja. Podržava IP i ne zahtjeva posebne *gateway* koji će prevoditi između IP i ne IP adresiranja.

Izvedba NB-IoT, odnosno uklapanje kanala u kanale za mobilnu telefoniju, zavisi od izbora mobilnog operatera. Moguće je NB-IoT kanal ubaciti u veći LTE kanal, ubaciti ga umjesto GSM kanala ili ga čak ubaciti u zaštitni pojas između dva LTE kanala. Ubacivanje i multipleksiranje NB-IoT kanala u LTE kanal omogućava povezivanje do, teoretski, 200.000 IoT uređaja po ćeliji.

Kao i za druge tehnologije, i za NB-IoT, potrebna je podrška proizvođača opreme. Sve veći broj mobilnih operatora nudi ovu uslugu. Njena budućnost će dijelom zavisiti i od cijena i pouzdanosti.

4.4.7 5G

Na kraju je potrebno razmotriti i dolazeći standard u mobilnoj telefoniji 5G i njegovu pogodnost za IoT. Nova peta generacija mobilnih telefonskih mreža razvijana je uz svijest o potrebi da podrži i IoT. Pošto je 5G još uvijek u fazi probnog uvođenja može se samo navesti koje prednosti donosi za IoT. Iako se o novim generacijama mobilnih mreža uglavnom govori sa aspekta veće brzine, to za, postojeći, IoT nije potrebno. Za IoT su bitnije neke druge promjene, o kojima se ne piše toliko. U nastavku je 5G razmotren iz ugla onoga što donosi za IoT sisteme.

Prema najavljenim osobinama 5G bi trebao da 10 do 100 puta poveća broj uređaja koji se mogu povezati, u odnosu na LTE (4G). Trebao bi omogućiti pokrivanje cijele zemaljske kugle. Kašnjenje bi trebalo biti smanjeno ispod jedne milisekunde, a dostupnost bi trebala biti preko 99,999%. Ovo trenutno

nije toliko bitno za IoT, ali može postati kako se društvo sve više bude oslanjalo na povezane uređaje i njihovu komunikaciju. Posebna mogućnost 5G planirana za povezivanje velikog broja uređaja je da se očekuje 1000 puta veća propusnost, nego do sada, po jedinici površine. To bi značilo da na jednom kvadratnom kilometru može biti milion uređaja. Potrošnja energije mreže bi trebala da se smanji za 90%. Posebno definisana osobina vezana za IoT je da bi potrošnja energije krajnjih IoT uređaja vezana za 5G komunikacije trebala biti takva da im baterije trebaju trajati do 10 godina.

Navedene prednosti se ostvaruju na različite načine i donose neke dodatne zahtjeve. Planira se upotreba frekvencija od 24 do 100 GHz. Kako na ovim frekvencijama nije gužva mogu se planirati kanali širine i po 100 MHz. Ovo omogućava povećanje brzine. Međutim upotreba ovako visokih frekvencija smanjuje domet i povećava gubitke usljed prepreka. Da bi se ovo prevazišlo biće potrebno mnogo više baznih stanica.

5G zvuči obećavajuće. Trenutno je faza uvođenje i može se očekivati da će igrati bitnu ulogu u umrežavanju IoT uređaja.

4.5 Mrežni sloj

4.5.1 Enkapsulacija

Za adresiranje IoT uređaja IPv6 se nameće kao rješenje. IPv6 ima dovoljno adresa za sve sadašnje i IoT uređaje u predvidljivoj budućnosti. Ovaj protokol uklanja potrebu za NAT i omogućava dvosmjernu komunikaciju i adresiranje i identifikaciju svakog pojedinog uređaja. Podrška za mobilnost uređaja, pa i IoT, je takođe ugrađena u IPv6.

U prethodnom dijelu poglavlja je pokazano da je za slanje po mrežama malih propusnosti koje su izložene smetnjama bolje slati manje okvire. Za manje okvire je bitno da zaglavlje bude što manje da bi veći dio okvira bio iskorišten za prenos podataka. Veličina IPv6 adresa, koje čine glavninu IPv6 zaglavlja, je značajna u odnosu na veličinu okvira koji se koriste u IoT protokolima na podatkovnom sloju. Takav prenos bi bio neefikasan i ozbiljno bi uticao na smanjenje, ionako prilično male, efektivne brzine prenosa podataka.

Iz ovih razloga su smišljene metode enkapsulacije paketa koje omogućavaju adresiranje IoT čvorova bez potrebe da se prenose kompletna IPv6 zaglavlja.

Postoji posebno pitanje da li svaki IoT uređaj treba imati IP(v6) adresu. Za neke namjene je dovoljno lokalno adresiranje čvorova na podatkovnom sloju. U tom slučaju obično *gateway* ima IP adresu i brine se za povezivanje i prenos podataka ka i od uređaja koji su vezani za njega.

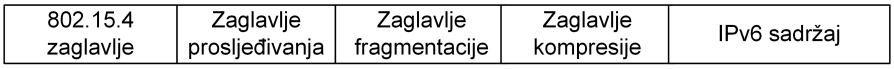
4.5.2 6LoWPAN

Sa ciljem da se formalizuje efikasniji način enkapsulacije IPv6 paketa u 802.15.4 okvire, IETF je formirao radnu grupu 6LoWPAN. Ta radna grupa

je usvojila RFC 4944 [21] koji standardizira odgovor na ovo pitanje. Standard definiše format 802.15.4 okvira koji prenosi IPv6 pakete i mehanizam kompresije zaglavlja da bi ovaj prenos bio praktičan. 6LoWPAN predstavlja adaptacijski sloj između mrežnog, IPv6, i podatkovnog, 802.15.4.

IETF nije razmatrao enkapsulaciju IPv4 paketa. IPv4 zaglavlje je znatno manje nego IPv6, prvenstveno zbog adresa, pa je problem efikasnosti manje izražen. Nadalje, prostor IPv4 adresa je iscrpljen postojećim, standardnim, uređajima i ne očekuje se da će IoT uređaji dobivati IPv4 adrese.

Uloga 6LoWPAN je da obavlja kompresiju IPv6 zaglavlja da bi se trošio manji dio sadržaja 802.15.4 okvira na zaglavlja IPV6, a više za sadržaj IPv6 paketa. Pored toga omogućava fragmentaciju IPv6 paketa tokom prenosa kroz 802.15.4 mrežu, ako je potrebna, i ponovno sastavljanje na kraju. Ovaj adaptacioni sloj omogućava i da se paketi prosljeđuju preko više čvorova na podatkovnom sloju, bez potrebe za adresama mrežnog sloja. Za svaku od ovih uloga postoji posebno zaglavlje. Zaglavlja se pojavljuju tačno definisanim redom i samo ako se koriste. Prvo je zaglavlje prosljeđivanja na podatkovnom sloju, pa fragmentacije, a posljednje, najbliže IPv6 sadržaju paketa, zaglavlje kompresije. Redoslijed 6LoWPAN zaglavlja unutar 802.15.4 okvira prikazan je na slici 4.12.



Slika 4.12: 6LoWPAN zaglavlja

Kod 6LoWPAN zaglavlje IPv6 paketa se mijenja sa 6LoWPAN zaglavljem, koje se na odredištu, izlasku iz dijela mreže u kom se koristi 6LoWPAN enkapsulacija, vraća u prvobitni oblik. Sadržaj IPv6 paketa se ne mijenja, ali može biti podijeljen i poslan u dijelovima kao sadržaj više 6LoWPAN paketa.

Prvo 6LoWPAN zaglavlje koje je najbliže sadržaju IPv6 paketa je zaglavlje kompresije IPv6 zaglavlja. Potreba za kompresijom IPv6 zaglavlja paketa koji se prenose unutar 802.15.4 okvira se može lako objasniti. MTU za 802.15.4 je 127 bajta. Od ove veličine treba odbiti 802.15.4 zaglavlja i FCS koji mogu biti od 9 pa do 46 bajta. To znači da za cijeli IPv6 paket ostaje od 81 do 121 bajta. IPv6 zaglavlje je minimalno dugačko 40 bajta. Od ovih 40, 32 bajta su izvorišna i odredišna adresa. Unutar IPv6 paketa je transportni protokol, UDP ili TCP, koji ima 8 ili 20 bajta zaglavlja. U najgorem slučaju za sadržaj paketa može ostati samo 21 bajt, što teško može biti dovoljno za bilo kakve podatke. Kako je IPv6 zaglavlje najveće, najveća ušteda se može postići njegovom kompresijom.

Nivo i tehnika kompresije mogu biti različiti, ovdje je samo objašnjena ideja, a više detalja može se naći u RFC. Ideja je da se standardna zaglavlja čije se vrijednosti ne mijenjaju, poput polja verzija IP koje je uvijek 6, ne prenose ili zapisuju sa manje bita. Ako su IPv6 adrese izvedene iz adresa

na podatkovnom sloju, što je često slučaj mogu se izostaviti, jer se mogu izračunati iz MAC adresa. Dužina paketa se može izračunati iz vrijednosti za dužinu 802.15.4 okvira. Neke uobičajene vrijednosti polja u zaglavlju, poput oznake toka koja je obično 0, mogu se izostaviti. Jedino polje koje se ne može skratiti je brojač skokova (*hop count*). Ovakvom kompresijom se 40 bajta IPv6 zaglavlja može sažeti na tri bajta, što je velika ušteda. Dodatna ušteda može se ostvariti kompresijom UDP transportnog zaglavlja, koju 6LoWPAN također omogućava, sa osam na četiri bajta.

Fragmentacijsko zaglavlje 6LoWPAN omogućava da se IPv6 pakte čiji sadržaj ne može stati u 802.15.4 okvir fragmentira na način da se može vratiti u originalni oblik. Fragmentacija se radi po sličnom principu kao i IPv4 fragmentacija. Za svaki fragment paketa se u ovo zaglavlje upiše ukupna veličina originalnog IPv6 paketa sa 11 bita, jedinstvena 16 bitna oznaka paketa, te osam bitna udaljenost tog dijela od početka originalnog paketa. Posljednje polje je nula za prvi fragment. pa se ne koristi za taj fragment. Pet bita prije ovih polja fragmentacije imaju vrijednost 11000, koja govori da se radi o fragmentiranom paketu, pa je ukupna dužina ovog zaglavlja pet bajta. Bitno je reći da u IoT sistemima, koji uglavnom šalju male količine podataka, kompresija zaglavlja je često dovoljna, i sadržaj može stati u jedan 802.15.4/6LoWPAN/IPv6 paket, te fragmentacija nije potrebna.

Zaglavlje prosljeđivanja na podatkovnom sloju (*Mesh Addressing*) pruža podršku da se za prosljeđivanje paketa koristi *mesh* rutiranje koje 802.15.4 omogućava. U ovom zaglavlju se navode izvorišna i konačna odredišna 802.15.4 adresa IPv6 paketa. Adrese mogu biti pune EUi-64 i kratke 16-bitne adrese. Koja vrsta adrese se koristi definisano je za izvorišnu trećim, a za odredišnu četvrtim bitom prvog bajta zaglavlja. Prva dva bita su 10 i označavaju da se radi o ovakvom (tzv. *mesh under*) rutiranju. Preostala četiri bita prvog bajta zaglavlja je brojač skokova koji se smanjuje za jedan na svakom 802.15.4 čvoru kroz koji prođe.

Enkapsulacija, odnosno adaptacijski sloj, koji 6LoWPAN pruža prilagođava IoT protokol podatkovnog sloja sa standardnim mrežnim protokolom. IETF razvija i druge standarde za pakovanje IPv6 paketa u okvire drugih resursima ograničenih WPAN mreže. Jedan od rezultata je i RFC 7668 "IPv6 over Bluetooth(R) Low Energy" [26].

Jedan od novijih IoT skupova protokola koji se oslanja na 6LoWPAN je Thread. Protokol je nastao 2014. godine i još se intenzivno razvija. Thread Group alijansa koja razvija protokol ima velike kompanije, poput Google, Apple, Samsung i Qualcomm, kao svoje članice. Protokol je fokusiran na automatizaciju domaćinstva. Konkurencija je Zigbee i Z-wave, ali se više oslanja na standardne mrežne protokole poput IPv6, UDP i TLS, te ima veću slobodu izbora aplikativnog protokola.

4.5.3 Rutiranje

IoT sistemi mogu koristiti protokole rutiranja iz tradicionalnih računarskih mreža poput OSPF ili IS-IS. Međutim, postoje neke razlike između tradicionalnih računarskih i IoT mreža koje stvaraju potrebu za novim protokolima. IoT mreže, odnosno prethodno pomenute tehnologije podatkovnog i fizičkog sloja koje se koriste, su mnogo manje pouzdane od Ethernet mreža. U Ethernet mrežama pogrešno primljeni biti su vrlo rijetki, dok je to u IoT mrežama mnogo češći slučaj. Ove mreže su pretežno bežične i rade u nelicenciranom dijelu spektra gdje se smetnje ne mogu spriječiti. Pored toga čvorovi u ovim mrežama uglavnom imaju ograničeno napajanje i time ograničenu količinu energije koju mogu trošiti na komunikaciju i prosljeđivanje. Ovakve mreže se nazivaju LLN (*Low-power and Lossy Networks*). U nastavku će se koristiti termin LLN. LLN uglavnom imaju i relativno malu propusnost u odnosu na Ethernet.

Tradicionalni protokoli rutiranja su dinamički i prilagođavaju se promjenama u mreži. Oni pronalaze nove putanje kroz mrežu kada dođe do otkaza neke od veza ili kada se veza oporavi ili pojavi nova. Ipak, ove promjene su u žičanim mrežama relativno rijetke i obično su dugotrajnije. Kod LLN ispadi veza su mnogo češći i kratkotrajniji. Protokoli koji bi za svaku takvu promjenu preračunavali nove putanje kroz mrežu u svim ruterima bi intenzivno radili i vrlo često računali. To bi dovelo do povećanje potrošnje energije u čvorovima, do moguće nestabilnosti u slučaju ispada i vračanja iste veze više puta u kratkom vremenu, te generisanja velike količine kontrolnog saobraćaja kroz mrežu. Sve od navedenog je u suprotnosti sa ograničenjima LLN.

Saobraćaj u IoT mrežama je vrlo često od velikog broja krajnjih senzora do centralnog čvora (*multipoint to point*), ili suprotno, od centralnog čvora ka velikom broju krajnjih čvorova (*point to multipoint*). Bilo bi pogodno da protokol rutiranja podržava ovakav saobraćaj, pored standardnog od jednog do drugog čvora (*point to point*). IoT mreže uglavnom imaju veći broj čvorova nego AS unutar kojih se koriste tradicionalni protokoli rutiranja. IoT ruting protokol bi trebao biti skalabilan i omogućavati efikasno računanje putanja i pri velikom broju čvorova. IoT čvorovi koji računaju putanje i prosljeđuju pakete imaju ograničeno napajanje, pa je potreban energetski efikasan protokol rutiranja.

Iz očigledne potrebe za novim protokolom rutiranja za IoT, IETF je formirao radnu grupu ROLL (*Routing Over Low-power and Lossy networks*). Razmotrivši navedene, i druge, razlike, posebnosti i potrebe LLN, radna grupa je predložila novi protokol IPv6 rutiranja za LLN i nazala ga RPL. Protokol je postao standard 2012. godine i objavljen je u RFC 6550 [62].

4.5.4 RPL

Kod RPL svaki čvor je dio *mesh* mreže i radi prosljeđivanje paketa. Rutiranje se radi na mrežnom nivou i paketi se prosljeđuju na osnovu IPv6 zaglavlja.

Ovo je suprotno od *mesh under* rutiranja koje je pomenuto u 6LoWPAN koje radi na podatkovnom sloju. Zato se ovakvo rutiranje naziva *mesh over*.

Postoje dva načina rada čvorova:

- Pohranjivanje (*Storing mode*) - Poput tradicionalnog rutiranja, svi čvorovi imaju kompletne tabele prosljeđivanje na osnovu kojih mogu prosljeđivati pakete za sva čvorove u RPL mreži. Ovaj način rada nije energetski optimiziran.
- Nepohranjivanje (*Non-storing mode*) - Samo rubni ruter domena ima kompletnu tabelu prosljeđivanja. Ostali čvorovi samo znaju čvorove iznad njih koji vode do rubnog rutera (*parents*), koji onda zna kako proslijediti paket do odredišnog čvora. Prosljeđivanje se uvijek vrši od čvora ka rubnom ruteru. Ovaj način rada štedi resurse na čvorovima, procesor i memoriju.

RPL je zasnovan na direktnom acikličkom grafu (DAG). DAG je usmjereni graf u kom nema petlji. To znači da nije moguće krenuti iz bilo kog čvora i prateći usmjerene ivice grafa doći u isti čvor. Sve veze (ivice) su organizovane u putanje koje od čvorova vode do jednog ili više korijenskih čvorova.

Pošto DAG može imati više korijenskih čvorova, RPL od takvog grafa pravi onaj sa jednim korijenskim čvorom. Takav DAG se naziva odredištu orijentisan (*destination-oriented*) DAG (DODAG). To jedno odredište je kod RPL rubni ruter. U DODAG, svaki čvor ima do tri čvora preko kojih može doći do korijenskog čvora. Ti čvorovi se nazivaju roditelji. Jedan od roditelja je preferiran, odnosno preko njega vodi preferirana putanja do korijena.

Graf rutiranja, nastao kad svi čvorovi ovako odrede roditelje, predstavlja cjelokupan skup putanja do korijena. Da bi onemogućio petlje, RPL zabranjuje izbor narednog čvora koji je dalje od korijena. RPL koristi DIO (DAG *Information Object*) poruke za otkrivanje i konfiguraciju putanja do korijena. Čvorovi na osnovu ovih poruka saznaju za promjene topologije. Iz DIO poruka utvrđuju se roditelji i najbolje putanje do korijena.

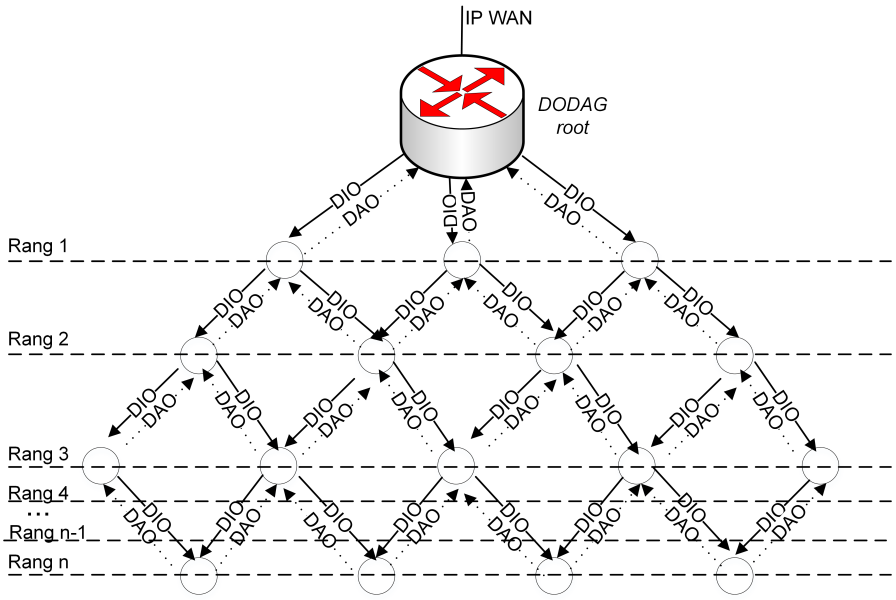
Putanje od korijena otkriva šaljući svojim roditeljima DAO (*Destination Advertisement Object*) pouke. Ovim porukama čvorovi informišu svoje roditelje o svom postojanju i dostupnosti čvorova ispod njih.

U slučaj RPL nepohranjujućeg načina rada svi čvorovi DAO porukama rubnom ruteru (korijenu) javljaju skupove svojih roditelja, jer u ovom načinu rada samo rubni ruter čuva informacija o putanjama. Na osnovu ovih informacija rubni ruter utvrđuje putanje do pojedinih čvorova u *mesh* mreži.

Kod pohranjujućeg načina rada, svaki čvor pohranjuje informacije o putanjama iz DAO poruka. Ovo zahtjeva više resursa po čvoru ali omogućava kratice između čvorova koje ne vode preko rubnog rutera. U ovom načinu rada svaki čvor može sam odrediti putanju.

DIO i DAO poruke šalju se unutar IPv6 paketa. Pomoću ovih poruka objavljuju se i razmjenjuju putanje ka i od korijena između rubnog rutera i ostalih čvorova. Na slici 4.13 prikazan je tok DIO i DAO poruka.

