

# Računarske mreže – odgovori na pitanja

Februar 2019.

Vladana Đorđevic

Dragica Andđelković

Marija Filipović

Đorđe Dimović

Đorđe Stanojević

Lazar Bojanić



## **1. Komponente mreze, tipovi veza, primeri mreza, mreze prema dimenziji, medjumreze.**

Primeri upotreba:

1.Poslovna:

razmena datoteka, deljeni resursi (npr stampaci), sredstvo komunikacije(e-posta, video konferencije), poslovne transakcije elektronskim putem(isporucioci I korisnici robe), poslovanje s potrosacima preko Interneta (e-trgovina)...

2.Kucne:

Prisupanje udaljenim informacijama, komunikacija, interaktivna zabava, e-trgovina..

3.Mobilne:

Pozivi, SMS, igrice, mape, pristup informacijama...

### **Komunikacija:**

VoIP (pozivi preko interneta)

video konferencije

cetovanje

socijalne mreze

→ potreban je brz pristup, odnosno malo kasnjenje za ovakve primene

### **Deljenje resursa:**

Vise korisnika pristupa istim uredajajima I servisima (3D stampac, indeks pretrage, racunari na zahtev(cloud))

Efektivnija upotreba od posvecenih resursa (kada se gleda po korisniku)

-Mrezni protok se deli statistickim multipleksiranjem

Statisticko multipleksiranje – deljenje mreznog protoka medju korisnicima na osnovu zahteva

-korisno, jer korisnici najcesce ne prenose nista

-funkcija prenosa kroz vreme je vrlo skokovita

### **Dostavljanje sadrzaja:**

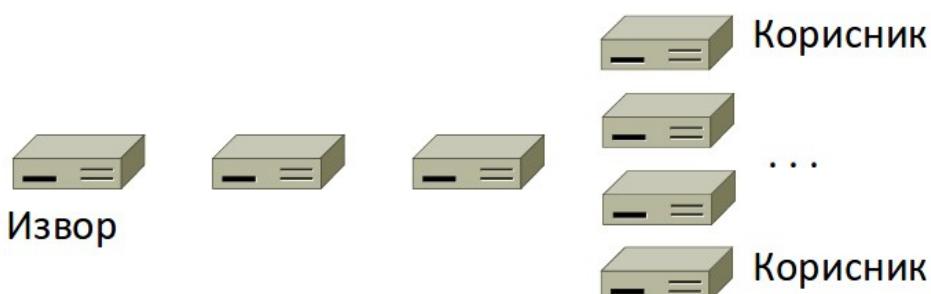
Isti sadrzaj vecem broju korisnika-video materijal, pesme, aplikacije, veb stranice...

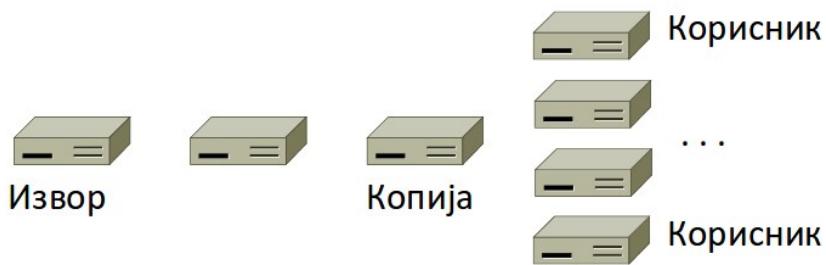
Efikasnije nego slanje kopije svakom korisniku pojedinacno

-upotreba distribuiranih replika sirom mreze

Slanje sadrzaja sa izvora do 4 korisnika uzima 12 "mreznih skokova" (network hops)

Slanje sa pametno pozicioniranom replikom uzima  $4 + 2 = 6$  skokova





### Komunikacija medju racunarima:

Racunari mogu komunicirati jedan s drugim  
 -elektronsko poslovanje, rezervacije karata  
 → Omogucava automatsku obradu informacija nad nezavisnim sistemima

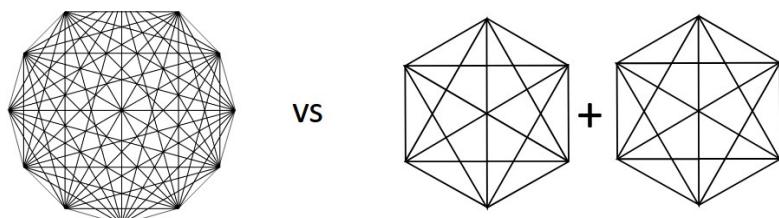
### Povezivanje racunara sa uredjajima:

Prikupljanje podataka sa senzora, manipulacija uredjajima  
 -kamere, lokacije na mobilnim uredjajima, detektori pokreta...  
 Ovo je područje primene u povoju, Internet za stvari (IoT – Internet of things)

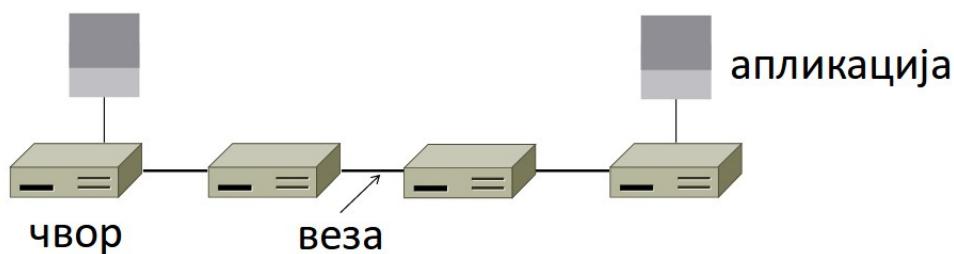
### Vrednost povezivanja:

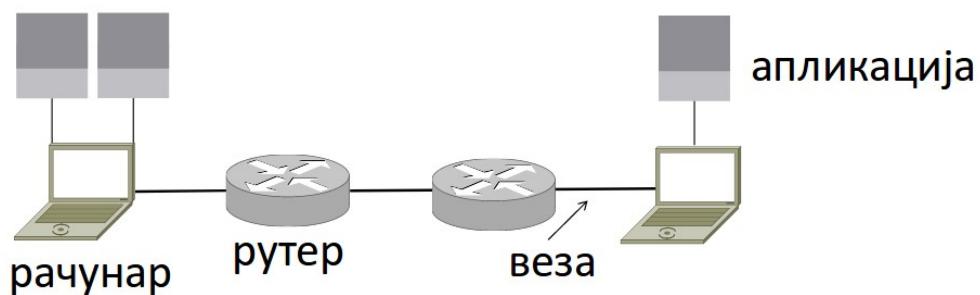
Metkalfov zakon~1980:

- Vrednost mreze sa  $N$  cvorova je proporcionalna vrednosti  $N^2$
- Velika mreza je vrednija nego vise malih sa istim ukupnim brojem cvorova  
 npr: jedna veca struktura od 12 cvorova je vise povezana nego druga struktura od dve mreze sa po 6 cvorova



### Komponente mreže:





Компонента - називи	Функција	Пример
<u>Апликација</u> , корисник, ...	Користи мрежу	Skype, iTunes, Amazon
<u>Рачунар</u> , или завршни чвор, извор, уређај ...	Подржава апликацију	Лаптоп, мобилни телефон, стони рачунар
<u>Рутер</u> , или усмеривач, средишњи чвор	Прослеђује поруке између чворова	Приступна тачка, кабловски/DSL модем
<u>Веза</u> , или канал	Спаја чворове	Жичани, бежични

#### Tipovi veza:

1. Simpleks – jedan smer

Komunikacija je jednosmerna. Jedan od dva uređaja uvek salje, a drugi uvek samo prima podatke.

2. Polu-dupleks – u oba smera

Oba uređaja mogu da salju podatke, ali ne u isto vreme. Dok jedan uređaj radi kao predajnik, drugi može kao prijemnik I obrnuto. Primer: Voki-toki radio

3. Puni dupleks – u oba smera istovremeno

Prenos podataka između dva uređaja može se obavljati simultano u oba smera (oba uređaja mogu u isto vreme da salju I primaju podatke). Signali koji se prenose u razlicitim smerovima dele raspolozivi kapacitet linije. Primer: telefonska mreža

#### Bezicna veza:

- Poruka se emituje, prihvataju je svi cvorovi u opsegu.

- Mesanje signala

#### Mala mreža:

- Povezuje nekoliko računara

#### Mrežni hardver:

Ne postoji opstevihrvacen sistem klasifikacije računarskih mreža, ali se ističu dva njihova najvažnija aspekta: tehnologija prenosa podataka I velicina.