RPL ima poseban RFC 6551 u kom su definisane različite metrike koje se mogu koristiti za izbor najbolje putanje.



Slika 4.13: RPL poruke

Neke od metrika su identične metrikama iz tradicionalnih protokola rutiranja poput:

- Broj skokova
- Propusnost
- Kašnjenje

Druge metrike se karakteristične za IoT, odnosno LLN, poput:

- Energija čvora - izbjegavaju se čvorovi manje energije (napajanja) time se omogućava duži rad čvorova sa baterijskim napajanjem.
- Nivo kvaliteta veze - pokazuje na pouzdanost linka i određuje se na osnovu intenziteta grešaka, koje mogu biti izazvane prigušenjem i smetnjama.
- Stanje i atributi čvora - utvrđuje čvorove agregatore i preopterećene čvorove, preferiraju se putanje preko agregatora.
- Očekivani broj transakcija (ETX) - očekivani broj transakcija potreban za isporuku paketa.
- Boja veze - omogućava prioritetizaciju veza administrativnim postavkama vrijednosti koje mogu biti statičke ili dinamičke.

Dozvoljene su i druge metrike i ograničenje koje nisu navedene u RFC 6551.

RPL čvorovi imaju rang. Rang je skalar koji predstavlja poziciju čvora prema drugim čvorovima u odnosu na korijen. Rang raste udaljavanjem, a opada sa približavanjem korijenu. Rang pomaže da se izbjegnu petlje. Način računanja ranga određen je funkcijom cilja.

RPL funkcija cilja definiše kako čvorovi biraju i optimiziraju putanje. Funkcija cilja određuje kako čvorovi prevode metrike i ograničenja u rang. Ova funkcija definiše kako čvorovi biraju roditelje. Više detalja o funkcijama cilja za različite namjene navedeno je u RFC 6552 i RFC 6719.

Protokolima RFC 6553 i RFC 6554 definisana su posebna IPv6 zaglavlja za prenos RPL informacija. Zaglavlje RPL opcija dio je *Hop-by-Hop* IPv6 zaglavlja. Zaglavlje izvornog rutiranja omogućava RPL ruterima slanje informacije o željenoj putanji.

Integracija RPL u ruting domen je slična kao i za tradicionalne protokole. Izvedba razmjene putanja, filtriranja, raspoređivanja opterećenja i dinamičkog preračunavanja putanja može biti ista kao i za druge, stare, protokole.

Kako se može zaključiti, RPL je nešto složeniji protokol od tradicionalnih zbog većeg broja metrika, ograničenja i funkcija cilja. RPL omogućava da se IPv6 koristi za mreže sa velikim brojem čvorova. Prilagođen je za mreže koje se sastoje od čvorova ograničenih resursa. Zbog svojih osobina postao je jedan od glavnih IPv6 protokola rutiranja u IoT mrežama.

4.6 Transportni sloj

Na transportnom sloju nema posebnih IoT protokola koji se dovoljno često koriste. Na ovom sloju ostaje da se riješi koji je od dva transportna protokola, TCP ili UDP, pogodniji za IoT. Kratak odgovor je da nema jasnog pobjednika i da izbor zavisi od namjene.

Prednosti UDP su u malom zaglavlju i jednostavnom protokolu. Mala zaglavlja troše manje ograničene propusnosti veza u IoT mrežama. Jednostavan protokol zahtjeva manje resursa i troši manje energije od čvorova. Primjene IoT u kojim pouzdanost nije od presudnog značaja pogodne su za UDP. To je slučaj za veliki broj aplikacija u kojim senzori redovno šalju podatke. Poneka izgubljena vrijednost nije presudna. Jedan od nedostataka UDP je upravo ta nepouzdanost. Ako je potreban oporavak izgubljenih poruka, onda to mora biti urađeno od strane aplikacije. Takve aplikacije su onda komplikovanije što nije pogodno ako se izvršavaju na ograničenim IoT čvorovima. Drugi bitan nedostatak UDP je nemogućnost segmentacije poruka koje šalje i prilagođavanja MTU na putu od pošiljaoca do primaoca. Kako je u IoT mrežama MTU često mali, na aplikacijama je da se pobrinu da poruke koje šalju po UDP budu veličine koja se može uklopiti u taj MTU.

I TCP ima svojih prednosti za primjenu u IoT sistemima. Pouzdanost je jedna od njih. IoT mreže često koriste nepouzdane veze. Takođe brzina prenosa im nije presudna. TCP je pogodan za ovakve sisteme, jer šalje nešto sporije u mrežama koje gube podatke, ali osigurava pouzdanost. Za komunikaciju sa vanjskim, ne IoT, sistemima, koji pretežno korite TCP, pogodno je ako IoT sistem koristi TCP. Nedostatak segmentacije, pomenut kao nepogodnost UDP, je prednost TCP. TCP, kroz MSS, može podijeliti aplikativne poruke u

segmente dovoljno male da se mogu poslati po vezama sa malim MTU, karakterističnim za IoT mreže. TCP se doima kompleksnim, pogotovo u uporedbi sa UDP. Dosta kompleksnosti otpada ne ostvarenje visoke propusnosti. Ako se izostave ovi mehanizmi za visoku propusnost, moguće je napraviti izvedbe TCP koje se mogu izvršavati na čvorovima sa ograničenim resursima. Pored kompleksnosti, nedostatak TCP je veličina zaglavlja koja je dosta veća nego kod UDP.

4.7 Aplikativni sloj

Kako je u poglavlju 2 o tradicionalnim računarskim mrežama rečeno, aplikativni sloj omogućava korištenje usluga računarske mreže. Usluge se koriste kroz razmjenu informacija. Mrežne aplikacije prave, adresiraju i šalju poruke na jednoj i primaju, analiziraju i poduzimaju odgovarajuću akciju u skladu sa sadržajem poruke na drugoj strani. Aplikativni protokol definiše sadržaj poruka, njihov format i akcije na prijemnoj strani. Savremene, pa i tradicionalne, mrežne usluge mogu biti vrlo kompleksne, pa u skladu sa njima i protokoli koji ih podržavaju mogu biti kompleksni. Međutim, kako je objašnjeno, najkorišteniji aplikativni protokol HTTP je relativno jednostavan. Ipak ovaj protokol omogućava veliki broj usluga. To je ostvareno aplikacijama koje koriste HTTP za prenos poruka, a obrada poruka i njihovo djelovanje ostvaruju se unutar aplikacije, nezavisno od HTTP.

Veliki broj primjena IoT je vrlo jednostavna razmjena informacija. U najjednostavnijem slučaju senzor šalje jednu vrijednost jedne veličine. To često radi u pravilnim vremenskim razmacima. Pošto se tačno zna šta senzor šalje nisu potrebna nikakva dodatna objašnjenja uz vrijednost koja se šalje. U ovakvim slučajevima sistem može funkcionisati i bez protokola aplikativnog sloja. Vrijednost koja se šalje se može jednostavno proslijediti transportnom sloju i tu vrijednost će dobiti primalac. Rad bez protokola aplikativnog sloja pojednostavljuje IoT sistem i štedi resurse u čvorovima, pogotovo onim koji šalju podatke. Iako ovo nije uobičajeno vrijedno je pomena i razmatranja u nekim situacijama. Veliki nedostatak ovog pristupa je nefleksibilnost. Bilo kakva promjena funkcionalnosti može biti komplikovana ili čak neizvodiva.

4.7.1 HTTP za IoT

Jedan pristup pitanju aplikativnog protokola za IoT je upotreba HTTP. HTTP je dobro poznat protokol i razvoj aplikacija koje rade po HTTP je dobro uhodan. Prikupljanje podataka i izdavanje komandi, putem HTTP može se organizovati na dva načina. Jedan način je da IoT krajnji uređaji imaju ulogu HTTP klijenata, a da neki centralni čvor koji prikuplja informacija i izdaje komande (i koji uglavnom ima stalno napajanje i veće resurse) ima ulogu HTTP servera. U ovom slučaj senzori mogu slati HTTP PUT zahtjeve sa onim što šalju u tijelu HTTP zahtjeva. Ovakvi HTTP klijenti se mogu

napraviti da budu vrlo jednostavni i da im je potrebno malo resursa za rad. Aktuatori bi, kao HTTP klijenti mogli periodično slati HTTP GET zahtjeve kojima traže komande. Server bi u HTTP odgovoru slao komande sa parametrima. Ovi HTTP klijenti bi bili nešto složeniji nego za senzore, ali još uvijek izvodivi na uređajima koji nisu ekstremno ograničenih resursa. Potrebno je povesti računa o resursima i opterećenosti HTTP servera u ovoj konfiguraciji, posebno ako je broj HTTP klijenata, senzora i aktuatora, veliki.

Druga mogućnost je da krajnji IoT uređaji rade kao HTTP serveri. U tom slučaju centralni IoT prikupljač podataka ima ulogu HTTP klijenta i putem HTTP GET zahtjeva definiše koje veličine želi od HTTP servera, IoT senzora. Centralni uređaj može izdavati komande aktuatorima putem HTTP POST zahtjeva u kojim specificira komande i njihove parametre. Ovaj pristup je nešto fleksibilniji jer omogućava centralnom uređaju da bude inicijator i upravlja radom sistema tako što prikuplja informacije i izdaje komande koje želi i kad želi. Ovaj pristup se čini zahtjevnijim po pitanju resursa krajnjih IoT uređaja koji moraju imati izvedbu HTTP servera. To je tačno, ali se potrebni resursi mogu smanjiti. Umjesto HTTP servera na krajnjem IoT uređaju dovoljno je imati REST usluge. Ovim se može smanjiti potrebna memorija i energija za rad. Istraživanja posvećena resursima potrebnim za izvedbu HTTP na IoT uređajima pokazala su da je dovoljno nekoliko kilobajta memorijskog prostora i nekoliko mW energije [66][44].

Da bi podaci koje šalju krajnji i centralni IoT uređaji bili razumljivi potrebno je da njihov format bude jasno definisan. Jedno rješenje je da svaka izvedbe IoT ili svi uređaji istog proizvođača koriste interno dogovoreni protokol. Ova izvedba može donijeti neke prednosti u formatu koji je pogodan za određenu namjenu, ali nije dovoljno univerzalna, dovodi do zatvaranja u ograničena rješenja pojedinih proizvođača ili za pojedine namjene. Za ostvarivanje veće kompatibilnosti i otvorenosti bolje je koristiti međunarodno prepoznate načine formatiranja. Vrlo čest format, koji je možda još uvijek preovladavajući u poslovnim aplikacijama koje koriste HTTP za prenos SOAP poruka, je XML. XML je specifikacija koja precizno iskazuje pravila kodiranja podataka. Vrlo je pogodna za razne upotrebe, ali zahtjeva priličan broj znakova da iskaže sva pravila. Ovo XML formatiranje podatke čini relativno obimnim i njihovo parsiranje može biti relativno sporo i zahtijevati više resursa. Ta osobina nije pogodna za IoT.

Laganija alternativa formatiranja podataka koja počinje da dominira u web aplikacijama koje koriste REST pristup u vrijeme pisanja je JSON (*JavaScript Object Notation*). JSON je manje obiman i lakši za parsiranje od XML-a. Ovo ga čini pogodnim za IoT namjene.

Još jedna prednost HTTP je lagano i standardizovano dodavanje zaštite podataka, ako je potrebna, putem TLS, Ovo se posebno odnosi na nove verzije HTTP/2 i HTTP/3, opisane u poglavlju 3 o HTTP. S obzirom na historijsku uspješnost HTTP za očekivati je da će REST i JSON igrati važnu ulogu i u IoT sistemima.

Ipak postoje namjenski IoT aplikativni protokoli koji su manje kompleksni od IoT i zahtijevaju manje resursa za rad. Dva takva protokola, koja se najviše koriste u praksi, opisana su u nastavku.

4.7.2 CoAP

Na osnovu dobrih iskustava i prednosti HTTP, odnosno REST arhitekture, vodeći računa o potrebama uređaja i mreža ograničenih resursa, IETF radna grupa Constrained RESTful Environments predložila je novi protokol CoAP (*Constrained Application Protocol*). Protokol je definisan u RFC 7252 [68]. U nastavku je dat kratak opis bitnih osobina ovog protokola, a više detalja se može pronaći u navedenom RFC.

CoAP koristi poznate dijelove REST arhitekture i HTTP. Resursi se definišu korištenjem URI. Podržane su metode GET, POST, PUT i DELETE. Ova sličnost sa HTTP olakšava razvoj aplikacija i mapiranje CoAP poruka na HTTP. Pogodnosti CoAP za IoT, u odnosu na HTTP, su što ima gotovo deset puta manje bajta i dva puta manje energije po transakciji [14].

CoAP minimizira količinu dodatnih bajta koje šalje uz sadržaj poruka. Ima vrlo malo zaglavlje i koristi UDP kao transportni protokol. Obavezno CoAP zaglavlje dugačko je samo četiri bajta. Uz UDP zaglavlje od osam bajta ukupna zaglavlja transportnog i aplikativnog sloja su 12 bajta što je mnogo manje od HTTP/TCP kombinacije. Manja zaglavlja uz manji sadržaj, što je uobičajeno za IoT mreže, omogućavaju da se CoAP/UDP paket može uklopiti u ograničenja IP/Ethernet, pa i ograničenijih IP/802.15.4, slojeva. Skraćivanje zaglavlja u odnosu na HTTP ostvareno je njihovim pojednostavljivanjem i binarnim kodiranjem.[4] Umjesto ASCII zaglavlja koristi se jednobajtni kod u koji se zapisuje metoda zahtjeva ili kod odgovora.

Način rada CoAP je vrlo sličan načinu rada HTTP. CoAP klijent šalje zahtjeve, a CoAP server šalje odgovore. Krajnji IoT uređaji mogu biti i klijenti i serveri. U većini CoAP softvera podržane su obje uloge istovremeno. Njihova uloga zavisi od načina izvedbe prikupljanja podataka ili slanja komandi. Primjer izvedbe je centralni uređaj koji prikuplja podatke sa krajnjih IoT uređaja putem slanja GET zahtjeva, što znači da ima ulogu CoAP klijenta. Krajnji IoT uređaj odgovara na GET zahtjev slanjem odgovora čiji sadržaj je, ako je sve kako treba, vrijednost mjerene veličine koju je centralni uređaj tražio. Veličina koja se traži definisana je kao REST resurs putem URI. URI ima standardizovanu generičku strukturu pri čemu je vrijednost šeme `coap`, pa je struktura CoAP URI:

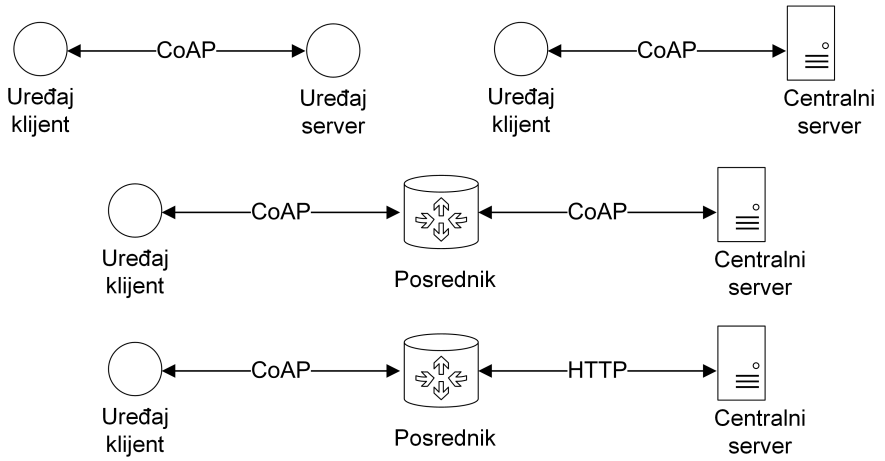
```
coap://host:port/putanja?upit
```

Dijelovi URI se upisuju u opcionalna polja CoAP zaglavlja. Da bi klijent tražio neki resurs od servera mora znati adresu servera, IP ili domensko ime, i putanju do resursa. Radi lakšeg proširenja i uvođenja novih usluga CoAP ima mehanizam za otkrivanje usluga. Ako se za vrijednost URI putanje u GET

[4] Podsjećanje da i HTTP od verzije 2 koristi binarno kodiranje zaglavlja.

zahtjevu navede "/.well-known/" server će vratiti listu svojih dostupnih resursa. Sa URI parametrom `upit` može se suziti lista na one koji zadovoljavaju taj upit. Ako se žele saznati usluge svih dostupnih CoAP čvorova poruka se može poslati na "All CoAP nodes" *multicast* adresu.

CoAP arhitektura omogućava i druga povezivanja i čvorove osim navedenog. Krajnje tačke, CoAP klijent i CoAP server, mogu biti dva krajnja (IoT) uređaja ili jedan krajnji uređaj i jedan centralni uređaj. Moguće je povezivanje preko posrednika (*proxy*). Posrednik može prihvatati i prosljeđivati CoAP zahtjeve i odgovore ili može raditi pretvaranje CoAP zahtjeva i odgovora u HTTP zahtjeve i odgovore i obratno. Zbog sličnosti CoAP i HTTP ovo prevođenje je prilično jednostavno. Posrednici mogu smanjiti opterećenje na mrežu i podržati uštedu energije čvorovima, koji mogu biti neaktivni dok posrednici privremeno pohranjuju (*cache*) poruke za njih. Posrednici mogu, ali ne moraju biti, na granici IoT mreže ograničenih resursa koja koristi CoAP i povezivati je sa Internet *cloud* serverima putem HTTP. Različite vrste povezivanja CoAP prikazane su na slici 4.14.



Slika 4.14: CoAP arhitektura

Pošto CoAP koristi UDP, ne može se osloniti na pouzdanost transportnog sloja. Iz tog razloga postoji tip CoAP poruka koje su potvrdive (*confirmable*), što znači da zahtijevaju potvrdu o prijemu od primaoca. Ovakva vrsta poruka se koristi kada je potrebno ostvariti pouzdanost prenosa. Za poruke za koje pouzdanost nije neophodna koristi se tip nepotvrdive poruke. Ovakav tip pogodan je za poruke kojim se u pravilnim vremenskim intervalima šalje vrijednost neke veličine. Nepristizanje neke od ovih poruke ne predstavlja problem. Pored ova dva tipa poruka postoje još dva: potvrde i *reset* poruke. Potvrde potvrđuju prijem potvrdive poruke. *Reset* poruke potvrđuju da su

dobili poruku (potvrdivu ili ne), ali da na nju ne mogu odgovoriti, iz nekog razloga.

Tip poruke odvojen je od vrste poruke, koja može biti zahtjev ili odgovor. Zahtjev može biti poslan kao potvrdiva ili nepotvrdiva poruka. Ako je zahtjev poslan kao nepotvrdiva poruka onda se i odgovor šalje kao nepotvrdiva poruka, mada server može odgovoriti i potvrdivom porukom. Ako je zahtjev poslan kao potvrdiva poruka moguće su dvije situacije. Ako server može odmah odgovoriti onda se odgovor šalje kao poruka tipa potvrda. Na ovaj način se objedinjuju (*piggybacking*) potvrda i odgovor. Ako server ne može odgovoriti odmah onda šalje poruku tipa potvrda, ali bez odgovora. Time se klijentu signalizira da je zahtjev stigao i da ga ne mora ponovo slati. Kad server ima spreman odgovor šalje ga kao potvrdivu poruku, čiji prijem onda klijent mora potvrditi.

CoAP pošiljalac čeka na potvrdu dok ne istekne vrijeme čekanja, a onda je šalje ponovo dok ne dobije potvrdu ili pošalje maksimalan dozvoljen broj puta. Vrijeme čekanja je inicijalno slučajan, ne obavezno cijeli, broj sekundi unutar ograničenja izvedbe. Sa svakim isticanjem vremena čekanja ono se duplira. Ako se nakon maksimalnog broja pokušaja ne dobije potvrda ili ako se dobije *reset* poruka slanje se obustavlja i aplikacija obavještava o neuspjehu slanja. Povezivanje potvrdivih poruka sa porukama potvrda ili *reset* porukama radi se na osnovu ID poruke koja je dio obaveznog CoAP zaglavlja. Ovaj ID poruka omogućava i prepoznavanje i eliminaciju duplih poruka.

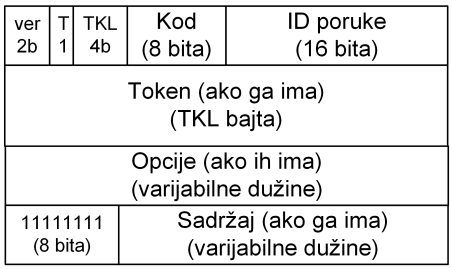
CoAP poruke, zahtjev i odgovor, sastoje se od metode ili koda odgovora. U zavisnosti od toga da li se radi o zahtjevu ili odgovoru, te koja je metoda ili kod odgovora, poruka može imati dodatne informacije u opcijama zaglavlja i/ili sadržaj. Četiri metode zahtjeva koje CoAP podržava, GET, POST, PUT i DELETE, su konceptualno slične HTTP metodama, ali dovoljno različite da je za pisanje aplikacija koje koriste CoAP neophodno konsultovati specifikaciju iz RFC. CoAP kod odgovora iskazuje da li je server razumio zahtjev i da li može odgovoriti na njega. Postoje tri klase odgovora: uspjeh, greška klijenta i greška servera, sa značenjima sličnim kao i kod HTTP. Neki od odgovora iz klase uspješnih su:

- `Content` - koji u sadržaju poruke vreća traženi resurs u GET zahtjevu. Odgovor je sličan 200 OK HTTP odgovoru.
- `Created` - koji u sadržaju poruke, ako ga ima, vreća rezultat akcije iz POST ili PUT zahtjeva.
- `Deleted` - koji u sadržaju poruke, ako ga ima, vreća rezultat akcije iz DELETE zahtjeva.
- `Changed` - koji u sadržaju poruke, ako ga ima, vreća rezultat akcije iz POST ili PUT zahtjeva.

Odgovori greške klijenta su slične kao i kod HTTP i uključuju `Not found` i `Method Not Allowed`. Slično je i sa greškama servera. Odgovori se povezuju sa zahtjevima na osnovu polja token u CoAP zaglavlju čiju vrijednost generiše klijent. Ovo polje je različito od polja ID poruke koje povezuje potvrde sa porukama.

Da bi se smanjilo zaglavlje, metod zahtjeva i kod odgovora su binarno kodirani u osmobitnom polju koje se naziva kod. Prva tri bita, brojevi od 0 do 7, označavaju klasu, a preostalih pet, brojevi 0-31 detalje. Četiri definisane klase su: zahtjev (0), uspješan odgovor (2), odgovor sa greškom klijenta (4) i odgovor sa greškom servera (5). Pet bita detalja kodiraju metodu za zahtjeve, a kod odgovora za odgovore. Brojevi koji odgovaraju definisanim metodama zahtjeva su 1 za GET, 2 za POST, 3 za PUT i 4 za DELETE. Cjelokupno polje kod za GET zahtjev je 00000001 i obično se piše kao 0.01. Slično se pišu drugi zahtjevi. Za svaku klasu odgovora definisani su pet bitni kodovi detalja. Svi kodovi navedeni su u RFC 7252. Primjeri dva pomenuta koda odgovora su 2.05 za `Content` i 4.04 za `Not found`. Poseban kod 0.00 označava praznu poruku.

Izgled CoAP poruke sa obaveznim, prvim, redom i opcionalnim zaglavljem, te sadržajem prikazan je na slici 4.15.



Slika 4.15: Format CoAP poruke

- Ver - označava verziju i za sada je jedina vrijednost 1 (01 binarno).
- T - označava tip poruke. Korištene vrijednosti su 0 za Potvrdivu, 1 za Nepotvrdivu, 2 za Potvrdu i 3 za Reset.
- TKL - označava dužinu tokena, dozvoljene vrijednosti su od 0 do 8.
- Kod - je prethodno opisani kod koji definiše metod zahtjeva ili kod odgovora.
- ID poruke - prethodno opisani identifikator poruke koji povezuje potvrdive poruke sa odgovarajućim potvrdama ili *reset* porukama.
- Token - opcionalo polje koje, kako je ranije objašnjeno povezuje zahtjeve sa odgovorima.
- Opcije - opcionalno polje, u koje se upisuju meta podaci o zahtjevu, poput dijelova URI, ili odgovoru, poput formata polja sa sadržajem. Pravila formatiranja ovog polja navedena su u RFC.
- Sadržaj - opcionalno polje u kom se nalazi sadržaj poruke, zahtjeva ili odgovora, ako postoji. Ovo polje je od opcija odvojeno sa jednim bajtom jedinica.

Ako je potrebno, sigurnost CoAP ostvaruje se upotrebom DTLS, koji ima sličnu funkciju kao TLS ali se koristi sa UDP. URI šema kada se koristi DTLS je `coaps`. Standardni brojevi portova za CoAP i CoAP po DTLS su 5683 i 5684, respektivno. Naravno, u URI polju `port` se može definisati i druga vrijednost.

Postoji veliki broj CoAP izvedbi, kako otvorenog koda tako i zatvorenog. CoAP se zasniva na dobro poznatim web tehnologijama, ali je optimiziran za uređaje i mreže sa ograničenim resursima. Prednost CoAP je što se razvija kroz IETF pa je usklađen sa ostalim IETF protokolima koji su ranije pomenuti, kao i sa onim koji nisu, zbog ograničenog prostora. Potencijalni nedostatak CoAP za IoT je povezivanje jedan na jedan koje nameće klijent/server arhitektura. Za IoT je često pogodnije povezivanje više na jedan i/ili jedan na više koje omogućava protokol obrađen u nastavku.

4.7.3 MQTT

Kako je u uvodu poglavlja rečeno, tradicionalni način dostave poruka između jednog pošiljaoca i jednog primaoca nije najpogodniji za IoT. IoT može imati veliki broj pošiljalaca, recimo senzora, ili veliki broj primalaca, recimo aktuatora, koji svi šalju jednom ili primaju od jednog čvora. Model razmjene poruka koji ovo omogućava je objavi/pretplati (*publish/subscribe*). Kod ovog modela pošiljaoci su odvojeni od primalaca. Sva komunikacija se odvija preko posrednika, servera, koji se obično naziva broker[5]. Pošiljaoci šalju poruke brokeru. Primaoci preuzimaju poruke od brokera. Za razliku od tradicionalnog modela zahtjev/odgovor pošiljaoci ne moraju čekati na odgovor primalaca, nego samo šalju podatke. Primalac i pošiljalac ne moraju bit aktivni istovremeno, ako broker podržava čuvanje poruka. Ovaj model omogućava primaocima da izaberu koje poruke će primati, od svih poruka koje broker dobiva. Krajnji čvorovi u ovom modelu, pošiljaoci i primaoci ne moraju jedni drugima znati adrese, sve poruke se šalju brokeru i od njega se traže. Ovo pojednostavljuje adresiranje. Naravno postoje neka otvorena pitanja ovog modela. Jedno je pitanje pouzdanosti isporuke, jer nema veze između pošiljaoca i primaoca kojom bi se potvrđivao uredan prijem. Drugo je skalabilnost, jer sistem ima potencijalno usko grlo, brokera.

MQTT (*Message Queuing Telemetry Transport*) je protokol koji koristi objavi/pretplati model dostave poruka. Protokol je nastao još 1999. godine u saradnji kompanija IBM i Eurotech za potrebe gasnih i naftnih cjevovoda. Pokazalo se da je protokol vrlo pogodan i za savremeni IoT. Organizacija koja danas brine o standardizaciji MQTT je OASIS (*Organization for the Advancement of Structured Information Standards*). Tekuća verzija protokola je 5.0 i detaljno je opisana u OASIS standardu [15]. U nastavku su opisane osnovne karakteristike protokola. Sve nedostajuće informacije mogu se naći u navedenom standardu.

[5] U nedostatku adekvatnog prevoda koristi se engleska riječ koja se odomaćila i kod nas.

MQTT definiše kako klijenti objavljuju poruke i kako se pretplaćuju na njihov prijem. Format sadržaja MQTT poruka nije propisan. Mogu se prenositi bilo kakvi podaci. Bitno je da su oglašavači (pošiljaoci) i pretplatnici (primaoci) usaglašeni i da razumiju format. U praksi se najviše koriste JSON i binarni zapisi podataka. Prema protokolu maksimalna veličina sadržaja MQTT poruke je 256 MB. U praksi broker može, i uglavnom to i radi, ograničiti veličinu sadržaja na mnogo manju vrijednost poput 128 ili 256 KB. Ovo je potrebno znati prilikom povezivanja klijenata na broker.

MQTT koristi TCP kao transportni protokol. To povećava potrebu za resursima na uređajima, ali povećava pouzdanost prenosa. Pošto je MQTT namijenjen za IoT gdje se ispadi, koje ni TCP ne može prevaziči, mogu očekivati, u protokol su ugrađeni dodatni mehanizmi za ostvarivanje pouzdanosti.

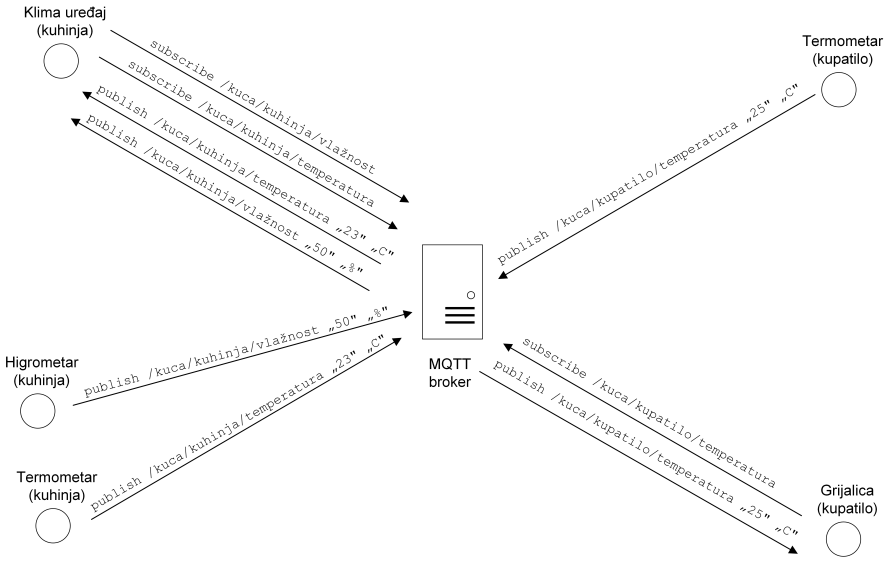
MQTT radi na slijedeći način. MQTT oglašavači šalju brokeru poruke uz čiji sadržaj navode temu. Tema govori na šta se poruka odnosi, a sadržaj obično sadrži vrijednost. Tema može biti "temperatura", a sadržaj "23". Nazivi tema su uobičajeno hijerarhijski, pri čemu su nivoi hijerahije razdvojeni znakom "/". Primjeri tema su: "kuca/kuhinja/temperatura", "kuca/kuhinja/vlaznost" ili "kuca/kupatilo/temperatura". Značenje i struktura hijerahije treba biti jednoznačno za neki broker i poznato svim oglašavačima i pretplatnicima. To je potencijalni nedostatak MQTT. Oni koji objavljuju i primaju poruke trebaju unaprijed znati teme da bi na njih mogli objavljivati i čitati objavljene poruke.

MQTT pretplatnici šalju poruke kojim se pretplaćuju na teme. Jednom porukom moguće je pretplatiti se na više tema upotrebom posebnih znakova "+" i "*" (*wildcards*). Znak "+" zamjenjuje sve vrijednosti na jednom nivou hijerahije. Pretplata na temu "kuca/+/temperatura" će izvršiti pretplatu na temperaturu u svim prostorijama u kući (ako je to značenje drugog nivoa hijerahije). Znak "*" zamjenjuje sve vrijednosti na svim nivoima hijerahije. Pretplata na temu "kuca/*" će izvršiti pretplatu na sve veličine u svim prostorijama u kući.

Kada broker dobije objavu od oglašavača, on ja obrađuje. Na osnovu teme objave i pretplatnika na temu odlučuje kome treba poslati i šalje im objavu. Ako nema pretplatnika na temu objave, objava se ne čuva. Oglašavač može u objavi navesti da želi da objava bude zadržana na brokeru. To znači da će ta objava biti poslana kasnije svim novim pretplatnicima koji se prijave na temu objave čim se prijave. Neće morati čekati na novu objavu. Ova mogućnost je vrlo korisna u IoT okruženju jer omogućava krajnjim uređajima da šalju i primaju vrijednosti u trenucima kad su aktivni, te da mogu preći u stanje neaktivnosti bez straha da će propustiti objave.

Potrebno je naglasiti da jedan uređaj može u isto vrijeme biti i oglašavač i pretplatnik. Opisani način rada MQTT prikazan je na slici 4.16.

MQTT ima ugrađene mehanizme za pouzdanost na aplikativnom sloju, bez obzira na upotrebu TCP. MQTT poruke se potvrđuju. Potvrde šalju čvorovi koji direktno dobiju poruku. Potvrda objave nije od pretplatnika za oglašavača. Objavu potvrđuje broker oglašavaču. Kad broker pošalje pret-



Slika 4.16: MQTT rad

platniku objavu na temu na koju se pretplatio pretplatnik potvrđuje prijem objave. Oglašavač ne zna da li je i koji pretplatnik dobio objavu.

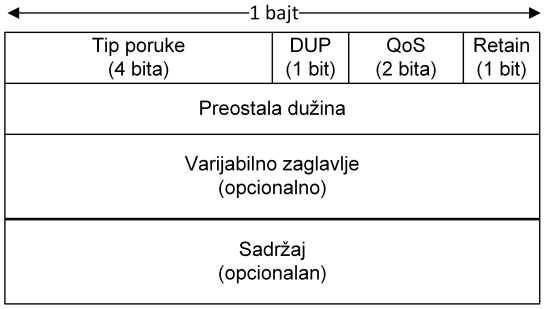
MQTT pruža podršku za QoS i nudi mogućnost izbora tri nivoa.

- QoS-0 - nema potvrđivanja poruka i garancije isporuke;
- QoS-1 - garantovana isporuka prijemniku bar jednom, ali moguće i više puta;
- QoS-2 - garantovana isporuka poruke tačno jednom.

Viši nivo QoS zahtjeva više saobraćaja i resursa na čvorovima. Nivo QoS definiše pošiljalac poruke.

Obavezno zaglavlje MQTT ima samo dva bajta, opcionalna zaglavlja su promjenljive dužine koja zavisi od tipa poruke, isto važi i za sadržaj paketa. Format MQTT paketa prikazan je na 4.17.

- Tip poruke - 4-bita koja označavaju tip MQTT poruke u paketu. Postoji 14 tipova poruka, od kojih su dvije pomenute PUBLISH i SUBSCRIBE.
- DUP - 1-bit koji označava da je poruka bila poslana ranije ali za nju pošiljalac nije dobio potvrdu. Koristi se samo za PUBLISH poruke, a za ostale je vrijednost 0.
- QoS - 2-bita označava izabrani nivo QoS. Koristi se samo za PUBLISH poruke.
- RETAIN - 1-bit koji govori brokeru da zadrži poruku i isporuči je klijentima koji se kasnije pretplate na temu iz poruke. Koristi se samo za PUBLISH poruke, a za ostale je vrijednost 0.



Slika 4.17: Format MQTT poruke

- Preostala dužina - 8-bita kojim se definiše broj bajta paketa nakon ovog zaglavlja.
- Varijabilno zaglavlje - opcionalno polje, promjenljive dužine čija polja i dužina zavise od tipa poruke. U polje ovog zaglavlja se upisuje naziv teme na koju se šalje PUBLISH poruka.
- Sadržaj - opcionalno polje, promjenljive dužine čija polja i dužina zavise od tipa poruke. U sadržaj se može upisati vrijednost koja se objavljuje na temu u sklopu poruke PUBLISH. U sadržaj se upisuje naziv teme na koju se šalje SUBSCRIBE poruka.

Ako je potrebno zaštiti sigurnost MQTT poruka koristi se TLS. Standardni broj porta za MQTT je 1883, a ta MQTT po TLS je 8883.

Kao i za CoAP, postoji veliki broj MQTT izvedbi. Neke su otvorenog koda, a neke su integrisane u proizvođačka rješenja. Jedan od vrlo poznatih i korištenih komunikacionih softvera koji koristi MQTT, ali nema veze sa IoT, je Facebook Messenger. MQTT brokeri mogu biti lokalni serveri ili *cloud* bazirani, poput Google MQTT *bridge*. MQTT je široko rasprostranjen i koristi se u mnogim produkcionim okruženjima. To svjedoči o njegovoj upotrebljivosti i pogodnosti za IoT umrežavanje. Potencijalni nedostatak za uređaje i mreže i mreže koji imaju vrlo ograničene resurse je upotreba TCP, koji traži više resursa nego UDP. Protokol MQTT-SN razvijen iz MQTT ne mora uopšte koristiti TCP/IP ili može koristiti UDP umjesto TCP. Naravno ova promjena ima uticaj na druge razlike u odnosu na MQTT.

* * *

Ovim je završen opis IoT umrežavanja. Ova tema je vrlo aktuelna i stoga još uvijek bez dominantnih protokola. U poglavlju su opisani samo neki od aktivnih protokolima na svim slojevima. Ako se nešto može naučiti iz dosadašnjeg razvoja komunikacija nakon burnog početnog perioda pojavi se skup protokola koji preuzme dominantnu ulogu. Još se ne može reći koji će to protokoli biti,

ali do sada je kombinacija HTTP/TCP/IP/802 (Ethernet ili WiFi) uvijek na kraju izašla kao pobjednik.

5

Umrežavanje podatkovnog centra

Sa porastom broja mrežnih usluga koje su serveri pružali klijentima povećavao se broj servera koji su bili potrebni. U početku su serveri bili smještani u server sobe, namjenske prostorije sa odgovarajućim okruženjem. To okruženje uključuje odgovarajuće kabliranje, adekvatno napajanje, klima uređaje, zaštitu od požara i poplave ... Kada je broj servera porastao toliko da jedna prostorija više nije bila dovoljna, server sobe su prerasle u podatkovne centre (*data center*). Podatkovni centri mogu se protezati na više prostorija u zgradi, cijelu zgradu ili više zgrada. U svakom slučaju to je veliki broj umreženih servera na malom prostoru. Podatkovni centri nastali su tokom perioda komercijalizacije Interneta krajem 90tih godina prošlog vijeka i početkom ovog vijeka. Pojavom *cloud computing* i društvenih mreža podatkovni centri postali su ono što su danas.

Umrežavanje podatkovnog centra ima različite izazove od umrežavanja udaljenih klijenata i servera. Ti izazovi i kako se na njih odgovara u savremenim centrima opisani su u nastavku.

5.1 Savremeni podatkovni centar

Tradicionalni, fizički, serveri su bili računari, posebne namjene. Oni su bili pravljeni da rade neprestano i pouzdano, ali su i dalje imali tradicionalne dijelove. Svaki server je imao procesor, radnu memoriju, trajnu memoriju, tastaturu, miša, monitor i mrežnu konekciju, te druge dijelove, po potrebi. Radi uštede prostora kućišta servera su pravljena sa standardnom širinom tako da je do 20-ak servera, jedan iznad drugog, moglo stati u jedan serverski ormar (*rack*). Da ne bi svaki server morao imati svoj monitor i tastaturu, jer im nisu bili neophodni, svi serveri su bili povezani na jedan uređaj, KVM (*Keyboard, Video, Mouse*) *switch*. Na ovaj KVM *switch* su bili povezani jedna tastatura, miš i monitor koji su se mogli koristi za jedan server u jednom trenutku, što se pokazalo sasvim dovoljnim. Povećavanje potreba pojedinih servera za resursima zadovoljavalo se ugradnjom bržeg procesora, više radne i

trajne memorije. U trenutku kad su potrebe porasle do mjere da ih jedan fizički server više nije mogao zadovoljiti promijenjen je pristup upotrebi serverskih resursa.

Umjesto fizičkih servera, aplikacije su počele da koriste apstraktne logičke servere, koji se nazivaju virtuelnim. Virtuelne mašine, a kasnije i kontejneri, su aplikacijama davale pristup resursima potrebnim za njihov rad. U pozadini su ti resursi mogli biti sa jednog ili više fizičkih servera, Aplikacije toga nisu bile svjesne i nije im ni bilo bitno, dok god su resursi dostupni. To je omogućilo da virtuelni serveri budu, gotovo, proizvoljno mali ili veliki.