Postoje dva tipa najcesce koriscenih tehnologija za prenos podataka:

1. Veza za neusmereno (difuzno) emitovanje
2. Veze od tacke do tacke

### 1. Mreze sa neusmerenim (difuznim) emitovanjem (broadcast networks)

Imaju jedinstven komunikacioni kanal koji dele svi umrezeni racunari. Kratke poruke (paketi), koje emituje bilo koji racunar, primaju svi ostali umrezeni racunari. Polje za adresu unutar paketa odredjuje primaoca (racunar kome je paket namenjen). Kada racunar primi paket I utvrdi da je namenjen njemu, on ga obradjuje; ako utvrdi da je namenjen nekom drugom racunaru, jednostavno ga zanemaruje. Zamislimo analogiju: poziv preko aerodromsog razglosa da se svi putnici na letu 644 upute ka izlazu 12. Sistemi za difuzno emitovanje najcesce imaju mogucnost da pakete usmere na sva odredista pomocu specijalnog koda u adresnom polju. Kada se paket s takvim kodom emituje u mrezu, prima ga I obradjuje svaki umrezeni racunar. Opisani rezim rada naziva se **neusmereno (difuzno) emitovanje (broadcasting)**. Neki takvi sistemi podrzavaju I usmeravanje paketa samo na odredjeni podskup racunara, sto se ponekad naziva **I visesmerno emitovanje (multicasting)**. Jedna mogucnost je da se u adresnom polju rezervise jedan bit za oznacavanje visesmernog emitovanja. Preostalih n-1 bitova adrese mogu da sadrze broj grupe. Svaki racunar moze da se ukljuci u jednu ili vise grupe. Kada se paket posalje odredjenoj grupi, on se isporucuje svim racunarima ukljucenim u tu grupu.

### 2. Mreze od tacke do tacke (point-to-point networks)

Ove mreze sadrze brojne veze izmedju pojedinih parova racunara. Da bi stigao od polazista do odredista, paket mozda mora da prodje kroz jedan ili vise drugih racunara. Cesto postoji vise putanja razlicite duzine, pa je pronalazenje optimalne putanje vazna stavka kod ovog tipa mreza. Uglavnom se u manjim, geografski lokalizovanim mrezama koristi difuzno emitovanje, dok vece mreze uglavnom koriste povezivanje od tacke do tacke. Prenos poruka od tacke do tacke (od jednog posiljaoca do jednog primaoca) cesto se naziva **jednosmerno emitovanje (unicasting)**.

## Primeri mreza:

WiFi (802.11), Poslovne/Ethernet, ISP (Internet Service Provider), Kablovska / DSL, Mobilna telefonija (2G, 3G, 4G), Bluetooth, telefon, sateliti...

## Podela mreza po velicini:

### Licne mreze (Personal Area Network, PAN)

-Mreze namenjene jednoj osobi. Primer: mreza koja pozveuje racunar s misem, tastaturom, stampacem.

### Lokalne mreze (Local Area Networks, LAN)

-Privatne mreze unutar jedne zgrade ili jednog organizacionog područja raspona do 5km. Koriste se za povezivanje licnih racunara I radnih stanica u kancelarijama radi zajednickog koriscenja resursa I razmene informacija.

### Gradske mreze (Metropolitan Area Network, MAN)

- Pokriva gradsko područje  
- Najpoznatiji primer je kablovska televizija

### Mreza sirokog područja (Wide Area Network, WAN)

- Pokriva veliko geografsko područje, npr citavu državu ili cak kontinent

Димензија	Тип	Пример
Непосредна близина	<u>PAN</u> (Personal Area Network)	Bluetooth
Зграда	<u>LAN</u> (Local Area Network)	WiFi, Ethernet
Град	<u>MAN</u> (Metropolitan Area Network)	Кабловска, DSL
Држава	<u>WAN</u> (Wide Area Network)	Велики ISP, нпр. Телеком, SBB
Планета	Internet (мрежа свих мрежа)	Интернет

### Medjumreze:

- Medjumreza, ili internet (namerno malo pocetno slovo), se dobija povezivanjem vise razlicitih mreza
- Internet (veliko slovo) je internet koji svi koristimo – samo jedna posebna medjumreza
- Cest oblik medjumreze je WAN koji povezuje vise LAN-ova.
- Medjumreza se obrazuje kada se medjusobno povezu jasno ogranicene mreze (npr. Ako se u razlicitim delovima mreze koriste razlicite tehnologije, difuzno emitovanje I prenos od tacke do tacke, onda se verovatno ne radi o jednoj, vec o vise medjusobno povezanih mreza)
- \* podmreza
  - ima najvise smisla unutar regionalne mreze, gde se odnosi na skup usmerivaca I komunikacionih linija u vlasnistvu operatora mreze. Npr kod telefonskog sistema, linije I oprema koje poseduje I odrzava telefonska kompanija predstavljaju podmrezu. Sami telefonski aparati nisu deo podmreze. Kombinacija podmreze I telefonskih aparata obrazuje mrezu. (Ne mesati pojmove podmreza, mreza I medjumreza)

### Ključni interfejsi:

#### 1. Mreza-aplikacija interfejsi

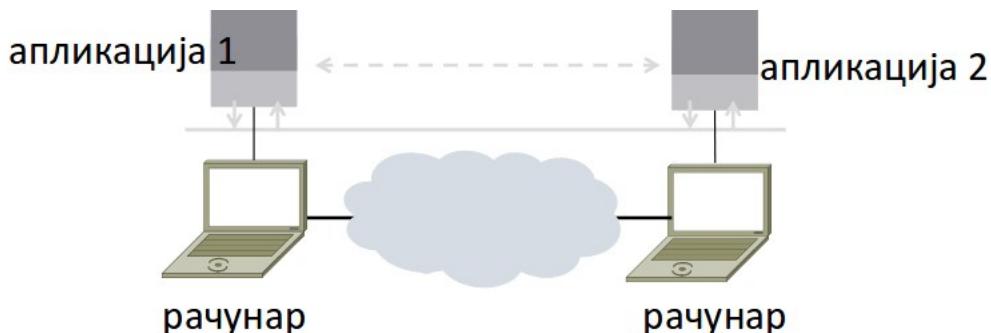
Definisu kako aplikacije koriste mrezu. Posmatramo aplikacije kao da direktno komuniciraju. Zanemarujemo detalje mreze

Primer primene: Jednostavni klijent-server scenario:

Klijentska aplikacija salje zahtev serverskoj aplikaciji. Serverska aplikacija vraca odgovor.

Ovo je polazna osnova za mnoge primene

- Prenos datoteka: posalji naziv, prihvati datoteku
- Pretrazivanje: posalji URL, prihvati stranicu



Soket API (Application programming interface)

Jednostavna apstrakcija za upotrebu mreze.

- Mrezni API koji se koristi za pisanje svih Internet aplikacija
- Deo zvih poznatijih operativnih sistema

Podrzava dva tipa mrežnih servisa

- Tokovi podataka: pouzdano slanje toka bajtova

- Datagram servis: nepouzdano slanje odvojenih poruka (bitno je da poruka stigne na cilj s visokom verovatnocom ali bez garancije. Nepouzdana usluga bez uspostavljanja direktnе veze tj usluga bez potvrđivanja prijema cesto se naziva usluga datagrama).

Soketi omogucavaju da se aplikacija prikljuci na lokalnu mrežu preko razlicitih portova.

Операција	Значење
SOCKET	Креира нову комуникациону тачку
BIND	Придружује сокету локалну адресу (сервер)
LISTEN	Означава да је сокет спреман да прихвата конекције (сервер)
ACCEPT	Пасивно прихвата и успоставља долазну конекцију (сервер)
CONNECT	Активно покушава да оствари конекцију (клијент)
SEND	Шаље податке преко успостављене конекције
RECEIVE	Прихвата податке преко успостављене конекције
CLOSE	Затвара конекцију

## Клијентски програм (оквирно)

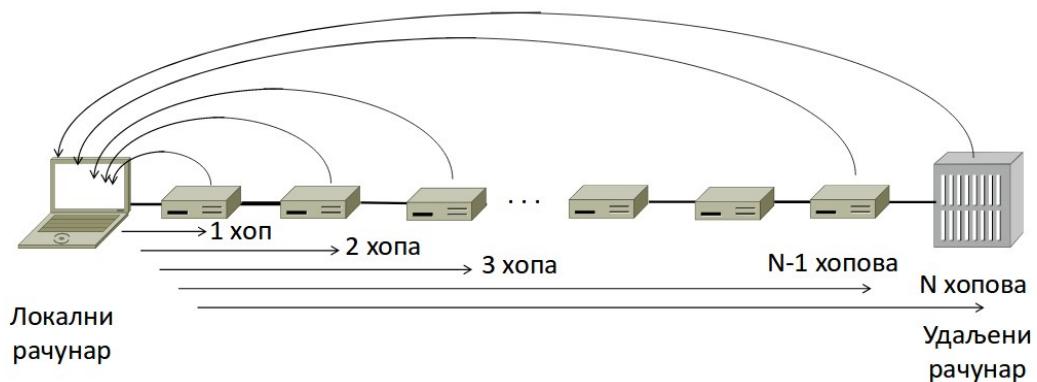
```
socket()  
// прављење клијентског сокета задавањем циљне IP адресе и порта  
//нпр. www.example.com:80  
connect() // повезивање на сервер [блокирање]  
...  
send() // слање захтева  
recv() // чекање одговора [блокирање]  
... // нешто радимо са одговором...  
close() // готово, прекид – не гаси серверски сокет!
```

## Серверски програм (оквирно)

```
socket()  
// прављење серверског сокета – адреса се имплицитно одређује, јер сервер  
// већ има додељену јавну IP адресу  
bind()      // придрживање порта сокету  
listen()     // припрема за прихватање долазних конекција  
accept()    // чекање на долазну конекцију [блокирање]  
...  
recv()      // чекање на захтев за прихваћену конекцију  
...  
send()      // враћање одговора  
close()     // гашење сервера – прекид свих долазних конекција
```

### 2.Mreza-mreza interfejsi

- Definisu kako cvorovi “saradjuju”
- Aplikacije komuniciraju bez realne predstave sta je unutar mreze, ali ako nas zanima sta je unutra.. Komanda traceroute :
  - moze da “zaviri” u unutrasnjost mreze
  - ispituje uzastopne hopove kako bi rekonstruisao celi putanje



## 2. Protokoli I slojevi.

Sta sve mreza radi za aplikacije:

- Pravi I prekida konekciju
- Pronalazi putanju za transfer podataka
- Pouzdano salje podatke
- Salje podatke proizvoljne velicine
- Brzina slanja se prilagodjava mogucnostima mreze
- Deli protok medju korisnicima
- Omogucava siguran prenos tokom tranzita
- Omogucava novo dodavanje racunara I uredjaja (cvorova)
- ...