Sa virtuelizacijom više nije bilo neophodno da se svi resursi jedne virtuelne mašine nalaze na istom fizičkom serveru. Sada su se virtuelni resursi mogli prikupljati sa različitih fizičkih servera. Više nije bilo neophodno ni da trajna memorija bude dio pojedinog fizičkog servera već se mogla, efikasnije, organizovati kao posebna komponenta u ormaru dostupna svim serverima. Serveri i memorija nisu morali biti u istom ormaru, već bilo gdje u podatkovnom centru. Ovaj proces se naziva disagregacija.

Fizički serveri su se takođe promijenili, Umjesto tradicionalnih servera sa svim elementima prešlo se na *blade* servere. Ovi serveri imaju samo procesor. radnu memoriju i integrisan mrežni kontroler, pa su, fizički, mnogo manji od tradicionalnih. Ubacuju se u posebna *blade* kućišta (*enclosure*), koja se onda ubacuju u odgovarajuće serverske ormare (*chasis*), koji obezbjeđuju, napajanje, hlađenje i druge komponente koje su potrebne osim trajne memorije. Trajna memorija se fizički nalazi odvojeno u serverskom ormaru, istom ili drugom, uglavnom u obliku *hard* diskova koji su kao prostor za pohranu dostupni kroz različite oblike organizovanja, od starijeg RAID do novijih kao što su NAS (*network attached storage*) ili SAN (*storage area networks*). Broj *blade* servera u jednom kućištu može biti veći od stotinu. Savremeni podatkovni centri imaju po više od 100.000 fizičkih servera. Na svakom od takvih servera danas je moguće pokrenuti 20-ak virtuelnih mašina. Iz navedenog slijedi da savremeni podatkovni centar može imati preko 2.000.000, virtuelnih, servera koje treba međusobno umrežiti.

5.2 Posebnosti umrežavanje podatkovnog centra

Tradicionalne računarske mreže i protokoli koji ih omogućavaju nastali su vrijeme kada je propusnost i pouzdanost veza bila mnogo manja nego danas. Količina podataka koja se prenosi po mrežama takođe je bila mnogo manja. Većina komunikacija bila je između klijenata i servera. Zbog toga je i način rada bio optimiziran za klijent/server model, gdje je klijent mogao biti na velikoj udaljenosti od servera.

Podatkovni centri se ne uklapaju ovaj model. Propusnost i pouzdanost veza unutar podatkovnog centra su velike. Udaljenosti su male. Sa druge strane razdvajanje trajne memorije od servera znači da server podacima pristupa preko računarske mreže podatkovnog centra, a ne više direktno. Ovaj

pristup mora biti dovoljno brz da ne ometa rad servera. Pored toga ovim se povećava i promet po mreži. Karakteristike saobraćaja su drugačije.

Mrežni saobraćaj u podatkovnom centru razlikuje se od tradicionalnog klijent server saobraćaja kakav je bio u server sobama. Stari saobraćaj je uglavnom bio između vanjskih klijenata i servera na lokaciji. Klijent je slao zahtjev jednom serveru i taj server je odgovarao klijentu. Ovakav saobraćaj se naziva "sjever-jug", jer ide gore dole, ka serverima i od njih. Nije bilo mnogo saobraćaja između servera, jer su pojedini serveri sami mogli odgovoriti na zahtjeve klijenata.

U savremenim podatkovnim centrima dominira unutrašnji saobraćaj. U velikim podatkovnim centrima unutrašnjeg saobraćaja je 10 puta više. Kako je glavni inicijator rada podatkovnog centra komunikacija sa klijentima, znači da se za svaki bajt komunikacije sa vanjskim svijetom generiše 10 bajta unutrašnjeg saobraćaja. Unutrašnji saobraćaj naziva se, "istok-zapad", odnosno lijevo desno, prema uobičajenom crtanju mrežnih arhitektura. Jedan od razloga za ovakav saobraćaj je disagregacija u kojoj server mora komunicirati preko mreže sa trajnom memorijom. Drugi razlog je u prirodi savremenih aplikacija. Kad korisnik društvene mreže pristupa svom korisničkom računu, server treba da vrati web stranicu sa svim njenim elementima. Ovi elementi su jako različiti. Neki su objave, neki slike, neki video, i odnose se na različite korisnike. Ovi podaci se uglavnom nalaze na različitim serverima u podatkovnom centru. Za njihovo prikupljanje i prikazivanje potrebna je uglavnom dodatna obrada, odnosno procesiranje ne nekom serveru. Sve ove razmjene se odvijaju preko mreže podatkovnog centra.

Statistike pokazuju da od svih mrežnih tokova manji broj njih zauzima većinu mrežne propusnosti [4]. Ovakvi tokovi se nazivaju "slon" (*elephant*) tokovi. Prenos videa, koji danas čini više od pola Internet saobraćaja, je primjer takvog toka. Slon tokovi predstavljaju poteškoću u podatkovnom centru jer mogu zauzeti dostupni kapacitet i onemogućiti ostalim manjim "miš" (*mouse*) tokovima da budu poslani.

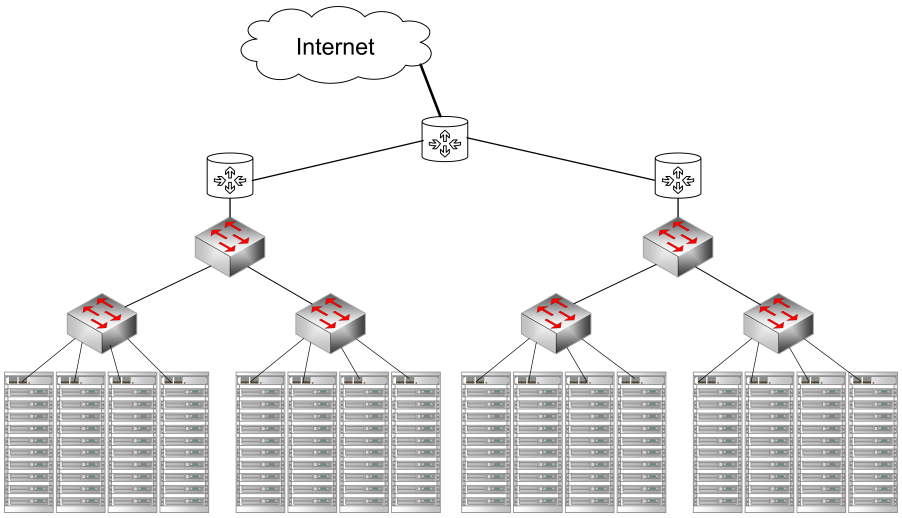
Podatkovni centar je veliki, ali homogen i pod jednom administrativnom kontrolom. Ovo znači da se koriste isti protokoli u cijelom podatkovnom centru. Prema tome nema potrebe da postoji kompatibilnost unazad i prilagođavanje za različite protokole. Iz tog razloga moguće je koristiti nove protokole namjenski razvijene za podatkovne centre. Ovi protokoli su optimizirani da minimiziraju kašnjenje koje bi trebalo biti do nekoliko milisekundi.

Sve navedeno nameće potrebu drugačijeg pristupa umrežavanju podatkovnog centra od tradicionalnog umrežavanja.

5.3 Mrežne topologije

Serverski ormari (*rack*) danas obično imaju po 40-ak fizičkih servera. Svi serveri unutar ormara su povezani sa *switch*-em, koji se nalazi u istom ormaru, obično na vrhu, Ethernet vezama brzine 40Gb/s. Ovaj *switch* se obično naziva

TOR (*Top Of Rack*) *switch*. TOR *switch*-evi predstavljaju pristupnu (*access*) ili, drugi naziv, rubnu (*edge*) mrežu. Oni su dalje povezani na *switch*-eve iznad njih koji predstavljaju agregacionu (ili distributivnu) mrežu. Ti *switch*-evi su povezani dalje na *switch*-eve u jezgrenoj (*core*) mreži. Ova klasična topologija umrežavanja podatkovnog centra je prikazana na slici 5.1.



Slika 5.1: Klasična topologija umrežavanja podatkovnog centra

Ova topologija je striktno hijerarhijska. Nema višestrukih veza između elemenata. Komunikacija između dva čvora, u ovom slučaju servera, odvija se od jednog servera ka korijenu stabla do pronalaska *switch*-a do kog vodi putanja po stablu od odredišnog servera. Pronalazak putanje, uspostavljanje razapinjućeg stabla (*spanning tree*) i prosljeđivanje u ovakvoj mreži su jednostavni. Ovakva topologija odgovara podatkovnim centrima sa dominantno "sjever-jug" saobraćajem. Ako sve grane stabla, nezavisno od nivoa hijerarhije, imaju istu debljinu (propusnost) to se onda naziva mršavo stablo (*skinny tree*).

Navedena topologija ima nekoliko nedostataka koji je čine nepogodnom za upotrebu u savremenim podatkovnim centrima. Jednaka propusnost svih grana znači da u jednom trenutku saobraćaj samo jednog TOR *switch*-a može biti proslijeđen od agregacione do jezgrene mreže. Ovo nije odgovarajuće čak ni za pretežno klijent-server saobraćaj, jer u tom slučaju propusnost grane ograničava broj zahtjeva koje podatkovni centar može prihvatiti ili na koje može odgovoriti. Ako serverski resursi mogu odgovoriti većem broju klijenata, mreža ne bi trebala biti usko grlo. Odnos zbira svih propusnosti veza od servera i ukupne propusnosti topologije u najužem dijelu, obično korijenu (prema vani), naziva se *over-subsription*. Ako je, kao na slici 5.1, u jednom ormaru 10 servera, a po četiri TOR *switch*-a povezana na jedan agregacioni

switch, te dva agregaciona *switch*-a povezana na jedan jezgreni *switch*, pri čemu su propusnosti svih veza jednake, onda je *over-subsription* 80:1 (2 x 4 x 10 servera povezano preko jednog jezgrenog *switch*-a). To znači da svaki server, u slučaju da svi koriste mrežu istovremeno, ima u prosjeku samo oko 1,2% mrežne propusnosti svoje veza sa TOR *switch*-em.

Sa aspekta mrežne propusnosti, idealno bi bilo da svi serveri imaju punu propusnost, da je *over-subsription* 1:1, odnosno da nema *over-subsription*. To bi značilo da mrežna topologija ni u jednom trenutku nije usko grlo. To se može postići tako što se propusnost svih *uplink* veza, i portova, na svakom od *switch*-eva poveća da bude jednaka zbiru propusnosti svih *downlink* veza na tom *switch*-u. Pri tome prespojna struktura (*fabric*) *switch*-a mora podržavati takvu propusnost. Ovakvo rješenje drastično povećava cijenu mreže i nije racionalno. Kod ovakvog rješenja će u situacijama kada svi serveri ne koriste istovremeno svoj puni kapacitet veze mrežni resursi biti neiskorišteni. Pravu mjeru *over-subsription* određuje svaki podatkovni centar na osnovu statističkih podataka o iskorištenosti mreže.

Topologija stabla u kom su grane koje su bliže korijenu deblje od hrana bliže listovima naziva se debelo stablo (*fat tree*). Ovakva topologija je i dalje jednostavna za pronalaženje razapinjućeg stabla. Nedostatak joj je visoka cijena, te ograničena skalabilnost. Povećavanjem broj servera i propusnosti njihovih veza povećavaju se zahtjevi na *switch*-eve koji su bliže korijenu i potrebna je njihova zamjena da bi mreža i dalje ostvarivala željenu propusnost. Već za sadašnje veze servera od 40Gb/s i broj servera od desetina hiljada u podatkovnom centru to je teško ostvariti. Ova topologija nepovoljna je i za, uobičajeni, saobraćaj u podatkovnom centru koji se sastoji od manjeg broja velikih i većeg broja malih tokova. Jedan veliki tok može potrošiti većinu *uplink* propusnosti na nekom od *switch*-eva i onemogućiti malim tokovima da koriste vezu. To je poseban problem ako su mali tokovi vremenski kritični. što je, danas, često slučaj kod prenosa multimedijalnih sadržaja.

Navedeni pristup smanjivanju *over-subscription* u kom se to ostvaruje nabavkom bolje, i skuplje, opreme, ovdje *switch*-eva naziva se skaliranje naviše *scale up*. Slično kao i kod servera u podatkovnom centru pokazalo se da se zadovoljavanje potreba povećavanjem kapaciteta pojedinih uređaja ne skalira dobro. Kod servera je to rješeno upotrebom više manjih servera i raspoređivanjem opterećenja među njima. Slična ideja primijenjena je kod umrežavanje podatkovnih centara. Umjesto povećavanje kapaciteta *switch*-eva i njihovih veza, povećan je broj *switch*-eva istih karakteristika, i broj veza među njima. Ovim se ostvaruje isti efekat povećanja propusnosti mreže, uz veću skalabilnost i po nižoj cijeni. Ovaj pristup naziva se skaliranje prema vani (*scale out*).

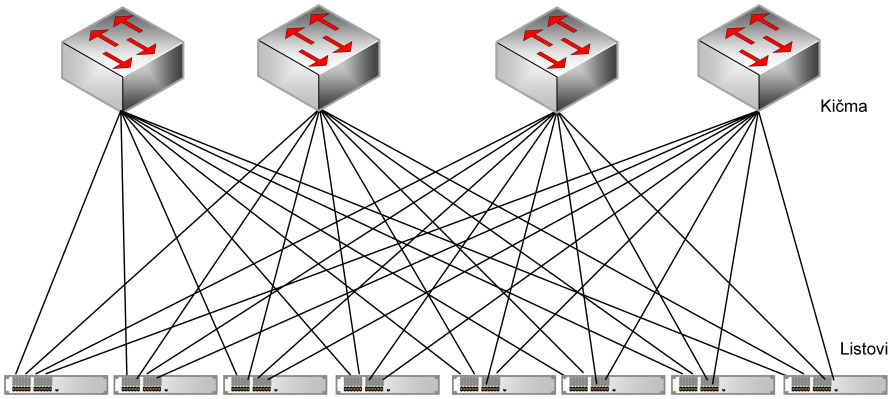
Moguće su različite izvedbe ovog pristupa, ali se najboljim pokazala takozvana kičma i list (*spine and leaf*) arhitektura [54]. Ova arhitektura je zapravo tro-stepena preklopljena (*tree stage folded*) Clos mreža. Često se za ovu strukturu koristi i naziv *fat tree*, jer ostvaruje isti efekat propusnosti kao da su grane stabla bliže korijenu deblje od grana bliže listovima. Ova dvoslojna arhitektura

zamjenjuje klasičnu troslojnu koja se sastojala od pristupnog (rubnog), agregacionog (distributivnog) i jezgrenog sloja. Kod ove arhitekture svaki čvor list povezan je sa svim čvorovima kičme sa vezama jednake propusnosti. Na taj način se osigurava da su svi krajnji krajnji čvorovi maksimalno udaljeni jedan skok (*switch*). Time se minimizira kašnjenje i vjerovatnoća pojave uskog grla. U ovakvoj arhitekturi između bilo koje dvije krajnje tačke mreže je jednaka prosječna propusnost i kašnjenje jer je mreža potpuno simetrična. Svi serveri u podatkovnom centru imaju utisak da su povezani na jedan jedini veliki *switch* koji ih sve povezuje. Iz ovog razloga proizvođači danas ovu arhitekturu nazivaju i reklamiraju/prodaju kao *fabric*, Ethernet *fabric*, *data center fabric* ili ponekad i *switch fabric* (što je malo zbunjujuće, jer se ne razlikuje od prespojne strukture jednog *switch*-a). Potrebno je napomenuti činjenicu da veliki broj servera koji su povezani u istu lokalnu mrežu može predstavljati problem radi rasta tabela sa MAC adresama, kao i broja potrebnih VLAN-ova. Velike MAC tabele trebaju biti podržane resursima u *switch*-evima, što im povećava kompleksnost i cijenu. VLAN brojevi se upisuju u polje u 802.1Q oznake dužine 12 bita. To znači da ih ukupno može biti 4096. Ovaj broj može biti nedovoljan za savremene podatkovne centre. Jedno od mogućih rješenja za ovo pitanje je obrađeno u poglavlju o SDN.

U slučaju povećanja broja servera povećava se broj serverskih ormara sa TOR *switch*-evima. TOR *switch*-evi su listovi i oni se povezuju na sve kičmene *switch*-eve, čiji se broj također povećava po potrebi. Kako je već rečeno, svi *switch*-evi mogu biti isti, te također mogu biti relativno jednostavni i nemodularni. Takvi *switch*-evi imaju nižu cijenu, manju potrošnju energije i manje kašnjenje. Kompanije sa velikim podatkovnim centrima, poput Google i Facebook, ovakve *switch*-eve dizajniraju sami i nabavljaju naveliko direktno od proizvođača *chipset*-ova i ostvaruju velike uštede u odnosu na nabavku od velikih prodavača mrežne opreme. Primjer je, danas javno dostupan za nabavku i sa svim specifikacijama, Facebook *switch* Wedge 100 sa Facebook *switch*-ing softverom FBOSS [6].

Još jedna prednost ove arhitekture je redundantnost veza. Za razliku od klasične, striktno hijerarhijske, topologije, ovdje prekid neke od grana ili ispad nekog od *switch*-eva ne znači prekid za sve čvorove koji su povezani preko te grane ili *switch*-a. Svaki *switch* ima više od jedne veze sa drugim *switch*-evima, pa postoje alternativne putanje na koje se saobraćaj može preusmjeriti. Primjer kičma i list arhitekture dat je na slici 5.2.

Ovakva topologija pogodna je i za saobraćaj "istok-zapad", kakav dominira u savremenim podatkovnim centrima. Postoji mnogo više poprečnih veza koje mogu biti iskorištene za saobraćaj između servera unutar podatkovnog centra. Nije neophodno da saobraćaj teče visoko uz hijerarhiju i bori se za propusnost sa saobraćajem "sjever-jug". Međutim, veći broj mogućih putanja između čvorova komplikuje proces prolaženja razapinjućeg stabla. Postojanje više putanja može predstavljati problem pronalaska najbolje, u klasičnom umrežavanju. U savremenom podatkovnom centru ovo može biti iskorišteno da se istovremeno koristi više putanja i saobraćaj preusmjerava u skladu



Slika 5.2: Kičma i list arhitektura umrežavanja podatkovnog centra

sa opterećenošću pojedinih veza. To je posebno pogodno za raspoređivanje, pomenutih, velikih i malih tokova. Postojanje više putanja donekle komplikuje prosljeđivanje, pa su razvijeni posebni algoritmi koji pogodni za takve slučajeve. Ovo pitanje dodatno je razmotreno kasnije u tekstu.

Proširivost i brži rast, ili izgradnja, podatkovnih centara danas se postižu modularnim dizajnom. Veća količina računarskih, i mrežnih, resursa se grupiše, skupa sa svom pratećom infrastrukturom. Ovaj skup resursa je jedinica povećanja resursa. Proširivanje podatkovnog centra se vrši dodavanjem potrebnog broja ovih jedinica. Jedan oblik ovih jedinica su takozvani kontejneri, nazvani tako jer su standardne veličine, i pakovanja, kao i kontejneri za slanje robe brodom. Ovi kontejneri sadrže nekoliko desetina serverskih ormara sa serverima, trajnom memorijom, napajanjem, hlađenjem, kabliranjem i umrežavanjem. Oni imaju, relativno, standardizovane mrežne interfejse putem kojih se uvezuju u podatkovni centar. Opštiji oblik modula podatkovnog centra je PoD (*Point of Delivery*). Slično kao kontejner ovo je skup uvezanih servera sa pratećom infrastrukturom, ali koji ne mora biti u standardnom kućištu za prevoz tereta. Veličina *pod*-a može varirati. Facebook koristi *pod*-ove sa 48 serverskih ormara [7]. Pošto su ovi moduli, kontejneri i *pod*-ovi, standardizovani, oni nisu predviđeni za veće održavanje i ažuriranje hardvera. Oni odrade svoj životni vijek od nekoliko godina i nakon toga se zamjenjuju.

Uvezivanje ovih modula predstavlja poseban izazov. Potrebna je jezgrena mrežna infrastruktura koja povezuje velike module međusobno i sa vanjskim svijetom. Ova infrastruktura mora imati veliku propusnost i pouzdanost. Uglavnom se gradi korištenjem arhitekture kičma i list, gdje su listovi sada moduli, kontejneri ili *pod*-ovi. Ovdje se javlja potreba za upotrebom i fiber optičkih veza i *switch*-eva. Dobar pregled topologija podatkovnog centra dat je u [31].

5.4 Prosljeđivanje

Posebno pitanje dizajna podatkovnog centra je na kom sloju protokola drugom ili trećem ili nekoj kombinaciji će se vršiti prosljeđivanje paketa. Svaki od pristupa ima svojih prednosti i nedostataka.