Da bi radila sve ovo, potreban je neki vid modularnosti kako bi se savladala kompleksnost I podrzala ponovno koriscenje mnogih funkcionalnosti (reuse).

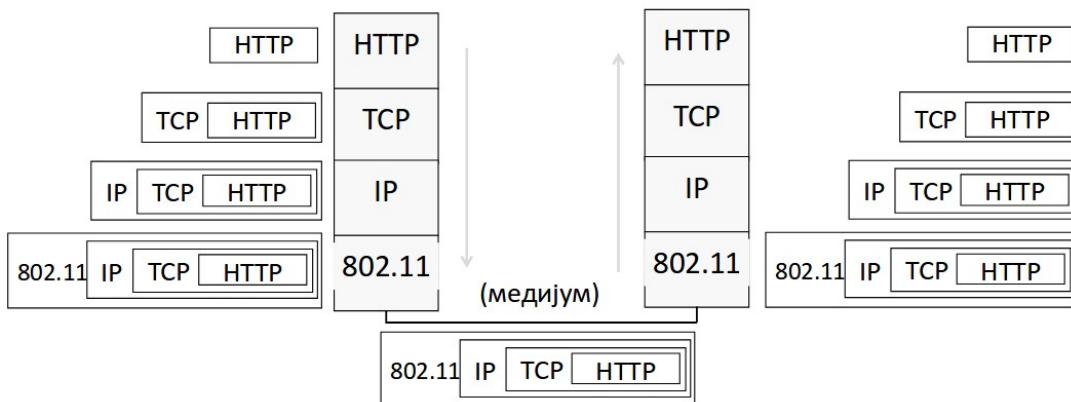
Protokoli I slojevi su glavni mehanizam struktuiranja koji mrezi daje modularnost.

- Svaka instance protokola komunicira vizuelno samo sa svojim parnjakom (peer) upotrebom dogovorenih metoda
- U stvarnosti, oni ne komuniciraju direktno, vec svaka instance koriti usluge (services) sloja koji je ispod

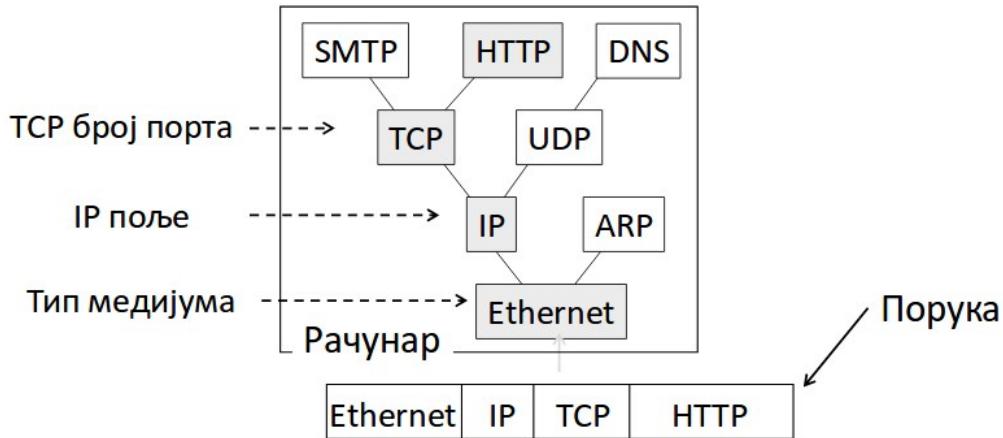
Poznatiji protokoli : TCP, IP, 802.11, Ethernet, HTTP, SSL, DNS...

**Enkapsulacija** je mehanizam slaganja slojeva protokola.

- Nizi sloj pravi omotac oko sadrzaja viseg sloja I dodaje svoje sopstvene informacije, tako pravi novu poruku za isporuku
- Poput slanja poste u koverti, postari nemaju pristup unutrasnjosti koverte



Vidimo na slici kako kod posiljaoca poruke svaki sloj odozgo nanize dodaje svoje zaglavljive poruci. Kod primaoca se kako se poruka penje na gore otklanja zaglavljive odredjenog sloja pre nego sto se poruka prosledi visem sloju.



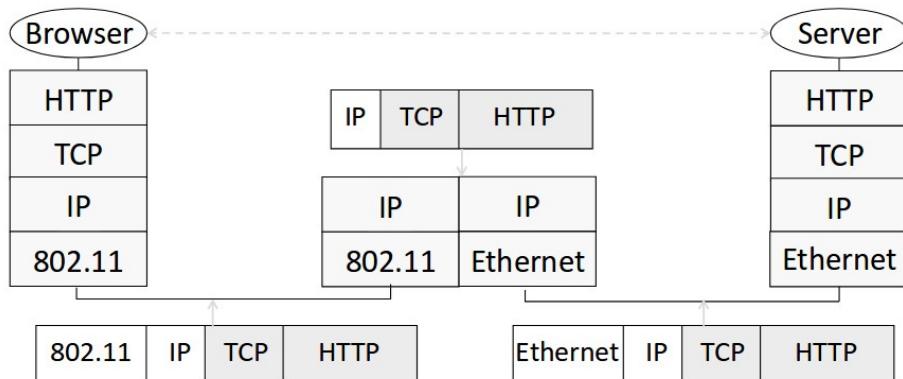
Kako poruka zna kuda treba da putuje? Tako sto svako zaglavje koje se dodaje na odredjenom sloju sadrzi upravljacke informacije i informaciju o tome gde je poruka usmerena. Npr. Na slici iznad, u okviru zaglavla od Ethernet nalazi se informacija da poruka treba da ide u IP, u IP zaglavlu data je informacija da nakon toga poruka ide u TCP, itd... Na taj nacin se obezbedi da se poruka kod primaoca kreće nagore ispravnim putem (da ne zaluta u neki pogresni protokol).

Ove informacije u okviru svakog sloja protokola se nazivaju demultiplexing kljucevi, zato sto nam pomazu da otklonimo **demultiplexiranje** I odaberemo ispravnu putanju.

Poruka koja pristize prosledjuje se protokolima koje koristi, a to se radi putem kljuceva na pocetku.

Prednosti raslojavanja:

- Prikivanje informacija I ponovna upotreba.
- Povezivanje razlicitih sistema (?)
- Konverzija poruka – tabele preslikavanja protokola



Ovde vidimo kako se radi konverzija poruka preko tabele preslikavanja. Poruka se prosledi iz 802.11 u jer ta dva mogu medjusobno da komuniciraju 802.11. Zatim navise IP sloju, ostaje u IP sloju I onda se spusti u Ethernet. Dva Ethernet sloja mogu medjusobno virtuelno da komuniciraju. Zaglavla koja su zatamnjena (TCP, HTTP) se ne menjaju tokom prolaza kroz mrežu, menjaju se samo 802.11 I Ethernet zaglavla. Ova lista protokola ispod Browser-a naziva se protokol stek.

Mane raslojavanja:

- Povecani troškovi memorije i obrade (overhead)
  - Manje bitno za duze poruke
- Prikivanje informacija
  - Generalno korisno, ali neke aplikacije možda npr. zele da znaju da li se podaci prenose putem kabla ili bezicno, a ne mogu to da saznaju

Dakle:

Da bi projektovanje bilo jednostavnije, mreze se organizuju kao skup **slojeva**. Svaki sloj nudi odredjene usluge visim slojevima, ne opretecujuci ih detaljima njihove realizacije. Svaki sloj je u izvesnom smislu virtualna masina koja nudi usluge sloju iznad sebe (u programiranju bi to bilo realizovano apstraktним tipovima podataka, kapsuliranjem podataka, I OOP-om).

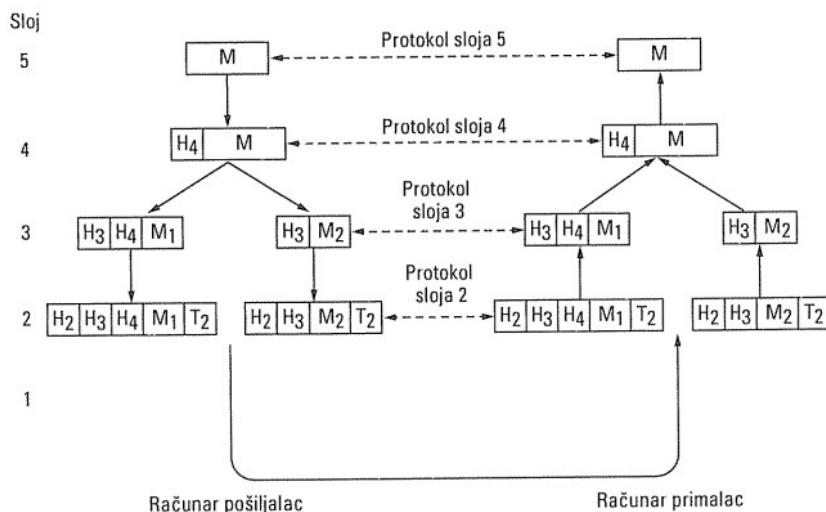
Sloj n na jednom racunaru komunicira sa slojem n na drugom racunaru. Pravila i konvencije koji se koriste u komuniciranju poznati su pod zajednickim nazivom **protokol** sloja n. Protokol predstavlja dogovor izmedju dve jedinke o tome kako treba da tece njihova medjusobna komunikacija.

Za elemente odgovarajucih slojeva na razlicitim racunarima se kaze da su ravnopravni (peers).

Ravnopravni elementi mogu da budu procesi, hardverski uredjaji...Protokolarno komuniciranje se odvija izmedju ravnopravnih strana.

U stvarnosti, nikada se podaci ne prenose direktno od sloja n na jednom racunaru ka sloju n na drugom racunaru, vec svaki sloj prosledjuje podatke I upravljacke informacije sloju neposredno ispod sebe, sve dok se ne dostigne najnizi sloj. Ispod sloja 1 je fizicki medijum kroz koji se stvarno odvija komunikacija. Izmedju svaka dva susedna sloja se nalazi interfejs, koji odreduje osnovne operacije I usluge koje donji sloj nudi gornjem. Skup slojeva I protokola se naziva arhitektura mreze. Lista protokola koju koristi odredjeni sistem (jedan protokol po sloju) naziva se **skup protokola (protocol stack)**. Npr. Skup protokola koji koristi Internet pregledac (Browser kao sa slikama gore) na racunaru koji je putem WiFi povezan na Internet.

Kazemo kako su protokoli "horizontalni", a slojevi "vertikalni". Treba zapaziti razliku izmedju virtuelne I stvarne komunikacije. Ravnopravni procesi u sloju 4 ponasaju se kao da medjusobno



komuniciraju "horizontalno" pomocu protokola sloja 4. Svaki od njih moze da ima neke procedure tipa PosaljiDrugomRacunaru ili PreuzmiSaDrugogRacunara, iako te procedure u stvari komuniciraju sa nizim slojevima preko interfejsa ¾, a ne direktno s drugim racunaram.

Dok paket putuje od polazista na dole, svaki sloj moze da mu dodaje zaglavlj, da deli paket na delove, ukoliko on prevazilazi dozvoljenu velicinu. Na racunaru koji prima poruku, ona se kreće uzlazno od jednog do drugog sloja, pri cemu se u svakom sloju s nje uklanja odgovarajuce zaglavlj. Nijedno zaglavlj slojeva ispod sloja n ne dolazi do sloja n.

Kada je nepogodno ili skupo da se za svaki par procesa koji medjusobno komuniciraju uspostavlja zasebna veza, odgovarajuci sloj moze da istu vezu upotrebi za vise istovremenih, nezavisnih konverzacija. Sve dok je ovo **multipleksiranje I demultipleksiranje** nevidljivo, moze ga koristiti svaki sloj. Multipleksiranje je potrebno na primer u fizickom sloju, gde se saobracaj za sve veze mora preneti preko najvise nekoliko fizickih linija.

### **3. Referentni modeli protokola I slojeva, jedinice podataka, organizacije za standarde.**

Postoji mnogo proizvodjaca I prodavaca mreza, a svaki od njih ima sopstveni stav o tome kako stvari treba da izgledaju. Bez koordinacije nastao bi potpun haos. Jedini izlaz je dogovaranje o nekakvim mrežnim standardima. Standardni omogucavaju da razliciti racunari medjusobno komuniciraju, I takodje prosiruju trziste proizvoda koji su usaglaseni sa standardom.

Standardi se dele u 2 kategorije:

**De facto** – cinjenicki, fakticki, standardi koji su prihvacioci bez prethodnog planiranja. IBM PC I njegovi naslednici predstavljaju ove standarde zato sto su desetine proizvodjaca precizno iskopirali IBMove proizvode. Slicno, UNIX je de facto standard za operativne sisteme u univerzitetskim racunarskim centrima.

**De jure** – zakonski standardi predstavljaju formalne, zakonske standarde koje je objavilo neko telo ovlašćeno za standardizaciju.

Koju funkcionalnost implementira neki sloj?

- To je ključno pitanje dizajna modela
- Referentni modeli odgovaraju na ovakva pitanja

#### **OSI Referentni model sa 7 slojeva (Open systems interconnection)**

Internacionalni standard za povezivanje sistema. Jako uticajan, ali nije koriscen u praksi.

7	<b>Application</b>	→ funkcije potrebne korisniku, rad sa porukama
6	<b>Presentation</b>	→ konverzija za razlicite reprezentacije
5	<b>Session</b>	→ upravljanje sesijom (povezivanje vise komponenti)
4	<b>Transport</b>	→ dostavljanje segmenata (segmentacija, potvrđivanje)
3	<b>Network</b>	→ adresiranje, rutiranje paketa, kontrola saobracaja
2	<b>Data link</b>	→ slanje okvira (skupova podataka)
1	<b>Physical</b>	→ slanje bitova putem realnih fizickih kanala

#### **Sloj aplikacija:**

Sadrzi vise protokola najcesce potrebnih korisnicima. Npr HTTP protokol

#### **Sloj prezentacije:**

Bavi se sintaksom I semantikom prenetih informacija. Da bi racunari koji podatke predstavljaju na razlicit nacin mogli da komuniciraju, strukture podataka koje se prenose mogu se definisati na apstraktan nacin i standardno kodirati u cilju prenosa. Sloj prezentacije obraduje te apstraktne strukture.

#### **Sloj sesije:**

Omgucava korisnicima na razlicitim racunarima da medjusobno uspostave sesiju. Sesije nude razlicite usluge, kao npr upravljanje dijalogom - vodjenje racuna o tome na koga je red da salje poruke, rad sa zetonima – sprecavanje ucesnika da istovremeno pokrenu istu kriticnu operaciju, i sinhronizovanje – proveravanje dugackog niza podataka tokom prenosa da bi se omogucilo nastavljanje od tacke prekida u slucaju pada sistema.

#### **Transportni sloj:**

Prihvata podatke odozgo, po potrebi ih razvrstava u manje grupe I prosledjuje ih mrežnom sloju, obezbedjujuci da svi delovi ispravno stignu na odrediste. I to sve treba da uradi efikasno, uz skrivanja detalja hardvera.

#### **Mrezni sloj:**

Pri njegovom projektovanju kljucno je odrediti kako se paketi upucuju od izvora ka odredistu. Putanje se mogu zasnivati na staticnim tabelama koje su ugradjene u mrežu, mogu se utvrdjivati na pocetku svake konverzacije, a mogu se i odredjivati dinamicki za svaki paket. Mrezni sloj kontrolise i zagusenja saobracaja do kojih moze doći kada se previse paketa istovremeno nalazi u saobracaju.

#### **Sloj veze:**

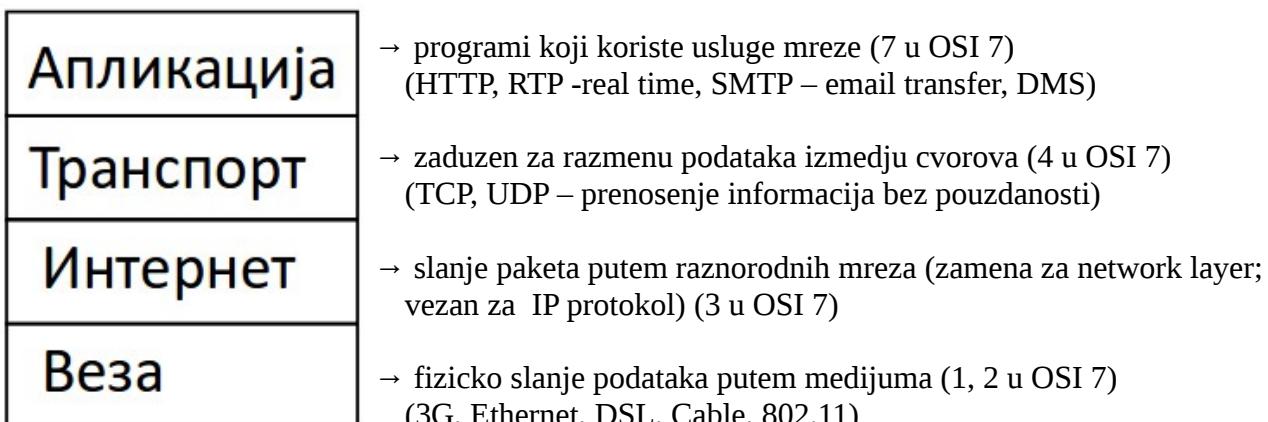
Posiljalac ulazne podatke deli na okvire podataka, I salje ih jedan za drugim. Ako je usluga pouzdana, primalac potvrđuje ispravan prijem svakog okvira saljuci okvir za potvrdu. Problem: neusaglasenost brzine slanja i primanja podataka.