5.4.1 Drugi sloj

Klasična striktno hijerarhijska topologija umrežavanja podatkovnog centra sastojala se uglavnom od *switch*-eva koji su radili prosljeđivanje na drugom sloju. *Switch*-evi koji su mogli prosljeđivati na trećem sloju nisu bili uobičajeni. Koristio se STP (*Spanning Tree Protocol*) za pronalazak putanja bez petlji. Rastom podatkovnih centara ovakav pristup postao je ograničavajući. čak i uz uvođenje RSTP (*Rapid* STP). Horizontalno širenje podatkovnih centara sa novim topologijama nije pogodovalo ovom načinu rada. sve veća uniformnost saobraćaja u podatkovnom centru značila je da je gotovo sav saobraćaj IP. Pojavile su se verzije protokola koje su radile agregaciju veza i time omogućile prosljeđivanje na drugom sloju koristeći mrežne putanje. Međutim broj veza koje su se mogle agregirati bio je vrlo ograničen. Jedan od najvećih problema su bili ispadi uređaja koji su izazivali velike poteškoće u pronalaženju novih putanja. Pojavili su se novi protokoli, navedeni kasnije, koji su poboljšali skalabilnost.

5.4.2 Hibrid drugog i trećeg sloja

Radi smanjivanja uticaja ispada uređaja koji prosljeđuju pakete i povećavanja skalabilnosti podatkovnih centara u jezgrenom i/ili distribucionom dijelu mreže, uvedena je kombinacija prosljeđivanja na drugom i trećem sloju. Time je umjesto jednog domena drugog sloja podatkovni centar bio podijeljen na više manjih. Ovo je omogućilo skaliranje, ali naviše (*scale up*). Kako je ranije objašnjeno ovakvo skaliranje povećava kompleksnost mreže u podatkovnom centru. Domeni drugog sloja, u pristupnoj i distribucionoj mreži, su zadržani jer se time broj potrebnih IP podmreža zadržao unutar praktično izvodljivih granica, Time je, takođe, očuvana mogućnost lakog premještanja virtuelnih mašina unutar domena drugog sloja bez potrebe za promjenom IP adrese, kao mogućnost raspoređivanja opterećenja među serverima koje se oslanaja na direktnu povezanost na drugom sloju. Nije zanemariva ni činjenica da *switch*-evi koji rade prosljeđivanje na trećem sloju imaju višu cijenu od onih koji rade na drugom.

5.4.3 Treći sloj

Danas se sve češće koristi rutiranje, prosljeđivanje na trećem sloju, i u pristupnom sloju podatkovnih centara. Na ovaj način se povećava stabilnost i

skalabilnost mreža usljed smanjivanja veličine domena drugog sloja. Sa porastom broja servera u podatkovnim centrima na desetine hiljada ovakav pristup postaje sve potrebniji. Ovakve mreže su jednostavnije, stabilnije i skalabilnije što je postalo važnije od drugih karakteristika koje donosi prosljeđivanje na drugom sloju.

5.5 Posebni protokoli

Radi posebnosti umrežavanja podatkovnih centara razvijeni su protokoli koji su bolje usklađeni sa tim posebnostima od standardnih mrežnih protokola. Ranije je pomenuto da homogenost podatkovnih centara i njihova administracija od jedne organizacije omogućava korištenje ograničene grupe posebnih protokola bez potrebe za kompatibilnošću unazad. Postoji veći broj novih protokola koji se koriste u podatkovnim centrima. Izbor protokola zavisi dijelom od korištene topologije, a može zavisiti i od korištene mrežne opreme. Neki proizvođači preporučuju neke protokole ili čak imaju svoje protokole koji rade sa njihovom opremom. U nastavku su navedeni neki od protokola sa naglaskom na njihovu namjenu da bi se ukazalo koje posebnosti novi protokoli rješavaju.

5.5.1 Putanje

U podatkovnim centrima postoji veći broj putanja između čvorova. Standardno umrežavanje podrazumijeva uspostavljanje razapinjućeg stabla (*spanning tree*) između svih čvorova radi sprečavanja pravljenja petlji. Nedostatak ovog pristupa, u podatkovnim centrima, je što se višestruke putanje ne mogu koristiti. Korištenje višestrukih putanja omogućava veću propusnost između čvorova i bolje uravnoteženje velikih dugotrajnih vremenski manje osjetljivih i manjih kratkotrajnih vremenski osjetljivih tokova.

Jedno od prvih rješenja bio je MSTP (*Multiple Spanning Tree Protocol*) koji omogućava da različiti VLAN koriste različita stabla za razmjenu podataka. Ovim se povećala stabilnost i olakšala raspodjela opterećenja u većim mrežama.

Protokoli za agregaciju veza LAG (*Link-Aggregation*) omogućavaju povezivanje više fizičkih veza u jednu logičku. Ovo može biti između *switch*-eva ili između servera i *switch*-a. Protokol MC-LAG (*Multi-Chassis Link-Aggregation*) omogućava da jedna strana agregirane veze bude razdvojena na dva uređaja čime se ostvaruje redundantnost na nivou uređaja.

Strategija pronalaska putanja koja se često koristi u podatkovnim centrima je ECMP (*Equal-cost multi-path routing*). Na svakom koraku (*hop*) putanje moguće je korisiti više veza do narednog čvora. Time se poboljšava propusnost jer se umjesto jedne veze koristi više njih istovremeno. Kako se višestruke putanje koriste na svakom koraku ECMP se može koristiti sa većinom protokola rutiranja.

ECMP je ugrađen u SPB (*Shortest Path Bridging*) IEEE 802.1Q standard. Ovaj standard omogućava upotrebu višestrukih putanja na drugom sloju. Umjesto da blokira sve redundantne putanje, radi sprečavanja formiranja petlji, kao stari STP (*spanning tree protocol*), SPB omogućava da sve veze budu aktivne. Koristi IS-IS (*Intermediate System to Intermediate System*) protokol rutiranja za pronalazak putanja do svih odredišta. Koristi enkapsulaciju MAC-in-MAC ili Q-in-Q.

Alternativa SPB, koja dolazi iz IETF i takođe se koristi u podatkovnim centrima je TRILL (*Transparent Interconnection of Lots of Links*). Slično kao i SPB i TRILL koristi *link-state* protokole rutiranja mrežnog sloja da sazna informacije o cjelokupnoj mreži i na osnovu toga omogući slanje po više putanja na drugom sloju.

Protokol rutiranja koji se sve više koristi u podatkovnim centrima je BGP. Iako se smatra da je BGP protokol za WAN konekcije između AS, on ima neke osobine koje ga čine pogodnim za upotrebu u podatkovnim centrima. Korisno je da se u cijelom podatkovnom centru koristi jedan protokol nezavisno od proizvođača opreme, odnosno protokol kog svi proizvođači podržavaju. BGP se pokazao vrlo skalabilnim i mnogo proširivijim od internih protokola rutiranja. BGP tabele na globalnom Internetu imaju po stotine hiljada IP prefiksa, pa očigledno može skalirati i do potreba podatkovnog centra. BGP dobro radi filtriranje prefiksa, inženjering i označavanje (*tag*) saobraćaja što omogućava dobro upravljanje i kontrolu tokova u podatkovnom centru.

Postojanje više putanja između čvorova povećava propusnost i pouzdanost, ali da bi bila adekvatno iskorištena potrebno je da transportni sloj podržava prenos i raspoređivanje opterećenja preko više putanja. MPTCP (*Multi-path TCP*) koristi višestruke istovremene konekcije između krajnjih tačaka, pri čemu postoji mala interakcija između veličina njihovih prozora. Uspostavlja više podtokova po različitim putanjama između dvije iste krajnje tačke koji svi skupa čine jednu TCP konekciju. Pri ovome se vodi računa i o zagušenju na način da se saobraćaj više usmjerava na manje zagušene putanje. Radi skupa sa ECMP.

5.5.2 Zagušenje

Zbog posebnosti umrežavanja podatkovnih centara, tradicionalni pristup TCP kontroli zagušenja, kao što je TCP Reno, nije pogodan. Te posebnosti su [63]:

- RTT je mnogu manji i mjeri se u mikrosekundama, a ne u desetinama milisekundi;
- Aplikacije trebaju veliku propusnost i malo kašnjenje;
- Većinu propusnosti putanje zauzima po jedan tok;
- Veličina međuspremnika (*buffer*) u *switch*-evima koji se koriste je manja i dijeljeni su među portovima, radi uštede troškova, pa se brzo napune.

Standardna TCP kontrola zagušenja dobro radi sa velikim međuspremnicima čiji kapacitet bi trebao biti jednak bar proizvodu kašnjenja i propusnosti da

bi TCP mogao potpuno iskoristiti dostupnu propusnost. Radi toga TCP drži međuspremnike punim, što izaziva veće kašnjenje nego što je to prihvatljivo u podatkovnim centrima.

Algoritmi za kontrolu zagušenja pogodni za upotrebu u podatkovnim centrima mogu se podijeliti u dvije velike grupe. Jednu čine algoritmi koji ne zahtijevaju, veće, promjene u mrežnim uređajima i oslanjaju se na kontrolu u krajnjim tačkama. Drugu grupu čine algoritmi koji se oslanjaju na veću podršku mrežnih uređaja koja nije dostupna u standardnim uređajima. Primjeri ovakve podrške su mogućnost rezervacije propusnosti, prioritetno raspoređivanje ili raspoređivanje opterećenja paket po paket.

Predstavnik prve grupe je jedan od prvih i najpoznatijih algoritama kontrole zagušenja namijenjen za podatkovne centre, DCTCP (*Data Center* TCP) [55]. DCTCP je dizajniran da ostvaruje dobru propusnost i kada su međuspremnici slabo popunjeni. Kako je rečeno saobraćaj u podatkovnom centru sastoji se od manjeg broja velikih dugotrajnih i većeg broja malih kratkotrajnih tokova. Ovi mali tokovi čine da saobraćaj bude vrlo varijabilan (*bursty*). DCTCP teži da reaguje na zagušenje u skladu sa stepenom zagušenja. Kontrola zagušenja se provodi na slijedeći način. Mrežni uređaji kroz koje paketi prolaze postavljaju zastavicu CE (*Congestion Experienced*) na pakete kada popunjenost njihovih spremnika pređe zadanu, malu, vrijednost. Prijemnik odmah šalje potvrde za sve pakete koje dobije i u njima postavlja ECN (*Explicit Congestion Notification*)-Echo zastavicu, ako je CE zastavica u dolaznom paketu postavljena. Pošiljaoci na osnovu procenta paketa sa postavljenom ECN zastavicom smanjuju proporcionalno veličinu prozora zagušenja. Ova mala izmjena u standardnom ECN, da se indikacija zagušenja postavlja pri manjoj napunjenosti međuspremnika doprinijela je da se očuva propusnost uz mnogo manju prosječnu napunjenost međuspremnika.

5.6 Trendovi

Umrežavanje podatkovnih centara je oblast koja se intenzivno razvija. Podatkovni centri imaju sve više servera i prostora za pohranu. Propusnosti veza unutar podatkovnog centra su sve veće. Širenje koje bolje skalira je horizontalno. Dodaje se više istih servera, veza i mrežnih uređaja, umjesto zamjene servera, veza i mrežnih uređaja većim. Broj veza se povećava da bi se povećala propusnost i pouzdanost. Proširenja se uglavnom rade modularno, gdje jedan modul predstavlja skup servera, veza i mrežnih uređaja koji se zajedno kao jedna jedinica dodaju na postojeći podatkovni centar ili zamjenjuju staru jedinicu. Velika količina podataka koja se prebacuje na male udaljenosti po višestrukim vezama velike brzine i pouzdanosti unutra vrlo homogenog podatkovnog centra različita je od modela saobraćaja za koje su tradicionalni mrežni protokoli pravljeni. Razvijaju se novi protokoli prilagođeni potrebama, kako otvoreni i standardizovani, tako i zatvoreni od strane proizvođača opreme. Nije lako predvidjeti koji protokoli će postati dominantni. Ono što se, na osnovu

iskustava, može očekivati je da će otvoreni standardi koji imaju podršku IETF prevladati. Pri čemu će na razvoj ovih standarda veliki uticaj imati kompanije koje su najveći ponuđači usluga *cloud* računarstva i društvenih mreža i upravljači najvećih podatkovnih centara, poput Google, Amazon, Facebook i Microsoft.

Jedan pristup koji se prirodno nameće, a već se i koristi u podatkovnim centrima je SDN (*Software Defined Networking*). Podatkovni centar je velika i dobro povezana mreža u kojoj se koriste jednostavni mrežni uređaji sa ciljem da što brže prosljeđuju saobraćaj koja je pod kontrolom jedne organizacije. Centralno upravljanje prosljeđivanjem i jednostavna mogućnost promjena politika i protokola prosljeđivanja su jako korisni. Navedeno su okolnosti koje su kao stvorene za SDN. Iz tog razloga su podatkovni centri jedno od mjesta gdje je SDN najviše ušao u upotrebu. SDN-u je posvećeno naredno poglavlje knjige.

6

Softverski definisano umrežavanje

Softverski definisano umrežavanje (SDN), ili softverski definisane mreže, je popularan termin i trend. Ovaj pristup je drugačiji od tradicionalnog pristupa umrežavanju. Donosi mnoge prednosti, ali ima i određene nedostatke. SDN može biti vrlo pogodan i koristi se za neke namjene. Da bi se SDN pravilno koristio potrebno je razumjeti kako radi i šta tačno ovaj termin znači. Ovo poglavlje predstavlja kratak pregled SDN. O ovoj temi su napisane neke vrlo dobre knjige u kojim se može pronaći više detalja [23][16].

6.1 Potreba za SDN

Tradicionalne računarske mreže dizajnirane su da dobro skaliraju. Ovaj dizajn se očigledno pokazao uspješnim jer je omogućio razvoj Interneta do sadašnje veličine. Ova skalabilnost omogućena je autonomnošću mrežnih uređaja i distribuiranim odlučivanjem o putanji paketa koje efektivno definiše samo slijedeći uređaj kome treba proslijediti paket. Zapravo ovaj način rada dobro skalira samo do neke veličine mreže. Iz tog razloga postoje dva nivoa adresiranja, na podatkovnom i mrežnom sloju, te hijerarhijsko rutiranje, unutar autonomnih sistema i između njih. Pored ovog ograničenog skaliranja postoje još neke osobine tradicionalnih mreža koje nisu idealne.

Iako je odlučivanje o prosljeđivanju distribuirano i automatizovanao, ipak pojedinačne uređaje treba, uglavnom ručno, podesiti. Za veliki broj uređaja to predstavlja veliki napor i ostavlja mogućnost za greške, koje su se dešavala i najvećim davaocima usluga. U podatkovnim centrima koji rastu brzo ovo stvara poteškoću i negativno utiče na skalabilnost sistema.

Pošto se odluka o prosljeđivanju uglavnom donosi na osnovu adresa na jednom sloju svi paketi koji idu na isto odredište putuju istom putanjom. Danas postoji potreba da različiti tokovi za isto odredište, koji imaju različite zahtjeve za kvalitet usluge (QoS), imaju različite putanje kroz mrežu. Svaka putanja je pogodna za potrebni QoS za pojedini tok. U tradicionalnim mrežama

to je teško ili nemoguće ostvariti. Čak i, relativno jednostavna, potreba za distribuciju opterećenja slanjem saobraćaja po više putanja je samo ograničeno podržana protokolima rutiranja.

Mreže i protokoli pravljeni da se "snađu" (automatski) u slučaju ispada raznih vrsta, ma koliko rijetki bili. Danas su promjene, pogotovo u podatkovnom centru, mnogo češće od ispada, a one se uglavnom moraju konfigurisati ručno. Računarska infrastruktura, serveri i prostor za pohranu, je virutelizovana, dok mreža nije (osim VLAN što nije dovoljno). Dodavanje novog servera može se okončati za nekoliko minuta, dodavanje novog mrežnog segmenta može trajati danima. Iako se informacije o prosljeđivanju razmjenjuju automatski, postoji dosta informacija koje neophodnu unijeti u sve uređaje, ručno, da bi ovo funkcionisalo. Troškovi rada i održavanja (OPEX) postaju veći od troškova nabavke (CAPEX) mrežne opreme

Kako je o poglavlju posvećenom HTTP rečeno, pored tradicionalnih uređaja za prosljeđivanja paketa na drugom (*switch*) i trećem (ruter) sloju, u mrežama je sve više drugih uređaja kroz koje saobraćaj prolazi, takozvani *middlebox*. Ovi uređaji se dodaju radi sigurnosti (*firewall*, IDS, IPS, ...), poboljšanja performansi (*proxy*, *cache*), ...) nadzora i evidentiranja saobraćaja ili za druge namjene (NAT, ...). Svaki od ovih uređaja zahtjeva poseban softver koji ostvaruje traženu funkcionalnost uređaja. Ovi dodatni uređaji i njihovi softveri dodatno usložnjavaju rad i održavanje tradicionalnih računarskih mreža.

Tradicionalna mrežna oprema je integrisana i zatvorena. Proizvođač isporučuje hardver, operativni sistem i aplikacije. Korisnik nema mogućnost da dodaje svoje aplikacije ili protokole. Za proizvođače je razvoj opreme i softvera skup, jer u zatvorenim sistemima nema oslanjanja na biblioteke trećih strana ili okruženja poput operativnog sistema. Proizvođači se moraju oslanjati na sebe ili na, što je danas vrlo često, kupovinu manjih proizvođača koji imaju proizvodi koji je velikom proizvođaču potreban. Da bi ostvarili prednost na tržištu ili samo ostali konkurentni, proizvođači opreme često dodaju "poboljšanja" u odnosu na standardne protokole. Ovim oni zapravo otežavaju upotrebu proizvoda drugih proizvođača i vezuju kupca za sebe. Iz njihovog ugla to je prednost i opravdanje nalaze u velikim troškovima razvoja koje realno imaju. Nezavisnost i autonomnost uređaja, u savremenim okolnostima, čini da se u njih treba ugraditi dosta funkcionalnosti. Nekoliko velikih proizvođača drži većinu tržišta i monopolom otežavaju izmjene i razvoj novih mogućnosti. Nema dovoljno konkurencije. Zatvoreni softver otežava inovacije i razvoj i testiranje novih ideja i protokola. Promjene se teže uvode. To nije dobro za razvoj umrežavanja.

6.2 SDN promjene

SDN uvodi neke konceptualne promjene u način rada mrežnih uređaja i mreža koje se od njih grade. Te promjene su napravljene sa ciljem otklanjanja us-

tanovljenih nedostataka tradicionalnih računarskih mreža. Promjene se prije svega odnose na otvaranje uređaja razdvajanjem hardvera i softvera čime se stvara prostor za intenzivniji razvoj novih ideja i protokola i snižavanje cijena umrežavanja, te centralizaciju kontrolne ravni što omogućava pogled na cijelu mrežu i uopšteno prosljeđivanje.

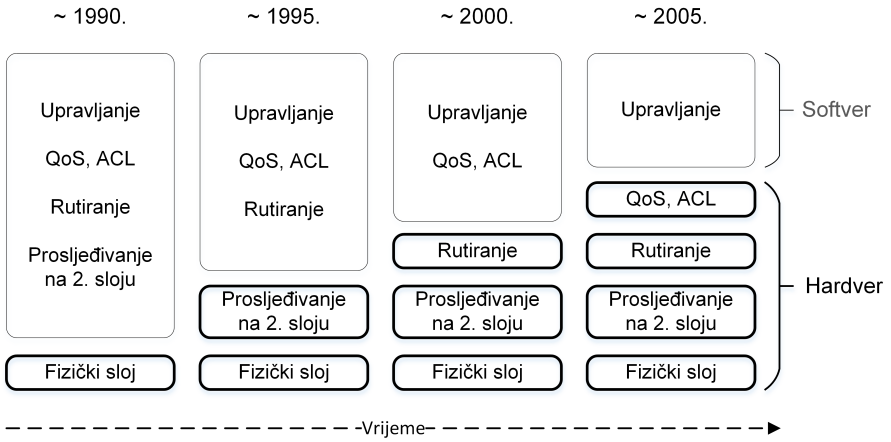
6.2.1 Razdvajanje hardvera i softvera

Ova promjena najuže je vezana za naziv softverski definisano umrežavanje. Šta to zapravo znači? Šta je to što se sada definiše softverski i kako se to definisalo u tradicionalnim mrežama/uređajima?