#### **Fizicki sloj:**

Ima ulogu da dobijeni niz bitova prenese duž komunikacionog kanala. Razmilja se o tome koliki napon treba da predstavlja 1 a koliki 0, da li se prenos može istovremeno obavljati u oba smera, kako se na pocetku uspostavlja veza, kako se prekida kada oba učesnika obave poslove...

### **Internet referentni model**

Cetvoroslojni model zasnovan na praksi.



5, 6 sloj - iz OSI modela se primetilo da ovi slojevi donose malu korist vecini aplikacija.

#### **Sloj aplikacija:**

Sadrži sve protokole viseg nivoa. Na pocetku su to bili TELNET – omogućava korisniku da se sa svog računara daljinski prijavi na drugi računar i na njemu radi, FTP – protokol za prenos datoteka koji omogućava efikasan prenos s jednog računara na drugi, SMTP – protokol za elektronsku postu.. Tokom vremena ovom sloju dodati su drugi protokoli: DNS – sistem imenovanja domena za preslikavanje imena računara u njihove mrežne adrese, HTTP – protokol za preuzimanje strana s World Wide Web-a...

#### **Transportni sloj:**

Namenjen za konverzaciju izmedju ravnopravnih procesa na izvornom i odredisnom računaru, isto kao kod transportnog OSI sloja. Ovde su definisana dva protokola.

Protokol za upravljanje (Transmission Control Protocol, TCP) predstavlja pouzdan protokol sa uspostavljanjem direktnе veze, koji omogućava da se tok bajtova potekao s jednog računara bez greske do bilo kog odredista u medjumrezi. On deli pocetni tok bajtova na zasebne poruke i svaku prosledjuje medjumrežnom sloju. Prihvati TCP proces na odredistu uređuje primljene poruke i od njih ponovo obrazuje tok bajtova. TCP upravlja i tokom podataka tako da brzi posiljalac ne može da zatrpa sporog primaoca s vise poruka nego sto ovaj može da obradi.

Protokol za korisnicke datagrame (User Datagram Protocol, **UDP**) predstavlja nepouzdan protokol bez uspostavljanja direktne veze, namenjen aplikacijama koje same određuju svoje pakete i upravljaju tokom podataka (umesto TCP). Koristi se kod aplikacija kod kojih hitnost isporuke ima vecu prednost nad tacnoscu, npr prenos govora ili videa, kao i za jednostavne upite klijentsko-serviskog tipa.

#### **Medjumrezni sloj (Internet):**

Medjumrezni sloj definise zvanicni format paketa i tzv. Internet protokol (**IP**). Zadatak je da IP pakete isporuci tamo gde treba da stignu. Paketi mogu da stignu na odrediste drugacijim redosledom, pa je zadatak visih slojeva da ih dovedu u red. Najveći problemi su usmeravanje i izbegavanje zagusenja.

#### **Sloj za povezivanje racunara s mrezom (Veza):**

Racunar mora da se poveze sa mrezom pomocu nekog protokola kako bi mogao da joj salje IP pakete. Sam protokol nije definisan I menja se od racunara do racunara, od mreže do mreze.

Fokus kod svih ovih protokola je na **KOMPATIBILNOSTI!**

Орг.	Област	Примери стандарда
ITU	Телекомуникације	G.992, ADSL H.264, MPEG4
IEEE	Комуникације	802.3, Ethernet 802.11, WiFi
IETF	Интернет	RFC 2616, HTTP/1.1 RFC 1034/1035, DNS
W3C	Веб	HTML5 стандард CSS стандард

#### **Jedinice podataka u razlicitim slojevima:**

Aplikativni → poruka

Transportni → segment

Mrezni → paket

Sloj veze → okvir

Fizicki → bit

#### **Neki uredjaji u mrezi:**

Hab (razvodnik) – ponavlja fizicki signal na sve izlaze

Svic (skretnica) – usmerava pakete samo onima kojima su potrebni

Ruter (usmerivac) – usmerava pakete, ali vodi racuna I o dobrim putanjama

#### **4. Fizicki sloj, uloga, pojednostavljeni model, kasnjenja, BDP, primeri.**

Koristimo hibridni pristup sa 5 nivoa:

Aplikativni  
Transportni  
Mrežni  
Sloj veze  
Fizicki

##### **Domen fizickog nivoa**

- Tice se slanja poruka putem komunikacionog kanala
  - Zice salju analogni (fizicki) signal
  - Mi zelimo da saljemo bitove, koji su digitalni

##### **Pojednostavljeni model**

- Uopšteni fizicki kanal – karakteristike:
  - Protok (brzina, kapacitet) meren kao bitovi po sekundi b/s
  - Kasnjenje u sekundama
- Druge bitne karakteristike:
  - Da li kanal emituje ili ne, raspodela verovatnoca gresaka...

##### **Kasnjenje poruka**

- Kasnjenje podrazumeva vreme potrebno da poruka stigne na ciljnu adresu
  - Kasnjenje prenosa (transmission delay): vreme potrebno da se M-bitna poruka postavi na komunikacioni kanal.  
 $T\text{-delay} = M \text{ (b)} / B \text{ (b/s)} = M/B \text{ s (sekundi)}$   
B – protok (rate)
    - Kasnjenje propagacije (propagation delay): vreme potrebno da bitovi prodju kroz komunikacioni kanal.

$$P\text{-delay} = \text{duzina kanala} / \text{brzina signala} (\text{brzina signala je najcesce } 2/3 c) = \text{duzina kanala} / 2/3 c = X \text{ s (sekundi)}$$

\*c – brzina svetlosti (nije svuda 2/3, razlicito za WiFi, optiku...)

- Sabiranjem dobijamo ukupno vreme:
  - $L = T + P = M/B + P$

##### **Jedinice mere**

<b>Оzn.</b>	<b>Вред.</b>	<b>Озн.</b>	<b>Вред.</b>
K(ilo)	$10^3$	m(illi)	$10^{-3}$
M(ega)	$10^6$	$\mu$ (micro)	$10^{-6}$
G(iga)	$10^9$	n(ano)	$10^{-9}$

B – bajt

b – bit

## **Primeri kasnjenja**

- Dialup sa telefonskim modemom (slanje ka racunaru u istom gradu)
  - $P = 5\text{ms}$ ,  $B = 56 \text{ kb/s}$ ,  $M = 1250 \text{ B}$
  - $L = 5 \text{ ms} + 1250 \times 8 / 56 \times 10^3 \text{ s} = 5 \text{ ms} + 179 \text{ ms} = 184 \text{ ms}$
- Sirokopojsna veza – kablovska ili DSL (slanje kroz drzavu)
  - $P = 50 \text{ ms}$ ,  $B = 10 \text{ Mb/s}$ ,  $M = 1250 \text{ B}$
  - $L = 50 \text{ ms} + 1250 \times 8 / 10 \times 10^6 \text{ s} = 50 \text{ ms} + 1 \text{ ms} = 51 \text{ ms}$

Dugacka veza ili mali protok proizvode vece kasnjenje. Obicno je jedna od komponenti kasnjenja P ili T dominantna.

## **Bandwidth-Delay Product BDP**

- Poruke zauzimaju prostor na kanalu (dok ne stignu od posiljaoca do primaoca)
- Kolicina podataka prisutnih na kanalu u nekom momentu je BDP
- Ako bismo posmatrali podatak kao materiju, onda bi ovo bila zapremina (kolicina) materije
  - $\text{BDP} = B \times D$
  - D-delay
- Meri se u bitovima
- Mali za kanale u lokalnim mrezama, npr WiFi, veliki za “velike debele” kanale

Primer za BDP:

- Slanje npr. Od Perta do Sidneja
- Opticki kanal
- Dugacak kanal
  - $B = 40 \text{ Mb/s}$ ,  $D = 50 \text{ ms}$
  - $\text{BDP} = 40 \times 10^6 \times 50 \times 10^{-3} \text{ s} = 2000 \text{ Kb} = 250 \text{ KB}$
  - Ovo se smatra velikim BDP-om , velika je kolicina podataka koji se nalaze u kanalu

## **5. Zicani i opticki komunikacioni medijumi.**

Medijum propagira signal sa informacijama u vidu bitova. Tri osnovna tipa medija su:

- Zicani
- Opticki (opticki kablovi)
- Bezicni

### **Zicani – upredena parica (UTP – unshielded twisted pair)**

Upredenu paricu cine dve izolovane bakarne zice, najcesce precnika oko 1mm. Zice su medjusobno spiralno uvijene. Zice se uvijaju zato sto dve paralelne zice predstavljaju odlicnu antenu. Kada se zice upredu ponistavaju se talasi generisani u razlicitim navojima, tako da ceo slop zraci mnogo manje (uvrtanjem se smanjuju smetnje).