Zapravo u prvim mrežnim uređajima sve do sredine 1990-tih godina većina funkcionalnosti bila je izvedena u softveru. Hardver je radio prosljeđivanje paketa sa nekog od ulaznih portova na neki izlaznih portova. Sve ostalo bio je softver, od tabela prosljeđivanja preko algoritama na osnovu kojih su one popunjavane do filtriranja i upravljanja. Kako su brzine prenosa podataka po računarskim mrežama rasle, softver više nije mogao radi obradu paketa potrebnom brzinom. Srećom i hardver se razvijao, pa je sve više i više funkcionalnosti bilo moguće prebaciti u hardver. Prvo je to bilo samo prosljeđivanje na drugom sloju, zatim i rutiranje. Današnji mrežni uređaji se uglavnom sastoje od hardverskih komponenti poput ASIC (*application specific integrated circuits*), FPGA (*fully programmable gate arrays*) i TCAM (*ternary content addressable memories*). Ove komponente omogućavaju da se odluke o prosljeđivanju mogu raditi potpuno u hardveru i pri brzinama većim od 40 Gb/s. Savremeni hardver mrežnih uređaja omogućava da se i odluke o rutiranju, filtriranju putem ACL (*Access Control List*) i ostvarenju QoS (*Quality of Service*) mogu raditi u hardveru. Ono što je ostalo definisano u softveru su upravljačke funkcije visokog nivoa odgovorne za saradnju između svih uređaja na mreži. Potrebno je naglasiti da se ovaj kontrolni softver izvršava u svakom mrežnom uređaju nezavisno, kod tradicionalnih mreža. Promjene kroz vrijeme u tome šta se u mrežnim uređajima radilo u hardveru, a šta u softveru prikazane su na slici 6.1.

Računarska industrija odavno je prošla kroz fazu razdvajanja hardvera i softvera. U početku su proizvođači računara prodavali zajedno vlastiti hardver, operativni sistem za taj hardver i aplikacije za taj operativni sistem. Promjena proizvođača hardvera značila je promjenu operativnog sistema i svih aplikacija. Slično kao kod tradicionalne mrežne opreme razvoj je bio skup i zatvoren unutar nekoliko velikih proizvođača, poput IBM i DEC. Bilo je vrlo malo prostora za male nezavisne proizvođače softvera i inovacije.

Savremeni računari su u potpunosti napustili ovaj zatvoreni vertikalno integrisani model. Današnji računarski hardver je pretežno standardizovan na nekoliko preovladavajućih arhitektura, poput x86 i ARM. Za ovaj standardizovani hardver postoje različiti operativni sistemi koji se mogu instalirati na njega poput nekog od Unix-oidnih OS ili Windows OS. Operativni sistemi pružaju okruženje za koje proizvođači softvera, veliki i mali, mogu pra-

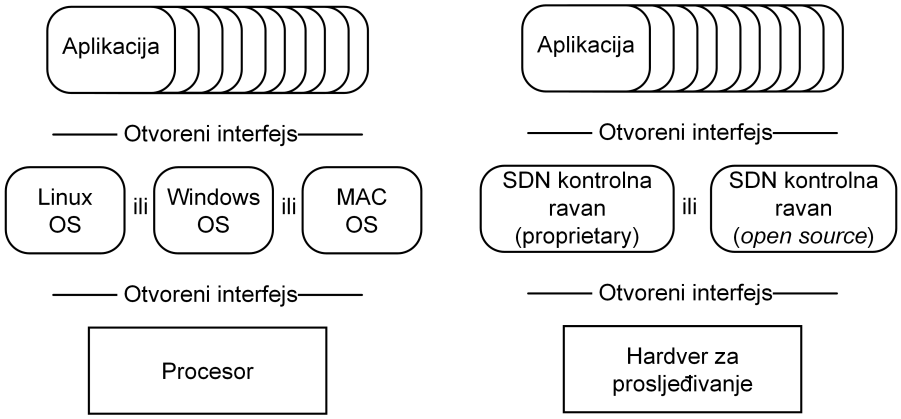


Slika 6.1: Promjena odnosa hardvera i softvera u mrežnim uređajima kroz vrijeme [23]

viti programe. Softver, za operativne sisteme i aplikacije, može biti otvoren ili vlasnički i biti razvijan od bilo koga. Standardizovani i otvoreni interfejsi između različitih slojeva, hardver, operativni sistem i aplikacije, stvorili su okruženje koje korisnicima pruža veće mogućnosti izbora. Ovakvo okruženje pogodno je za nezavisan razvoj softvera što je stvorilo konkurenciju i potaklo intenzivan razvoj, poboljšanja i industriju. Poseban primjer predstavlja Android okruženje koje je omogućilo ogromnom broju, onih koji razvijaju softver, da ga ja jednostavan način ponude korisnicima.

Softverski definisano umrežavanje uvodi ovaj pristup u mrežne uređaje i umrežavanje. Umjesto monolitnih vertikalno integrisanih mrežnih uređaja manjeg broja velikih proizvođača predlaže se otvorena arhitektura. Hardver za prosljeđivanje mogu praviti različiti proizvođači, specijalizirani za to. Bitno je da ovaj hardver ima otvorene i standardizovane interfejse putem kojih softver, koji može razviti neko drugi, može komunicirati sa hardverom. Taj softver provodi logiku prosljeđivanja i ostalih funkcija koje se mogu ostvariti na hardveru za prosljeđivanje na kom se izvršava. Pored logika koje provodi, taj softver treba da pruža mogućnost razvoja različitih mrežnih aplikacija. Mrežne aplikacije ne treba da budu zavisne od hardvera za prosljeđivanje, jer ih softver apstrahuje. Ova apstrakcija se ostvaruje stvaranjem logičke slike mreže kojoj aplikacije imaju pristup. Da bi to bilo moguće softver treba da pruža otvorene i standardizovane interfejse ka aplikacijama. Uporedba SDN pristupa sa savremenim računarima prikazana je na slici 6.2.

Na ovaj način se umrežavanje otvara. Korisnici dobijaju mogućnost izbora između različitih proizvođača hardvera i softvera. Proizvođači softvera, a i hardvera, dobivaju lakši pristup tržištu. Istraživači dobijaju mogućnost



Slika 6.2: Savremeni koncept računara i umrežavanja [59]

istraživanja novih algoritama i protokola. Sve ovo stimuliše inovacije i konkurenciju, te snižava cijene umrežavanja uz poboljšanje usluga.

6.2.2 Centralizovana kontrolna ravan

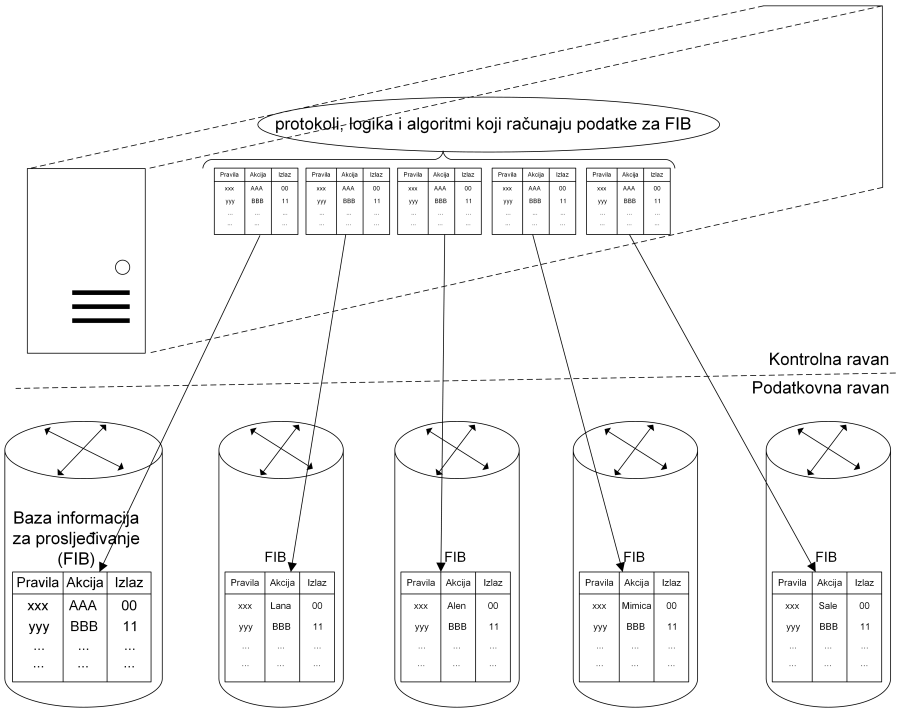
Autonomnost tradicionalnih mrežnih uređaja značila je da svaki od njih ima kompletan sistem sa hardverom i softverom za prosljeđivanje, kao i sa softverom za logiku prosljeđivanja. Ovakvi uređaji su bili kompleksniji i skuplji. Distribuirana logika pronalaženja putanja činila je da je konvergencija mogla trajati duže.

Zaglavlja paketa koji stignu na uređaj za prosljeđivanje (*switch*, ruter), na neki ulazni port, porede se pravilima upisanim u tabelu prosljeđivanja i na osnovu toga se obrađuju, ako je potrebno, i šalju na odgovarajući izlazni port. Tabela prosljeđivanja popunjena je na osnovu logike prosljeđivanja. Da bi paketi bili brže proslijeđeni sa ulaznog na odgovarajući port tabela prosljeđivanja bi trebala biti popunjena unaprijed. Logika prosljeđivanja bi trebala da popunjava i ažurira tabele prosljeđivanja nezavisno od dolaska paketa. Uobičajeno je da se dio mrežnog uređaja koji se bavi prosljeđivanjem naziva podatkovna ravan (*data plane*). Dio uređaja koji se bavi logikom prosljeđivanja, odnosno popunjavanjem tabela prosljeđivanja, naziva se kontrolna ravan (*control plane*). SDN koncept kontrolnu ravan odvaja od uređaja.

Umjesto kontrolne ravni distribuirane po mrežnim uređajima SDN koristi centralizovanu kontrolnu ravan. Važno je naglasiti, odmah na početku, da centralizovana kontrolna ravan ne znači da mora biti izvedena na jednom centralnom uređaju, kako se to obično pojavljuje na dijagramima. To samo znači da je odvojena od uređaja za prosljeđivanje, ali njena izvedba može biti na više odvojenih "kontrolnih" uređaja radi rasporeda opterećenje ili redundantnosti.

Na uređajima, u podatkovnoj ravni, je i dalje ostala baza informacija o prosljeđivanju, FIB (*Forwarding Information Base*). U centralizovanoj kon-

trolnoj ravni su izvedeni protokoli, logika i algoritmi koji računaju podatke za FIB. Za većinu ovih protokola je korisno što sad imaju globalan pogled na mrežu. Putanje se računaju na jednom mjestu i distribuiraju svim uređajima. Svi uređaji istovremeno dobivaju informacije koje su sinhronizovane. Nema čekanja da distribuirani protokol računanja putanja konvergira u stacionarno stanje. Nema mogućnosti da se pojave petlje usljed nesinhroniziranosti tabela prosljeđivanja. Izvedba centralizovane kontrolne ravni prikazana je na slici 6.3.



Slika 6.3: Centralizovana kontrolna ravan

Izdvajanjem kontrolne ravni iz uređaja za prosljeđivanje oni su postali mnogo jednostavniji. To znači da mogu raditi brže i da im cijena može biti niža. To dodatno otvara vrata drugim proizvođačima, stvara konkurenciju i dalje obara cijene.

Centralizovana kontrolna ravan nudi i druge zanimljive mogućnosti od kojih će neke biti razmotrene u nastavku.

6.2.3 Uopšteno prosljeđivanje

Tradicionalni mrežni uređaji prosljeđuju pakete na osnovu odredišne adrese mrežnog sloja na kom rade. *Switch* to radi na osnovu adrese na drugom, a

ruter na trećem sloju. Drugi mrežni uređaji (*middlebox*) koriste druge informacije na osnovu kojih obrađuju i prosljeđuju ili odbacuju paket. *Firewall* može analizirati zaglavlja trećeg i četvrtog sloja, slično kao i NAT. Svaku od ovih funkcija obavljaju različiti namjenski uređaji sa različitim hardverom i softverom koje treba posebno nabavljati konfigurisati i održavati.

U suštini ovi uređaji se uglavnom razlikuju u kontrolnoj ravni. Podatkovna ravan svih ovih uređaja radi istu stvar, prosljeđuje pakete na neke od portova ili ih odbacuje, pri čemu može mijenjati neka od zaglavlja paketa. Kada se kontrolna ravan izmjesti sa uređaja nema više potrebe da se podatkovna ravan u ovim uređajima razlikuje. Jedna generička podatkovna ravan koja obavlja navedene funkcije može biti dovoljna. To znači da se svi ovi uređaji mogu zamijeniti sa jednim uređajem koji se naziva SDN *switch* (ponekad i paket *switch*). Funkcija koju će ovaj uređaj obavljati zavisi od pravila koja će u njegovu bazu informacija za prosljeđivanje (FIB) upisati kontrolna ravan. Isti SDN *switch* može obavljati funkciju tradicionalnog *switch*-a koji radi prosljeđivanje na drugom sloju. a može obavljati i funkciju rutera, NAT-a, *firewall*, IDS, ... ovakvi jednostavni uređaji imaju nižu cijenu, i lakši su za konfiguraciju i održavanje.

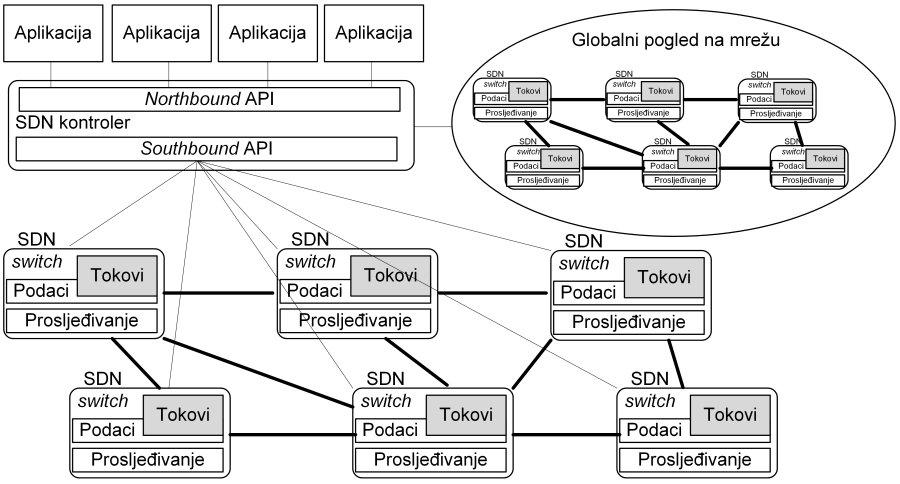
Mogućnost da isti uređaj obavlja različite funkcije je samo jedna od mogućnosti koje SDN pruža. Isti uređaj može obavljati i sasvim nove funkcije, koje nisu bile moguće u tradicionalnom umrežavanju. Prosljeđivanje paketa se sada, u principu, može raditi po skupu polja zaglavlja, pa čak i sadržaju, paketa koje kontrolna ravan definiše. Konkretne izvedbe SDN moraju to podržavati o čemu će biti više riječi kasnije u poglavlju. Sa SDN je moguće prosljeđivati pakete po više različitih putanja između dva ista čvora u mreži na način da svaki tok putuje putanjom najpogodnijom za QoS koji mu je potreban. Odredišna adresa više nije jedini element po kom se donosi odluka o prosljeđivanju paketa, Dodatna polja poput odredišnog porta mogu razdvojiti tokove koji pripadaju različitim aplikacijama i imaju različite QoS zahtjeve.

Uopšteno prosljeđivanje otvara priliku za razvoj novih algoritama i protokola koji ranije nisu bili mogući. SDN stvara okruženje u kom je testiranje novih protokola moguće u okviru mreže koja je pod kontrolom jedne centralizovane kontrolne ravni, bez potrebe za promjenom protokola u drugim mrežama.

6.3 Kako SDN radi

Navedene nove mogućnosti SDN ostvaruju se putem novog pristupa izvedbi mreže. Dio ovog pristupa može se naslutiti iz promjena koje su opisane. Na slici 6.4 prikazani su elementi SDN, njihove veze i uloge.

SDN *switch*-evi povezani su u računarsku mrežu po kojoj se prosljeđuju podatkovni paketi. Ovi uređaji imaju ugrađenu funkcionalnost koja im omogućava prosljeđivanje paketa. Uređaji takođe sadrže podatke na osnovu kojih rade



Slika 6.4: SDN način rada

prosljeđivanje. Pravila prosljeđivanja sastoje se od akcija koje se mogu uraditi sa paketom i filtera koji definiše na koje pakete se to pravilo i akcija odnose. Primjeri akcija su: proslijedi na neki port, odbaci, izmijeni polje zaglavlja. Filteri su zasnovani na konceptu toka. Tok je skup paketa koje treba tretirati na isti način. To uglavnom znači da putuju istom putanjom kroz mrežu i imaju iste zahtjeve na QoS. Tok je najčešće rezultat komunikacijske sesije dvije aplikacije. Filter definiše dijelove paketa, najčešće polja zaglavlja, zajedničke za sve pakete iz nekog toka. SDN *switch*-evi sa vezama između njih čine podatkovnu ravan.

Svi SDN *switch*-evi povezani su sa kontrolerom koji je izvedba centralizovane kontrolne ravni. Kontroler je funkcionalnost koja može biti izvedena na jednom ili više posebnih uređaja ili računara opšte namjene. U nastavku teksta koristi se termin kontroler, koji obuhvate sve moguće izvedbe. Po ovim vezama kontroler šalje *switch*-evima podatke na osnovu kojih rade prosljeđivanje. Po istim vezama *switch*-evi šalju kontroleru informacije o svojim vezama sa drugim *switch*-evima i stanju svojih interfejsa. Da bi se ova informacije mogle razmjenjivati neophodno je da postoji protokol koji podržavaju i kontroler i SDN *switch*-evi. Tih protokola im više, a najraširenij će biti opisan u narednom potpoglavlju.

Kontroler komunicira sa *switch*-evima putem takozvanog južnog (*sothbound*) API, koji podržava protokol koji se koristi za komunikaciju sa *switch*-evima. Kontroler na osnovu informacija koje dobiva od *switch*-eva stvara globalnu sliku mreže. Ta globalna slika mreže uključuje povezanost *switch*-eva, stanje njihovih interfejsa, te moguće opterećenje svakog od njih. *Switch*-evi prijavljuju svaku promjenu kontroleru, tako da kontroler uvijek ima ažurnu sliku cijele mreže. Ova slika potrebna je da bi sa na osnovu nje i politike

prosljeđivanja mogla formirati pravila koja se šalju prema *switch*-evima putem južnog API. Pravljenje ovih pravila zadatak je kontrolne ravni, čiji je kontroler dio. Ipak, u opštem slučaju, pravila se ne računaju na kontroleru već to rade aplikacije. Kontroler aplikacijama nudi interfejs po kom imaju uvid u povezanost mreže, bez nepotrebnih implementacionih detalja. Ovaj interfejs naziva se sjeverni (*northbound*) API. Kontroler ima ulogu neke vrste mrežnog operativnog sistema koji (mrežnim) aplikacijama daje apstraktnu logičku sliku (mrežnog) hardvera.

Aplikacije mogu obavljati različite funkcije poput pronalaska najboljeg puta kroz mrežu (rutiranje), pravljenja pravila za propuštanje i odbacivanje paketa (*firewall*, ACL) ili raspoređivanja opterećenja. Prostor aplikacija je otvoren za sve proizvođače, a funkcionalnosti su ograničene mogućnošću mreže i idejama autora. Ovo otvoreno okruženje može dovesti do revolucionarnog napretka u računarskim mrežama, kao što se desilo u svijetu računara i mobilnih uređaja.

Potrebno je napomenuti, da nema dominantnog niti široko rasprostranjenog protokola koji se koristi između aplikacija i kontrolera, sjeverni API. Iz tog razloga se u praksi često aplikacije, pogotovo standardne mrežne za rutiranje i ACL, izvršavaju na kontroleru.