Skoro svi telefoni su sa telefonskom centralom povezani pomocu upredene parice. Koristi se I za LAN kablove. Ona moze da se proteze vise km bez pojacivaca, ali su oni neophodni za vece radaljine.

Upredenom paricom se mogu prenositi I analogni I digitalni signali. Brzina prenosa zavisi od debljine zice I rastojanja. Zahvaljujuci niskoj ceni I prihvatljivim perfomansama ona se masovno koristi.

Upredena parica kategorije 3 sastoji se od 2 blago uvijene izolovane zice.

Upredene parice 5. kategorije lice na parice 3. kategorije ali su bile gusce upredene, cime je omogucen kvalitetniji signal na vecim radaljinama, a to znaci brzu komunikaciju.

### **Zicani – koaksijalni kabl**

Bolje je oklopljen od UTP pa podatke prenosi vecom brzinom I na vece daljine. Ima jezgro od cvrste bakarne zice oko koje se nalazi izolator. Oko izolatora je cilindricni provodnik napravljen od gusto upletene bakarne mrezice. Preko njega se nalazi zastitni plasticni omotac. Konstrukcija I elektricna zastita omogucavaju mu dobru kombinaciju velikog propusnog opsega I otpornosti na smetnje. Cest je u upotrebi.

Drugi tipovi zica takodje mogu da prenose podatke, npr. elektricne zice za sprovodjenje struje.

### **Zicani – instalacije za prenos struje**

- Prakticne za upotrebu (vec postoje)
- Jako lose karakteristike prenosa (nisu dizajnirane za to)

## **Optika**

Dugacka, tanka I cista vlakna stakla. Ima ogroman protok zbog opsega frekvencije. Podnose velike udaljenosti zbog malog slabljenja.

Opticki sistem za prenos podataka sadrzi 3 glavne komponente: svetlosni izvor, prenosni medijum I detektor. Po konvenciji svetlosni impuls označava bit 1, a odsustvo impulsa bit 0. Prenosni medijum je ultratanko stakleno vlakno. Detektor proizvodi elektricni impuls kada na njega padne svetlosni zrak. Spajajući svetlosni izvor s jednim krajem optickog vlakna, a detektor s njegovim drugim krajem dobijamo jednosmerni sistem prenosa podataka koji prihvata elektricni signal, pretvara ga u svetlosni impuls I prenosi, a zatim ga na drugom kraju ponovo pretvara u elektricni signal.

Kada svetlosni zrak prelazi iz jedne u drugu materijalnu sredinu, npr iz kvarcnog stakla u vazduh, on se prelama na granici kvarc/vazduh. Svetlosni zrak na granici medijuma dolazi pod upadnim uglom  $\alpha$  i napusta je pod uglom  $\beta$ . Upadni ugao zavisi od prirode dva medijuma. Kada upadni ugao predje odredjenu kriticnu vrednost, zrak uopste ne prelazi u vazduh, vec se vraca u kvarc. Na taj nacin zrak sa upadnim uglom vecim ili jednakim kriticnom zauvek je zarobljen u vlaknu I moze da prolazi kroz njega kilometrima.

Kroz vlakno moze istovremeno da prolazi vise svetlosnih zraka od kojih se svaki odbija pod drugacijim uglom, uvek vecim od kriticne vrednosti – takvo vlakno se naziva **multimodalno vlakno**.

Ali ako se precnik vlakna svede na nekoliko talasnih duzina svetlosti, vlakno je toliko tanko da se svetlost kroz njega prostire samo pravolinijski, bez odbijanja – **monomodalno vlakno**. Ova vlakna su skuplja ali se siroko koriste za veca rastojanja (do 100km).

Opticki kablovi se sastoje od staklenog jezgra, od stakleneobloge I plasticnog omotaca. Vlakna se najcesce grupisu u snopove I zasticuju dodatnim spoljnim omotacem.

## 6. Bezicni komunikacioni medijumi.

- Posiljalac emituje signal kroz prostor
  - emituje signal u mnogim (svim) pravcima, za razliku od zice
  - potencijalno veliki broj primalaca
  - bliski signali (slicne frekvencije) se mesaju kod primaoca, potrebno je koordinirati upotrebu

### Elektromagnetni spekter

Kada se elektricno kolo spoji sa antenom odredjene velicine, elektromagnetni talasi se mogu slati kroz prostor, gde ih na nekoj udaljenosti moze primiti odgovarajuci prijemnik. Sve bezicne komunikacije se zasnivaju na ovom principu.

Svi delovi elektromagnetskog spektra mogu se koristiti za prenos informacija ako im se modulira amplituda, frekvencija ili faza. Kolicina infomracija koju moze da prenese elektromagnetni talas zavisi od njegove frekvencije.

Za mreze je najinteresantniji opseg mikrotalasa (3G, 4G, WiFi), ali se koriste i ostali delovi spektra. Kako bi se izbegla mesanja signala, opsezi (bandovi) se pazljivo dodeljuju. Takodje se prodaju na aukcijama za najvise ponude.

#### Interesantni delovi spektra (bandovi):

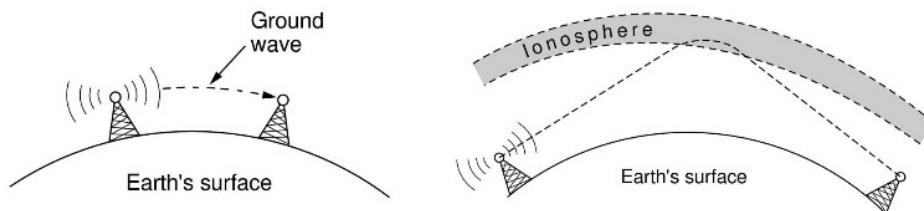
Za bezicnu mrezu se opisirno koriste : mikrotalasi, 3G I nelicencirane frekvencije (ISM)  
– obicno zbog fragmentacije drugih bandova npr WiFi

### Radiotalasi

Radio signali mogu da prelaze velika rastojanja, i lako prolaze kroz zgrade, zbog cega se naveliko koriste u komunikacijama. Pri nizim frekvencijama oni lako prolaze kroz prepreke, ali im snaga naglo opada sa rastojanjem od izvora. Pri visim frekvencijama radio talasi teze da se prostiru pravolinijski i da se odbijaju od prepreka. Signal im slabi iz raznih razloga: biva apsorbovan, zbog odbijanja, ometa ga zracenje elektricnih uredjaja itd.

U slucaju VLF, LF i MF radio talasi prate krivinu Zemlje. Glavni problem talasa iz ovog područja jeste njihov mali propusni opseg.

U područjima HF i VHF frekvencija talase koji se prostiru povrsinom apsorbuje tlo. Medjutim talasi koji dosegnu jonasferu odbijaju se od nje i vracaju na Zemlju.



У опсезима VLF (very low freq.), LF (low), и MF (medium) радио таласи прате закривљеност Земље

У HF (high) опсезу, радио таласи се одбијају од јоносфере

### Mikrotalasi

Prostiru se pravolinijski, i zato se mogu fokusirati. Lose prolaze kroz zidove. Mogu se i odbijati od niskih atmosferskih slojeva i na cilj doci kasnije.

Imaju veliki frekventni opseg I koriste se cesto za zatvorene namene poput WiFi, kao I za otvorene poput 3G I sateliti. Signal slabi I reflektuje se od objekata iz okruzenja. Jacina varira zbog udaljenosti, sabiranja signala, I sl.

### Svetlost

Svetlosni signali se mogu koristiti kao komunikacioni medijum. Svetlost je vrlo usmeren talas I ima veliki frekventni opseg. Moze se iskoristiti upotrebom lasera I fotodetektora. Takav laser ima veliku propusnu moc I izuzetno je jeftin. Veoma uzak snop lasera je ovde I njegova mana. Zbog toga se u sistem obicno ugradjuju rasipna sociva koja prosiruju snop. Nedostatak snopa je njegova nemogucnost da prodre kroz kisu ili maglu.

Bezicne ili zicane/opticke komunikacije?