Prosljeđivanje paketa kroz SDN može se odvijati na više načina. Jedan je da paket dođe na ulaz SDN *switch*-a, gdje se uklopi u neko od pravila koje uradi specificiranu akciju. Druga situacija je da se paket ne uklapa ni u jedno od pravila u kom slučaju SDN *switch* ne zna šta da radi sa paketom. Takvi paketi se prosljeđuju kontroleru. To je jedna od mogućih akcija u pravilima prosljeđivanja, koja ranije nije spomenuta. Kontroler analizira paket i određuje postupanje sa njim u skladu sa politikom koja je definisana u kontrolnoj ravni. Nakon odluke kontroler SDN *switch*-u šalje novo pravilo na osnovu kog kontroler postupa sa ovim paketom, kao i svim drugim koji se uklapaju u pravilo, pripadaju istom toku. Kontroler može, a to uglavnom i radi, da kad odredi putanju takvog, novog, paketa kroz mrežu, svim SDN *switch*-evima na toj putanji poslati pravilo tako da kad paket dođe do njih već znaju šta da sa njim rade. Ovim se omogućava da pravila budu dinamička, pravljena po potrebi, bez statičkog opterećivanja svih uređaja u podatkovnoj ravni sa pravilima koja nisu neophodna.

6.3.1 OpenFlow

OpenFlow je konkretna izvedba protokola za komunikaciju između kontrolera i SDN *switch*-a. To je najraširenij protokol sa tom funkcijom. često se čak SDN i OpenFlow poistovjećuju, mada je jedno koncept, a drugo protokol koji realizuje jedan dio tog koncepta. Rad koji je predstavio OpenFlow ideju i princip rada pojavio se već 2008. godine [35]. OpenFlow, kao i SDN, je došao u žižu javnosti kad je Google, 2012. objavio da za svoju internu mrežu, koja se prostire na tri kontinenta koristi SDN, odnosno OpenFlow, i da je time znatno poboljšao efikasnost rada [17].

OpenFlow definiše komunikacioni protokol između kontrolne i podatkovne SDN ravni, kao i logiku rada SDN OpenFlow *switch-a*. OpenFlow specifikacije objavljuje Open Network Foundadtion (ONF). Detalji tekuće specifikacije 1.5.1 javno su dostupni [19]. Protokolom su uređene poruke između OpenFlow kontrolera i OpenFlow *switch*-a, te ponašanje OpenFlow *switch*-a u različitim situacijama i reakcija na komande koje dobija od OpenFlow kontrolera. OpenFlow protokol podrazumijeva postojanje sigurnog kanala od *switch*-a do kontrolera, i obratno, po kom se razmjenjuju poruke. Sigurni kanal je u principu preko TLS, ali nije obavezno. U zatvorenom okruženju bi moglo bez TLS jer je brže i jednostavnije.

Savremeni hardverski uređaji za prosljeđivanje uglavnom imaju programabilne tabele. OpenFlow traži poopšten način njihovog programiranja, Treba napomenuti da postoje čisto softverske izvedbe OpenFlow, i uopšte, SDN *switch*-eva, ali one nisu dovoljno brze za produkciju. Takođe postoje hibridni *switch*-evi, koji imaju podršku za OpenFlow, ali mogu raditi i kao tradicionalni, Ethernet, *switch*-evi.

OpenFlow *switch* može da proslijedi paket na port uz moguće modifikacije zaglavlja, odbaci ga ili proslijedi OpenFlow kontroleru. Moderni Internet switch-evi donose milione odluka u sekundi. Ovakva brzina, 100 Gb/s, postiže se u podatkovnoj ravni, kontrolna ne može raditi tako brzo. Iz tog razloga se baza informacija za prosljeđivanje (FIB) nalazi na OpenFlow *switch*-evima.

FIB se ažurira iz OpenFlow kontrolne ravni. Ova ravan kod OpenFlow ima tri razlike u odnosu na tradicionalnu:

- različiti *switch*-evi se programiraju istim standardnim OpenFlow jezikom;
- izvršava se na odvojenom uređaju (kontroleru) od onog za prosljeđivanje (*switch*);
- više *switch*-eva programira se sa jednog kontrolera.

OpenFlow kontroler programira sva pravila u OpenFlow *switch*-evima. Kontroler definiše tok i kaže *switch*-u šta da radi sa paketima koji pripadaju tom toku. Razvojem OpenFlow protokola i moguće akcije evolviraju.

U prvoj kompletiranoj verziji OpenFlow protokola V1.0 definisani su osnovni principi rada. Ti principi su izloženi u nastavku. Nakon njih su navedene bitne izmjene koje su nove verzije protokola donijele.

OpenFlow (*switch*) port, je u verziji 1.0 samo fizički, i može imati više, izlaznih, redova čekanja. Baza informacija za prosljeđivanje (FIB) se kod OpenFlow naziva tabela tokova. Tabela tokova sastoji se od unosa koje čine polja zaglavlja, brojači i akcije za taj unos. Po poljima zaglavlja se paket poredi (*match*), brojači služe za statistike, a akcije određuju šta se radi sa paketom koji se podudara sa definisanim poljima za poređenje.

Poređenje paketa vrši se po 12 polja:

- SWITCH ulazni port
- VLAN ID
- VLAN prioritet

- Ethernet izvorišna adresa
- Ethernet odredišna adresa
- Ethernet tip
- IP izvorišna adresa
- IP odredišna adresa
- IP protokol
- IP TOS biti
- TCP/UDP izvorišni port
- TCP/UDP odredišni port

Prvi unos u koji se paket uklapa određuje šta se radi sa njim. Paket se dalje ne poredi sa ostalim unosima. Iz tog razloga je redoslijed unosa u tabeli tokova vrlo bitan. Ako se zaglavlja paketa ne podudaraju ni sa jednim unosom, paket se šalje kontroleru.

Unosi polja zaglavlja za poređenje ne moraju obuhvatati svih 12 polja. Moguće je definisati da se poredi samo jedno ili bilo koji broj polja do 12. Polja za koja nije definisana vrijednost ne porede se sa poljima paketa, odnosno svaka vrijednost tog polja u paketu se poklapa sa uslovom poređenja. Vrijednost za poređenje za polja sa kojim se ne vrši poređenje u filteru je obično znak "*" koji se naziva *wildcard*. Na ovaj način se definisanjem poređenja samo po polju odredišne MAC adrese i akcije prosljeđivanja na odgovarajući port može realizovati funkcionalnost *switch*-a ili istom akcijom ali poređenjem po odredišnoj IP adresi ostvaruje se funkcionalnost rutera.

Akcije koje OpenFlow *switch* može raditi su *output/formard* (proslijedi) ili *drop* (odbaci) paket. Prosljeđivanje može biti na neki od fizičkih portova *switch*-a. Nadalje, da bi se ostvarile različite potrebne funkcionalnosti, paket može biti proslijeđen na neki od virtuelnih portova. Primjer virtuelnog porta je ALL/FLOOD koji prosljeđuje paket na sve fizičke portove osim onog po kom je došao. Ovim se simulira *broadcast* slanje. Slanje paketa kontroleru se ostvaruje slanjem na virtuelni port CONTROLLER.

Dvije opcionalne, ali bitne akcije su *modify-field* (izmijeni polje) i *enqueue* (stavi u red čekanja). "Izmjeni polje" definiše koja polja paketa i na koji način *switch* treba da izmjeni. Ova akcija omogućava rad OpenFlow *switch*-a kao recimo rutera koji mijenja polja sa izvorišnom i odredišnom MAC adresom. "Stavi u red čekanja" definiše u koji od redova čekanja, ako ih ima više, vezanih za port na koji se paket prosljeđuje treba staviti paket. Ovim se omogućava podrška za QoS.

Komunikacija između OpenFlow kontrolera i OpenFlow *switch*-a odvija se, slično kao kod većine aplikativnih protokola, putem razmjene poruka iz predefinisanog skupa. Kako je ovo aplikativni protokol poruke se razmjenjuju korištenjem TCP transportnog protokola. Da bi se pružio siguran kanal za komunikaciju preko TCP konekcije uspostavlja se TLS konekcija po kojoj se razmjenjuju OpenFlow poruke.

Ove poruke, uz tabelu tokova, omogućavaju da OpenFlow obavlja svoju ulogu u mreži. Poruke se dijele na tri grupe: simetrične, asinhrone i kontroler-*switch*.

Simetrične poruke može poslati bilo koja strana. Prva od ovih poruka, a i prva poruka koja se razmjenjuje nakon uspostavljanja TCP i TLS konekcije je OFPT_HELLO poruka. Ovu poruku šalju i kontroler i *switch*. U toj poruci navode najvišu verziju OpenFlow protokola koju podržavaju. Nakon toga komunikacija se nastavlja korištenjem niže od verzije protokola koje su strane ponudile. Druge simetrične poruke su OFPT_ECHO_REQUEST i OFPT_ECHO_REPLY. Namjena ovih poruka je da omoguće bilo kojoj od strana da provjeri da li je druga strana još dostupna, te kakvi su propusnost i kašnjenje u komunikaciji sa njom.

Asinhrone poruke šalje *switch* kontroleru kad god dođe u situaciju da je to potrebno. To se dešava u više situacija. Jedna je kad OpenFlow *switch* dobije paket koji se ne podudara ni sa jednim od unosa u tabeli tokova ili ako se podudara sa unosom za koji je akcija da se paket šalje kontroleru. Takav paket se dostavlja kontroleru putem poruke OFPT_PACKET_IN. U slučaju promjene statusa nekog od portova na *switch*-u, kontroleru sa šalje poruka OFPT_PORT_STATUS. Ako se iz tabele toka ukloni neki tok, jer je od posljednjeg paketa koji se podudario sa tim unosom isteklo podešeno vrijeme čekanja, *switch* kontroleru šalje poruku OFPT_FLOW_REMOVED. Postoji i poruka OFPT_ERROR kojom se kontroler obavještava o greškama koje se dese na *switch*-u. Ovaj skup poruka omogućava da kontroler dobije pakete sa kojim treba da odluči šta da radi (podatkovni paketi) i pakete koji su njemu namijenjeni (kontrolni paketi, poput paketa sa informacijama o rutiranju), te da bude pravovremeno informisan o promjenama na *switch*-evima bitnim za kontrolnu ravan.

Najveći broj poruka spada u grupu onih koje kontroler šalje *switch*-u. Ove poruke podijeljene su u pet kategorija: konfiguracija *switch*-a, komande od kontrolera, statistike, konfiguracija redova čekanja i barijera. Pošto nazivi svih poruka počinju sa "OFPT_", radi kraćeg pisanja, ovaj dio će u nastavku teksta biti izostavljen iz naziva svih poruka.

Poruke konfiguracije *switch*-a su one kojom kontroler pita *switch* za njegove mogućnosti (FEATURES_REQUEST) i kojom pita za njegovu konfiguraciju (GET_CONFIG_REQUEST), kao i poruke kojim *switch* odgovara na ova pitanja (FEATURES_REPLY) i (GET_CONFIG_REPLY). Postoji i poruka kojom kontroler može postaviti konfiguracione parametre *switch*-a na željene vrijednosti (SET_CONFIG). Ovaj skup poruka omogućava kontroleru da prikupi informacije o mogućnostima i konfiguracijama *switch*-eva sa kojim je povezan, te uradi dostupne izmjene konfiguracija. Ove informacije pomažu donošenju odluka u kontrolnoj ravni.

Poruke komandi od kontrolera su jednosmjerne, od kontrolera ka *switch*-u. Kada kontroler vraća *switch*-u paket (koji je ranije dobio putem poruke PACKET_IN) na dalje prosljeđivanje, za to koristi poruku komande PACKET_OUT. Kontroler ovim porukama nože mijenjati unose u tabeli tokova na

switch-u (FLOW_MOD) i status OpenFlow porta na *switch*-u (PORT_MOD). Ova kategorija poruka omogućava da kontroler upravlja tabelama tokova na *switch*-evima i time definiše pravila prosljeđivanja.

Poruke statistika su samo zahtjev kontrolera za dobivanje statistika od *switch*-a (STATS_REQUEST) i odgovor (STATS_REPLY) koji dolazi od *switch*-a. Statistike su bitne za kontrolnu ravan jer pokazuju opterećenost uređaja i veza u mreži.

Poruke barijere omogućavaju kontroleru da zahtjeva da *switch* izvrši sve komande koje je dobio (BARRIER_REQUEST) i o tome obavijesti kontroler (BARRIER_REPLY). Tako kontroler može imati barijeru, razdvojiti stanja *switch*-eva znajući da su sve poslane komande izvršene, prije izvršavanja drugih komandi koje se oslanjaju na ove prethodne.

Pouke konfiguracije redova čekanja zapravo samo omogućavaju slanje zahtjeva ka *switch*-u za ovu konfiguraciju (QUEUE_GET_CONFIG_REQUEST) i odgovor od *switch*-a (QUEUE_GET_CONFIG_REPLY). OpenFlow protokol se na bavi konfiguracijom redova čekanja na *switch*-evima.

Niz OpenFlow poruka koje se razmjenjuju može se pokazati na primjeru sa kraja prethodnog poglavlja o načinu rada SDN. Potrebno je napomenuti da se OpenFlow kontrolna ravan ponaša reaktivno. To znači da se putanje računaju kad se pojavi paket za koji se ne zna putanja, a ne unaprijed kao kod standardnih protokola rutiranja. Naravno, moguće je da konkretna izvedba ima podešenja da radi proaktivno.

Neka na neki OpenFlow *switch* dođe paket koji se ne uklapa ni u jedno od pravila u tabeli tokova. Taj *switch* šalje poruku PACKET_IN svom kontroleru. U toj poruci navodi razlog za slanje (OFPR_NO_MATCH) i dostavlja sva zaglavlja paketa. Cijeli paket se čuva u *switch*-u, ali se zbog uštede propusnosti samo zaglavlja dostavljaju kontroleru (ako *switch* to podržava i ima prostora u memoriji), osim ako nema posebne potrebe za sadržajem paketa (što je slučaj kod kontrolnih paketa). Kontroler pregleda zaglavlja paketa, odluči šta treba da se radi sa njim (uglavnom poredeći ga sa definisanim politikama ili na drugi način), te na osnovu toga pronađe putanju kroz mrežu, niz OpenFlow *switch*-eva sa kojim je povezan, kojom paket treba da bude proslijeđen do odredišta. Svakom od OpenFlow *switch*-eva na ovom putu kontroler šalje poruku FLOW_MOD kojom dodaje unos u tabele tokova svakog od njih koje će omogućiti da ovaj paket, i svi koji pripadaju istom toku, bude proslijeđen narednom OpenFlow *switch*-u na utvrđenoj putanji. Kontroler zatim porukom PACKET_OUT "vraća" paket *switch*-u koji mu ga je poslao. U toj poruci daje upute šta raditi sa paketom, što može biti da se pošalje na neki izlazni port ili da se ponovo uporedi sa tabelom tokova u kojoj sada postoji unos sa kojim se poklapa i koji definiše izlazni port po kom treba biti proslijeđen. Nakon prosljeđivanja sa prvog *switch*-a, paket će prolaziti kroz niz *switch*-eva u kojim već postoji unos u tabeli tokova za tok kom paket pripada. Na osnovu tih unosa paket će biti proslijeđen do svog odredišta, u skladu sa politikom, bez potrebe za daljim kontaktiranjem kontrolera. I svo naredni paketi, koji pripadaju istom toku će biti proslijeđeni po istoj putanji. Kada istekne

vremenski period bez prosljeđivanja ijednog paketa koji pripada ovom toku (podudara se sa unosom u tabelu tokova, definisan u poruci FLOW_MOD kojom je tok dodan u tabelu), svaki od *switch*-eva na kom je istekao ovaj period poslaće kontroleru poruku FLOW_REMOVED, navodeći tok koji je uklonjen iz tabela. Na ovaj način se racionalizira upotreba resursa na *switch*-evima. To je relativno uobičajen pristup u računarskim mrežama, da neki unos u tabeli ističe, ako se ne osvježava (*soft state*). ARP tabele u standardnim Ethernet *switch*-evima su primjer.

Kasnije verzije OpenFlow protokola dodavale su nove mogućnosti uporedbe i obrade paketa. Princip rada se nije mijenjao. Bez ulaska u detalje i navođenje svih promjena navode se neke bitne. Detalji tekuće verzije OpenFlow protokola mogu se pronaći na web lokaciji Open Network fondacije [20].

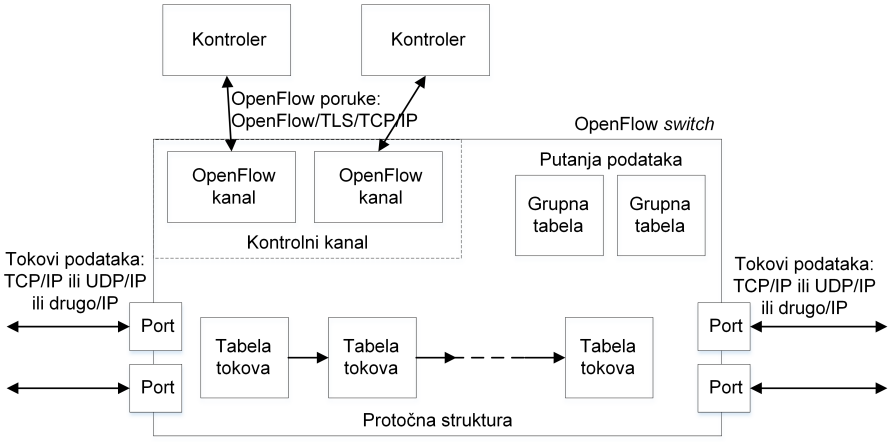
Dodatne mogućnosti podudaranja i obrade paketa ostvarene su uvođenjem višestrukih tabela tokova i grupnih tabela. Višestruke tabele su konstrukcija gdje se podudaranje paketa može raditi ne samo sa jednom, već sa nizom tabela tokova. To omogućavaju da se jedan paket propusti kroz više tabela i dobije se preciznije podudaranje sa željenim kriterijem. Ove tabele omogućavaju i ugnježdene tokove, gdje se u prvoj tabeli tokovi razdvajaju na veće grupe, a onda u svakoj narednoj razdvajanje pa podgrupe i pojedinačne tokove. Grupne tabele omogućavaju da se grupa portova posmatra kao jedan logički port. Time se povećava podrška za *broadcast* i *multicast*. Pojam port proširen je sa fizičkog i na virtuelni čime se prosljeđivanje dodatno poopštilo. Povećan je broj polja zaglavlja sa kojim se može vršiti poređenje. Bitan dodatak su IPv6 adrese, kao i drugi dijelovi zaglavlja. Poboljšana je podrška za rad sa više kontrolera, radi povećanja pouzdanosti. Prošireni su brojači da obuhvataju statistike po tokovima i redovima čekanja. Uvedeni su *cookie* koji smanjuju količinu podataka koji se razmjenjuju između *switch*-eva i kontrolera, što pozitivno utiče na efikasnost i bitno je za mreže velike propusnosti.

Elementi OpenFlow *switch*-a i njegova interakcija sa okolinom prikazani su na slici 6.5

OpenFlow je protokol koji je ostvario SDN ideju. Kao i SDN oblast i ovaj protokol se još razvija. Razvoj se kreće u pravcu uklanjanja postojećih ograničenja. Jedno od njih je da se podudaranje paketa radi samo po zaglavljima. Sve više postoji potreba da se paketi mogu filtrirati i po sadržaju (*deep packet inspection*). Drugo ograničenje ima veze sa razvojem hardvera za OpenFlow *switch*-eve, koji trenutno ograničava šta je moguće uraditi.

6.4 Upotreba SDN

SDN ima potencijal da bude upotrebljen u različitim računarskim mrežama. U nastavku će biti navedena dva primjera upotrebe koji se, u vrijeme pisanja, najviše koriste.



Slika 6.5: OpenFlow *switch* [59]

6.4.1 SDN u podatkovnom centru

Logično mjesto za upotrebu SDN su podatkovni centri. Njihove potrebe, kako je objašnjeno i poglavlju 5, teško se mogu zadovoljiti tradicionalnim mrežama. SDN nudi mogućnost rješavanja nekih od otvorenih pitanja.

Višestruke putanje između servera u podatkovnom centru omogućavaju veću propusnost i pouzdanost veza između servera. Međutim, tradicionalni mrežni protokoli nisu dizajnirani za takve situacije. Centralizovana kontrolna ravan omogućava da SDN pruža uvid u cjelokupnu mrežu na jednom mjestu. Taj uvid podrazumijeva i opterećenost pojedinih veza. Na osnovu uvida u stanje mreže i potrebe tokova, moguće je donositi optimalne odluke o prosljeđivanju, koje su pored toga i predvidive. Ove odluke se direktno isporučuju uređajima za prosljeđivanje (*switch*) u obliku unosa u tabele tokova. Ovim odlukama se upravlja saobraćajem u mreži putem izbora putanja i propusnosti. Moguća je i prioritetizacija pojedinih tokova u skladu sa QoS oznakama u paketima, ako se koriste.

Ispadi nekih od veza, ma kako rijetki u podatkovnom centru, su mogući. Ovakve situacije treba što brže obraditi i preusmjeriti saobraćaj po vezama koje rade. Ponovo je centralizovana kontrolna ravan sa globalnim pogledom na mrežu vrlo pogodna za pronalaženje i uspostavljanje novih putanja kroz mrežu.

Računarski resursi u podatkovnom centru su virtuelizovani i lako se povećavanju, smanjuju i premještaju po potrebi. Tradicionalna mreža nije virtuelizovana i ne može se tako brzo izmijeniti da bi u potpunosti podržala promjene na, virtuelizovanim, računarskim resursima. SDN može omogućiti automatizaciju koja će uraditi potrebne izmjene na mreži brzo i efikasno. Aplikativni sloj SDN može dobivati informacije od sistema za upravljanje računarskim resursima u podatkovnom centru. Na osnovu tih informacija

može pripremiti upute za kontrolni SDN sloj da izvrši izmjene pravila koje će podržati izmjene računarskih resursa. Kontrolni sloj priprema izmjene i šalje ih podatkovnom sloju.

SDN može izaći na kraj sa velikim tabelama MAC adresa i ograničenim brojem dostupnih VLAN oznaka. SDN može omogućiti tuneliranje i enkapsulaciju koji će putem slaganja VLAN oznaka povećati broj mogućih VLAN ili raditi drugu vrstu tuneliranja za koje nisu potrebne VLAN oznake. Zbog izvedbe tabela tokova u *switch*-evima broj MAC adresa koji se nalazi u tabelama može biti mnogo manji.

Kompanije koje imaju velike podatkovne centre već uveliko koriste SDN. Google, Microsoft Azure i Goldman Sachs koriste OpenFlow protokol u svojim podatkovnim centrima. eBay i Rackspace koriste VMware's Nicira rješenje u kom je većina mrežne funkcionalnosti, uključujući *switch*-eve i uređaje, virtuelizovano u hipervizoru.

6.4.2 SD-WAN

SDN ideja razdvajanja podatkovne i kontrolne ravni našla je plodno tlo u WAN mrežama. Prelazak organizacija na *cloud* doveo je do velike količine saobraćaja sa različitih lokacija organizacije prema, obično, centralizovanom *cloud*. Ovo povezivanje može biti putem javne Internet mreže ili posebne mreže organizacije. Prvo rješenje ima nižu cijenu, ali slabu podršku za QoS koji može biti neophodna nekim poslovnim aplikacijama. Ponekad se potrebna propusnost može nadoknaditi sa višestrukim Internet vezama, čija cijena opada iz dana u dan. Drugo rješenje, koje se najčešće ostvaruje upotrebom MPLS, može omogućiti potrebni QoS, ali je vrlo zahtjevno novčano i po potrebnim ljudskim resursima.

SD-WAN kombinuje ova dva pristupa. Sve WAN konekcije, kojih može biti više različitih (putem ISP, iznajmljena linija, zakupljena propusnost, ...) posmatra kao podatkovnu ravan. Kontrolna ravan aplikacijama apstrahuje podatkovnu ravan i svakoj od aplikacija obezbjeđuje vezu potrebnog QoS, u okviru mogućnosti. Softverski upravljiva kontrolna ravan kontinualno prati stanje i karakteristike svih putanja u podatkovnoj ravni. Putanje se mogu razlikovati po pouzdanosti (gubitku paketa), varijabilnosti kašnjenja (*jitter*) i opterećenosti. To omogućava da se u svakom trenutku zahtjevi za prenos podataka preko ovih veza rasporede prema prioritetima i QoS potrebama. SD-WAN omogućava definisanje politika na osnovu kojih se može raditi ovo raspoređivanje. Politike mogu biti definisane na nivou aplikacija.

Kontrolna ravan brine se i za šifriranje saobraćaja, ako je to potrebno radi ostvarivanja sigurnosti podataka prilikom putovanja po javnoj Internet mreži. U slučaju ispada neke od veza kontrolna ravan radi preusmjeravanje saobraćaja po vezama koje rade i odgovarajuće preraspoređivanje saobraćaja. Da bi ovo bilo moguće potrebno je da kontrolna ravan ima mogućnost nadzora veza i njihovog upravljanja dovoljno brzo. Za ovo je neophodno da uređaji u

podatkovnoj ravni podržavaju ove mogućnosti. SD-WAN izvedbe se obično sastoje iz nekoliko komponenata:

- Posebni uređaji, hardverski ili virtuelni, koji zamjenjuju rutere u krajnjim tačkama WAN veza
- Izmijenjene verzije protokola rutiranja koje omogućavaju pronalaženje putanja i distribuciju politika prosljeđivanja kroz mrežu
- Sigurnosni protokol, obično TLS ili IPSec, za osiguravanje prenosa podataka
- Kontroler za nadzor stanja veza i prilagođavanje podešavanje prema potrebama

Kod SD-WAN odluku o izboru putanje ne donosi onaj koji konfiguriše mrežu, što je kod WAN često slučaj, niti standardni protokoli rutiranja. Odluke donosi softver u realnom vremenu na osnovu definisane politike. Ovo rješenje dobro skalira u slučaju povećanja broja veza ili proširivanja njihovog kapaciteta. Nema potrebe za posebnim dodatnim konfigurisanjem.

Na ovaj način aplikacije imaju utisak da su povezane putem posebne mreže organizacije, dok veze zapravo mogu biti isključivo javne Internet.

Nedostatak dostupnih SD-WAN rješenja je što uglavnom nisu *open source*, već su pojedinih proizvođača. Tu dominiraju tradicionalne kompanije poput VMWare i Cisco.

6.5 Nedostaci SDN

Uz sve prednosti koje donosi, SDN ima i neke potencijalne nedostatke. Neki od njih su suštinski, a neki se mogu otkloniti u budućnosti. Slijedi njihov kratak pregled.

Vjerovatno najveći, potencijalni, nedostatak dolazi od njegove centralizovane prirode. Distribuirani sistemi tradicionalnih mreža pravljeni su da rade u slučaju ispada pojedinih čvorova. Svaki čvor ima kontrolnu ravan koja mu omogućava da izračuna putanje prema tekućem stanju u mreži. U slučaju ispada SDN centralizovane ravni mreža postaje nefunkcionalna. Zapravo bi podatkovna ravan mogla nastaviti rad sa postojećim pravilima prosljeđivanja, ali se ona ne bi više mogla prilagođavati okolnostima u mreži ili novim tokovima. Centralizovana ravan je izvedena na čvoru, koji je obično računar povezan sa svim SDN *switch*-evima. Na tom računaru može otkazati hardver, softver i veze. Ranije je napomenuto da centralizovana ravan ne podrazumijeva jedan uređaj, već samo centralizovanu funkciju. Za osiguravanje pouzdanog rada SDN ovo je neophodno. Centralizovana ravan može biti izvedena na više SDN kontrolera koji mogu podijeliti radi i služiti kao rezerva u slučaju ispada nekog od njih. SDN *switch*-evi trebali bi imati veza se više od jednog od ovih kontrolera.

Drugi potencijalni nedostatak SDN je vezan za isto pitanje. Centralizovana kontrolna ravan može predstavljati usko grlo i predstavljati smetnju skaliranju. Centralizovana kontrolna ravan računa i dostavlja pravila prosljeđivanja

za sve SDN *switch*-eve. Sa porastom mreže, raste broj *switch*-eva i opterećenje na kontrolnu ravan. Ako kontrolna ravan ne može obaviti sve što je potrebno, na vrijeme, mreža neće raditi kako treba. Iz tog razloga bi centralizovana kontrolna ravan trebala biti dizajnirana da dobro skalira. Očigledno da je potrebno više kontrolera i veza sa *switch*-evima. Mogući dizajn kontrolne ravni i veza sa *switch*-evima, koji bi bio skalabilan je nešto slično *fabric*-u iz podatkovnih centara. Broj kontrolera i veza sa *switch*-evima je uvijek takav da ne utiče negativno na propusnost i performanse.

Neki od izazova koje SDN može prevazići, ali još nije, vezani su za neke uređaje i načine rada koji postoje u tradicionalnim mrežama. Neki od uređaja u tradicionalnim mrežama, poput NAT i *firewall*, oslanjaju se na čuvanje informacija o tokovima koji prolaze kroz njih, što se naziva čuvanje stanja. SDN *switch* posmatra svaki paket odvojeno i ne čuva stanje. Filtriranje saobraćaja. recimo na *firewall*, se sve više oslanja na pregled ne samo zaglavlja već i sadržaja paketa (*deep packet inspection*). SDN *switch* koristi samo polja zaglavlja za izbor akcije koju će uraditi nad paketom. Oba ova pitanja bi se mogla riješiti i vjerovatno će takva funkcionalnost biti dodana u buduće izvedbe SDN.

Još jedna prepreka na putu bržeg usvajanja SDN, koja nije nedostatak već okolnost sa kojom se susreću sve nove ideje koje narušavaju stare koncepte, je velika promjena koncepta. Uvođenje SDN znači promjenu sve mrežne opreme i softvera u mreži. Ovo je neprihvatljivo za postojeće mreže, osim one vrlo male. Veliki proizvođači mrežne opreme, naravno, nisu oduševljeni da njihova oprema, u čiji su razvoj mnogo uložili, bude zamijenjena drugom opremom generičkih proizvođača hardvera za prosljeđivanje. Očigledno je da će se SDN morati polagano uvoditi i postojaće period koegzistencije sa tradicionalnim mrežama.

Postoje i dva alternativna načina uvođenja SDN. Veliki proizvođači mrežne opreme kreću u podršku SDN putem postojećih API (*Application Programming Interface*). To je korak u pravcu mreža kojim upravljaju kontroleri, koji mogu biti upravljani softverom. Drugi način je da se nad postojećim fizičkim mrežnim uređajima i mrežom formira virtuelna mrežna infrastruktura. Krajnji uređaji svjesni su samo ove virtuelne mrežne infrastrukture. Fizička infrastruktura se ne mora mijenjati. virtuelna mrežna infrastruktura formira, virtuelne, veze i, virtuelne, uređaje u skladu sa potrebama krajnjih čvorova. Ovakav pristup izvedbi SDN naziva se *hypervisor-based overlay networks*.

* * *

Ovim je završen kratak opis SDN mogućnosti i izvedbi. SDN više nije tehnologija koja obećava, već ona koja se koristi. Za očekivati je dalji razvoj SDN ideja i uređaja.

Literatura

1. *IEEE Standard for Low-Rate Wireless Networks*. IEEE std. 802.15.4-2015, 2015.
2. IEEE Standard for Information technology–Telecommunications and information exchange between systems - Local and metropolitan area networks–Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 2: Sub 1 GHz License Exempt Operation. *IEEE Std 802.11ah-2016 (Amendment to IEEE Std 802.11-2016, as amended by IEEE Std 802.11ai-2016)*, pages 1–594, 2017.
3. Bluetooth Core Specification. `https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=478726`, 2019.
4. Dennis Abts and Bob Felderman. A guided tour through data-center networking. *Queue*, 10(5):10–23, May 2012.
5. Mohammad Afaneh. *Intro to Bluetooth Low Energy: The easiest way to learn BLE*. Independently published, 2018.
6. Reza Niazmand Xu Wang Alex Eckert, Luis MartinGarcia. Wedge 100: More open and versatile than ever, 2016.
7. Alex Eckert Alexey Andreyev, Xu Wang. Reinventing facebook's data center network, 2019.
8. Lora Alliance. LoRaWAN 1.1 Specification. https://lora-alliance.org/resource-hub/lorawanr-specification-v11, 2017.
9. ZigBee Alliance. Zigbee specification - 2007, 2008.
10. APNIC. Bgp routing table analysis reports. `https://bgp.potaroo.net/`, 2020. [Online; pristupano 21.1.2020.].
11. A. Barth. HTTP State Management Mechanism. RFC 6265, RFC Editor, April 2011.
12. Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, pages 13–16, New York, NY, USA, 2012. ACM.
13. A. Keranen C. Bormann, M. Ersue. Terminology for Constrained-Node Networks. RFC 7228, RFC Editor, May 2014.
14. Walter Colitti, Kris Steenhaut, and Niccolò De Caro. Integrating wireless sensor networks with the web. *Extending the Internet to Low power and Lossy Networks (IP+ SN 2011)*, 2011.

15. R Coppen, A Banks, E Briggs, K Borgendale, and R Gupta. Mqtt version 5.0. *Standards Track Work Product*, 2019.
16. Thomas Nadeau D. and Ken Gray. *SDN: Software Defined Networks*. O'Reilly Media, Inc., 1st edition, 2013.
17. Art Fewell. Google showcases openflow network. `https://www.networkworld.com/article/2222173/google-showcases-openflow-network.html`, 2012. [Online; pristupano 20.12.2019.].
18. Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, UNIVERSITY OF CALIFORNIA, IRVINE, 2000.
19. Open Network Foundadtion. Openflow switch specification - version 1.5.1 (protocol version 0x06). `https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf`, 2015. [Online; pristupano 25.12.2019.].
20. Open Network Foundadtion. Software defined standards - specifications. `https://www.opennetworking.org/software-defined-standards/specifications/`, 2019. [Online; pristupano 25.12.2019.].
21. J. Hui D. Culler G. Montenegro, N. Kushalnagar. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944, RFC Editor, September 2007.
22. Pedro Garcia Lopez, Alberto Montresor, Dick Epema, Anwitaman Datta, Teruo Higashino, Adriana Iamnitchi, Marinho Barcellos, Pascal Felber, and Etienne Riviere. Edge-centric computing: Vision and challenges. *SIGCOMM Comput. Commun. Rev.*, 45(5):37–42, September 2015.
23. Paul Goransson and Chuck Black. *Software Defined Networks: A Comprehensive Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2014.
24. David Hanes, Gonzalo Salgueiro, Patrick Grossetete, Robert Barton, and Jerome Henry. *IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things*. Cisco Press, 1st edition, 2017.
25. M. Thomson Ed. J. Iyengar, Ed. QUIC: A UDP-Based Multiplexed and Secure Transport draft-ietf-quic-transport-27. Internet-Draft 27, RFC Editor, February 2020.
26. M. Isomaki B. Patil Z. Shelby C. Gomez J. Nieminen, T. Savolainen. IPv6 over BLUETOOTH(R) Low Energy. RFC 7668, RFC Editor, September 2015.
27. Keith Ross James Kurose. *Computer Networking: A Top-Down Approach*. Pearson, 7 edition, 2016.
28. Jamil Y Khan and Mehmet R Yuce. *Internet of Things (IoT): Systems and Applications*. CRC Press, 2019.
29. Bruce S. Davie Larry L. Peterson. *Computer Networks: A Systems Approach*. Morgan Kaufmann, 5 edition, 2011.
30. Perry Lea. *Internet of Things for Architects: Architecting IoT solutions by implementing sensors, communication infrastructure, edge computing, analytics, and security*. Packt Publishing Ltd, 2018.
31. Brian Lebiednik, Aman Mangal, and Niharika Tiwari. A survey and evaluation of data center network topologies, 2016.
32. M. Thomson Ed. M. Belshe, R. Peon. Hypertext Transfer Protocol Version 2 (HTTP/2). RFC 7540, RFC Editor, May 2015.
33. Ed. M. Bishop. Hypertext Transfer Protocol Version 3 (HTTP/3) draft-ietf-quic-http-27. Internet-Draft 27, RFC Editor, February 2020.

34. Patrick Maigron. World - autonomous system number statistics - sorted by number. `https://www-public.imtbs-tsp.eu/~maigron/RIR_Stats/RIR_Delegations/World/ASN-ByNb.html`, 2020. [Online; pristupano 21.1.2020.].
35. Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, March 2008.
36. P. Mockapetris. DOMAIN NAMES - CONCEPTS AND FACILITIES. RFC 1034, RFC Editor, November 1987.
37. P. Mockapetris. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. RFC 1035, RFC Editor, November 1987.
38. RIPE NCC. The ripe ncc has run out of ipv4 addresses. `https://www.ripe.net/publications/news/about-ripe-ncc-and-ripe/the-ripe-ncc-has-run-out-of-ipv4-addresses`, 2019. [Online; pristupano 20.1.2020.].
39. P. McManus P. Hoffman. DNS Queries over HTTPS (DoH). RFC 8484, RFC Editor, October 2018.
40. K. Egevang P. Srisuresh. Traditional IP Network Address Translator (Traditional NAT). RFC 3022, RFC Editor, January 2001.
41. J. Postel. User Datagram Protocol. RFC 768, RFC Editor, August 1980.
42. J. Postel. INTERNET PROTOCOL. RFC 791, RFC Editor, September 1981.
43. J. Postel. TRANSMISSION CONTROL PROTOCOL. RFC 793, RFC Editor, September 1981.
44. Nissanka B. Priyantha, Aman Kansal, Michel Goraczko, and Feng Zhao. Tiny web services: Design and implementation of interoperable and evolvable sensor networks. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, SenSys '08, page 253–266, New York, NY, USA, 2008. Association for Computing Machinery.
45. D. Massey S. Rose R. Arends. R. Austein, M. Larson. DNS Security Introduction and Requirements. RFC 4033, RFC Editor, March 2005.
46. J. Reschke Ed. R. Fielding, Ed. Hypertext Transfer Protocol (HTTP/1.1): Authentication. RFC 7235, RFC Editor, June 2014.
47. J. Reschke Ed. R. Fielding, Ed. Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests. RFC 7232, RFC Editor, June 2014.
48. J. Reschke Ed. R. Fielding, Ed. Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. RFC 7230, RFC Editor, June 2014.
49. J. Reschke Ed. R. Fielding, Ed. Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. RFC 7231, RFC Editor, June 2014.
50. M. Nottingham Ed. J. Reschke Ed. R. Fielding, Ed. Hypertext Transfer Protocol (HTTP/1.1): Caching. RFC 7234, RFC Editor, June 2014.
51. Y. Lafon Ed. J. Reschke Ed. R. Fielding, Ed. Hypertext Transfer Protocol (HTTP/1.1): Range Requests. RFC 7233, RFC Editor, June 2014.
52. H. Ruellan R. Peon. HPACK: Header Compression for HTTP/2. RFC 7541, RFC Editor, May 2015.
53. E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, RFC Editor, May 2015.
54. Ethan Banks Russ White. *Computer Networking Problems and Solutions: An innovative approach to building resilient, modern networks.* Addison-Wesley Professional, 2018.

55. P. Balasubramanian L. Eggert G. Judd S. Bensley, D. Thaler. Data Center TCP (DCTCP): TCP Congestion Control for Data Centers. RFC 8257, RFC Editor, October 2017.
56. R. Hinden S. Deering. Internet Protocol, Version 6 (IPv6) Specification. RFC 8200, RFC Editor, July 2017.
57. Sandvine. The Global Internet Phenomena Report. Technical report, September 2019.
58. C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948.
59. William Stallings. *Foundations of modern networking: SDN, NFV, QoE, IoT, and Cloud*. Addison-Wesley Professional, 2015.
60. Domain Name Stat. Domain name registration's statistics. `https://domainnamestat.com/statistics/overview`, 2020. [Online; pristupano 6.1.2020.].
61. L. Masinter T. Berners-Lee, R. Fielding. Uniform Resource Identifier (URI): Generic Syntax. RFC 3986, RFC Editor, January 2005.
62. A. Brandt J. Hui R. Kelsey P. Levis K. Pister R. Struik JP. Vasseur R. Alexander T. Winter, P. Thubert. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550, RFC Editor, March 2012.
63. Subir Varma. Chapter 7 - congestion control in data center networks. In Subir Varma, editor, *Internet Congestion Control*, pages 205 – 230. Morgan Kaufmann, Boston, 2015.
64. W3Techs. Usage statistics of http/2 for websites. `https://w3techs.com/technologies/details/ce-http2/all/all`, 2019. [Online; pristupano 16.10.2019.].
65. D. Karrenberg G. J. de Groot E. Lear Y. Rekhter, B. Moskowitz. Address Allocation for Private Internets. RFC 1918, RFC Editor, February 1996.
66. Dogan Yazar and Adam Dunkels. Efficient application integration in ip-based sensor networks. In *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, BuildSys '09, page 43–48, New York, NY, USA, 2009. Association for Computing Machinery.
67. J. Heidemann A. Mankin D. Wessels P. Hoffman Z. Hu, L. Zhu. Specification for DNS over Transport Layer Security (TLS). RFC 7858, RFC Editor, May 2016.
68. C. Bormann Z. Shelby, K. Hartke. The Constrained Application Protocol (CoAP). RFC 7252, RFC Editor, June 2014.