Bezicne:

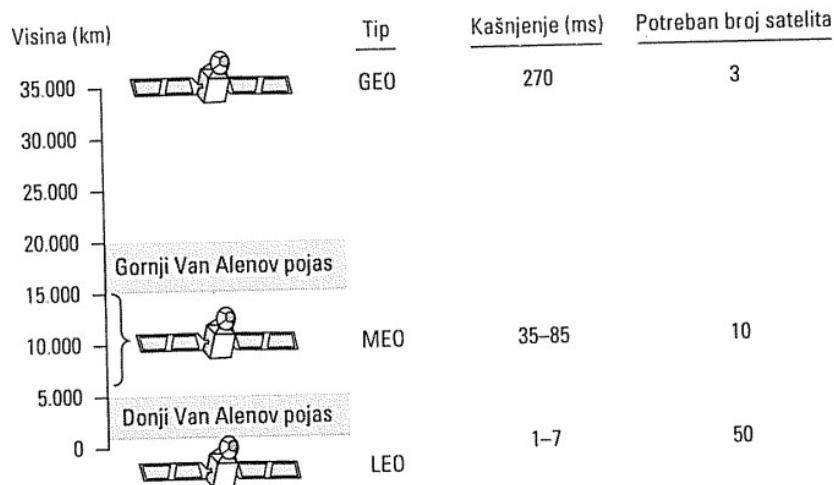
- jednostavne za postavljanje I jeftine
- prirodno podrzavaju mobilnost
- prirodno podrzavaju emitovanje
- mesanje signala se mora razresavati
- jacina signala, pa samim tim I proto izuzetno varira

Zicane/optika:

- lako se projektuje fiksni protok duz odabralih ruta
- skup za postavljanje, posebno na vecim udaljenostima
- nije projektovan za mobilnosti ili emitovanje

## 7. Komunikacioni sateliti.

Vestacki satelit moze da pojaca primljeni signal pre nego sto ga ponovo posalje, na taj nacin je postao moca komunikacioni sistem. Sateliti su efikasni za emitovanja I komunikaciju "bilo kada/bilo gde". Ukoliko bi se satelit kretao kroz Van Alenove pojaseve slojeva nanelektrisanih cestica koje na okupu drzi zemljino magnetno polje, bio bi razoren udarima ovih cestica. Zbog toga se sateliti smestaju u tri bezbednija visinska područja.



Komunikacioni sateliti I neka njihova svojsta – visina iznad zemljine povrsine, kasnjenje povratnog signala I broj satelita potreban za pokrivanje cele planete.

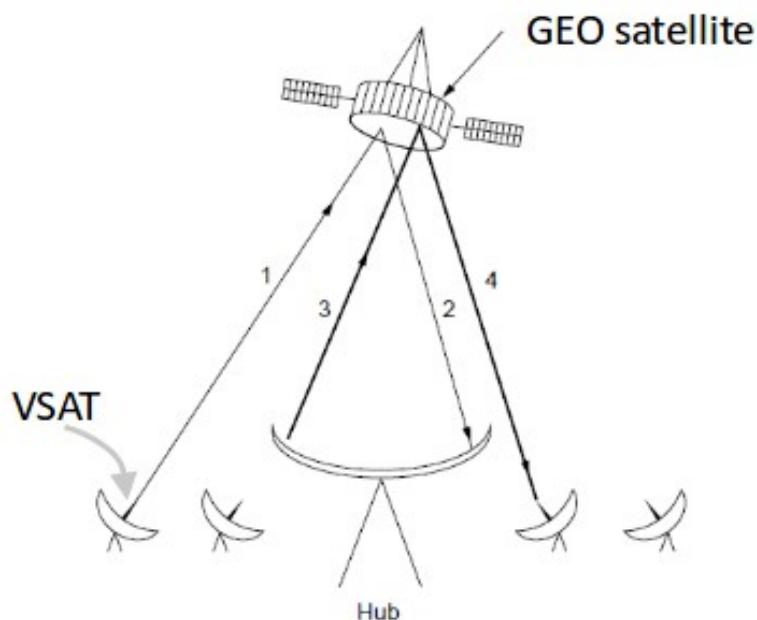
- Tipovi satelita
  - Geostacionarni (GEO)
  - Srednje-orbitni (MEO)
  - Nisko-orbitni (LEO)

### Geostacionarni sateliti GEO

Orbitiraju 35 000 km iznad fiksne lokacije – na veoma velikoj visini.

Jeftine mikrostanice – terminali s vrlo uskim emisionim snopom – (Very Small Aperture Terminals, VSAT). Tehnologija se koristi za jednosmeran prenos u satelitskoj TV difuziji. U VSAT sistemima, mikrostanice nemaju dovoljnu snagu da direktno komuniciraju jedna s drugom (preko satelita) vec se saobracaj odvija preko tzv.razvodnika (haba) – zemaljske antene visokog ucinka. VSAT dobija I salje signal ka centralnom uredjaju (habu), a postoje I sistemi bez centralizovanog uredjaja. Hab npr. odasilja televizijski program na GEO, a ovaj emituje na delu zahvacene Zemljine teritorije, te ka svim pripadajucim VSAT uredjajima.

Iako signali ka satelitu I od njega putuju brzinom svetlosti, velika udaljenost GEO satelita dovodi do znacajnog kasnjenja. Sateliti su po prirodi medijumi za neusmermeno emitovanje – slanje signala hiljadama stanica ili jednoj stanici kosta isto. Sa aspekta bezbednosti I privatnosti nisu dobri: svako moze sve da cuje. Satelitski prenos je izuzetno imun na greske.



### Srednje-orbitni MEO

Kruze na mnogo manjoj visini od GEO satelita (18 000km), pa na Zemlji ostavljaju manji otisak, a zemaljska stanica moze da bude manje snage. Ne koriste se za telekomunikacije pa ih ne razmatramo.

### Nisko-orbitni LEO

Kruze na jos manjim visinama. Zbog toga sto se brzo krecu sistem koji tezi da pokrije citavu povrsinu Zemlje mora da ima mnogo satelita. Nisu geostacionarni, ali ako ih ima vise, onda mogu da garantuju stalnu pokrivenost na odabranim regijama. Posto su sateliti vrlo blizu povrsine Zemlje, nisu potrebne snazne zemaljske stanice, a kasnjenje signala iznosi nekoliko ms (manje nego kod GEO).

Satelit ili optika?

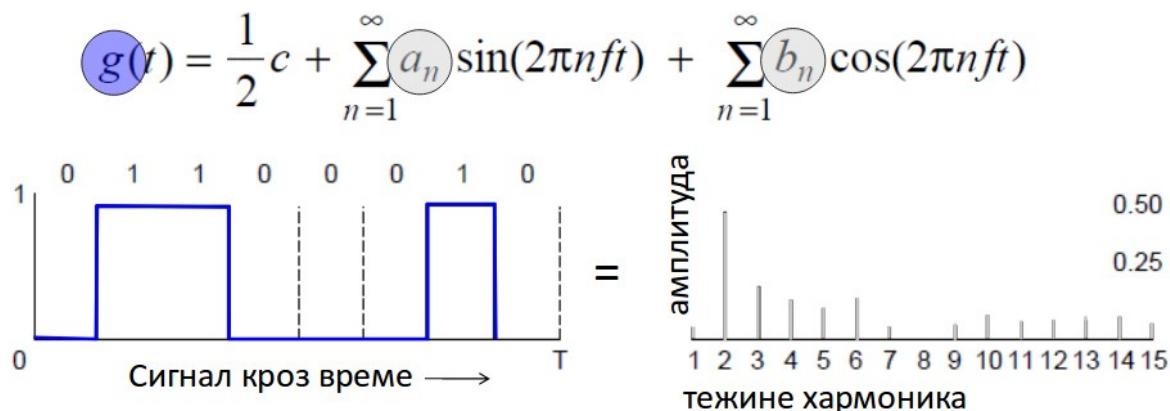
- Satelit:
  - Nakon lansiranja komunikacija se moze brzo uspostaviti bilo gde I bilo kada
  - Emitovanje na velika područja
  - Ograniceni protok I mesanje signala
- Optika:
  - Ogroman protok duz velikih udaljenosti
  - Instalacija skupa I komplikovana

## 8. Signali, prenos, frekvenciona reprezentacija, signal u zicanim, optickim, bezicnim medijumima.

Prenos signala

- Nasi podaci su digitalni (bitovi), ali se kroz mrezu salje analogni signal → moramo da predstavimo bitove kao signale
- Analogni signali kodiraju digitalne bitove
- Sta se desava sa signalom prilikom propagacije?
- Signal se kroz vreme moze predstaviti putem svojim frekvencijskih delova (Furijeova analiza)

Furije je dokazao da se svaka normalna periodica funkcija  $g(t)$  periode  $T$  moze predstaviti kao zbir broja sinusnih I kosinusnih funkcija:



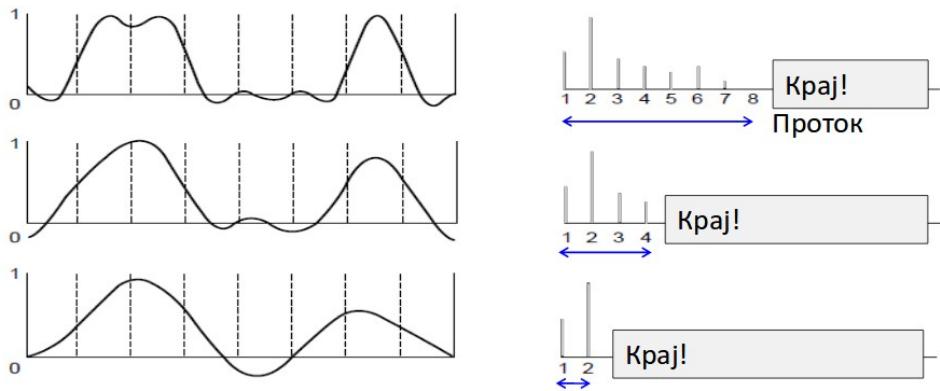
an I  $b_n$  su amplitude n-tog harmonika (clana) sinusne I kosinusne funkcije,  $c$  je konstanta, a  $f = 1/T$  je osnovna frekvencija.

Tako razlozena periodica funkcija je Furijeov niz. Ako je poznata perioda  $T$  I ako su zadate amplitude, prvobitna funkcija se dobija sabiranjem niza iz jednacine.

Signal podataka koji ima ograniceno trajanje moze se rastaviti u Furijeov niz ako zamislimo da se njegov profil stalno ponavlja, tj. Da se profil iz intervala od 0 do  $T$  istovetno ponavlja u intervalu od  $T$  do  $2T$ , itd.

Amplitude za svaku funkciju  $g(t)$  I konstanta  $c$  se mogu izracunati mnozenjem obe strane jednacine odredjenim vrednostima I integracijom izmedju 0 I  $T$ .

Sta se zapravo ovde desava? Kada saljemo neki podatak, mi imamo njegovu reprezentaciju bitovima. Leva strana gornje like prestavlja napon signala koji salje racunar. Zatim preko Furijeove analize izracunamo koeficijente  $a_n$ ,  $b_n$  I  $c$ . Sredjekvadratne amplitude  $\sqrt{a_n^2 + b_n^2}$  (oba su pod korenom, desna strana slike) su vazne jer su njihovi kvadrati proporcionalni energiji koja se pri odredjenoj frekvenciji prenese. U svim prenosnim medijumima razlicite komponente Furijeovog niza razlicito slabe, sto dovodi do izoblicenja signala. Amplitude signala obicno se prenose bez slabljenja pocev od frekvencije 0 pa do neke frekvencije  $f_c$ , a iznad te granicne frekvencije sve amplitude se smanjuju. Opseg frekvencija koje se prenose bez veceg slablenja naziva se **propusni opseg** (bandwidth). Granicna frekvencija u praksi nije tako ostra, pa se propusni opseg cesto definise kao opseg frekvencija od 0 do frekvencije pri kojoj snaga signala opadne na polovinu. Propusni opseg je fizicko svojstvo transportnog medijuma I obicno zavisi od konstrukcije, debljine I duzine medijuma. Ogranicavanje propusnog opsega ogranicava I brzinu prenosa, cak I kroz savrsene kanale.



Suzavanjem opsega gubimo ostre ivice. Kako se smanjuje opseg, tako se I signal menja, tako da je sve teze I teze prepoznati da li je u pitanju bila jedinica ili nula. Koji od ovih signala bi bio najpogodniji za prenosenje podataka? Sredisnji, zato sto moze da se prepozna gde je rec o nuli, a gde o jedinicu, nije nam potreban siri opseg, mozemo I ovako na drugom kraju da iskodiramo podatke nazad u digitalne.

### Signal preko zice

- Sta se desava sa signalom dok prolazi kroz zicu?
  - 1. signal kasni (brzina je  $\sim 2/3c$ , a ne beskonacna) - signalu treba neko vreme da se propagira, I dok se to desava prolazi odredjeno vreme
  - 2. signal slabi (sa porastom udaljenosti) – slabljenje je smanjenje energije singala na putu zbog gubitaka. Gubici se izrazavaju brojem decibela po km I zavise od frekvencije.
  - 3. frekvencije iznad neke granice brze slabe – videli smo na slici kako signal, umesto da ima ostre ivice I da bude pravougaon, pocne sa smanjenjem opsega da uzima kruzniji oblik, kao zakrivljene linije, to se desava zbog gutbitka nekih frekvencija
  - 4. desava se sum (zbog spoljnih efekata) – sum je energija koja ne potice od izvora.

Elektroinzenjeri:

Protok = sirina frekвencionog opsega (Hz)

Racunari:

Protok = kapacitet prenosa informacija (b/s)

Sinal preko optike:

- svetlo se prenosi sa veoma malim gubitkom u tri siroka frekventna opsega
  - Krajnji desni zalazi u opseg infracrvenih talasa
  - slabljenje dato u decibelima po kilometru je jako malo, mogu se slati signali na velike daljine a da slabljenje bude jako malo

Singal u bezicnim komunikacijama

- zbog visokih frekvencija bezicnih prenosa, nije moguce digitalni signal direktno kodirati u analogni vec se kroisti koncept singala nosaca.
- Putuje brzinom svetlosti, ali jako brzo slabi (sa kvadratom rastojanja)
- Visestruki signali na istoj frekvenciji se mesaju kod primaoca - ako bismo imali primaoca na nekoj drugoj lokaciji on bi video jedan jak signal I ostale slabe signale, bio bi problem razdvojiti ovakve signale I posmatrati ih zasebno jer se oni medjusobno mesaju
- Ako su lokacije dovoljno udaljenje, moguce je koristiti istu frekvenciju – ako su lokacije dovoljno razdvojene onda ce signali koji se mesaju sa odredjenim signalom biti slabi I nece smetati u tumacenju naseg jaceg sinala.
- Jos neki otezavajuci efekti:
  - propagacija bezicnog signala je slozena I zavisi od okruzenja (jer se ne ide kroz kanal vec signal propagira kroz prepreke)

- Karakteristike zavise I od frekvencije (razlicite frekvencije vezane su za razlicite fizicke efekte)
- postoji problem sa sabiranjem odbijenih signala kod mikrotalasa

Sabiranje odbijenih talasa:

- Signali mogu da se odbijaju od objekata I putuju kroz vise nezavisnih putanja
  - posle kada stignu visestruki signali kod primaoca, oni se mogu lose sabrati



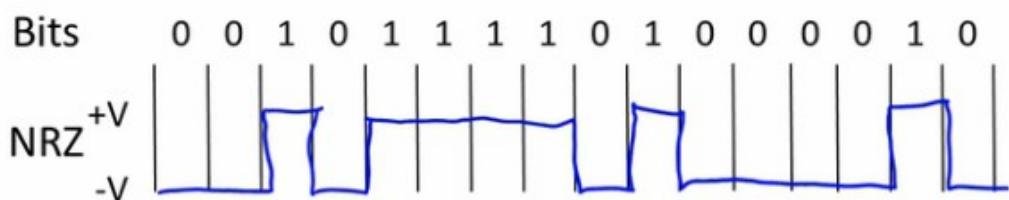
Signal od transmitera ide u vise pravaca, neki mogu ici direkto do primaoca, neki se mogu odbijati I takodje doci do primaoca. Primalac ce videti dva razlicita signala koja su dosla iz vise putanja. Tako bi na primer jedan primalac (na slici telefon) dobio dva slicna signala sto bi proizvelo efekat jednog jakog signala, dok bi neki drugi primalac (racunar) dobio npr jedan signal, I onda jos jedan koji je isti samo sifovan . Kada spojimo sve ovo rezultat je da se dobije izbledeli (faded) signal. Ovo se zove multipathing, I nezgodno je jer se moze desavati na malim daljinama.

## 9. Modulacija I multipleksiranje signala.

Nacin predstavljanja digitalnih infomacija u okviru fizickog medijuma.  
Kako se predstavljaju bitovi signalima? (to je modulacija)

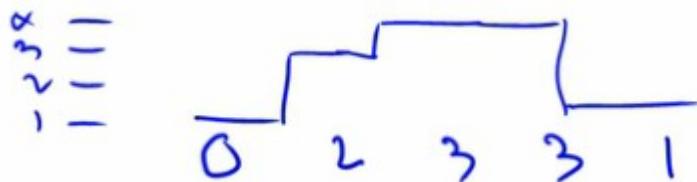
### Jednostavna modulacija:

Visoki napon (+V) predstavlja 1, niski (-V) predstavlja 0  
Ovo se zove NRZ kodna sema (Non-Return to Zero)



### Mnoge druge seme kodiranja:

Moze se koristiti i vise od dva nivoa odnosno simbola npr 4 nivoa = 2 bita  
Ovo zavisi od tehnoloskih karakteristika medijuma i mogucnosti dekodiranja



### Clock recovery – dilema oko broja nula

Sta ako imamo dug niz nula (ili jedinica)? Moze se desiti da primalac nije siguran koliko ih je.  
Kako se ovo moze resiti?

Primalac ne vidi bitove, on vidi signal. Dakle vidi jasno na primer jednu jedinicu, ali onda ako nakon nje sledi dug niz nula on ne moze jasno da primeti koliko je nula u pitanju.

Neka od resenja:

Npr. scrambling – koristi se random signal,

Mancestersko kodiranje – ugradjuje se promena (transition) u svaki signal (i u 0 i u 1), itd.

### (De) kodiranje signala – sema 4B/5B

Svaka 4 bita na 5 bitova bez dugackih nizova nula:

0000 → 11110 , 0001 → 01001, 1110 → 11100, ....1111 → 11101

Iznad vidimo kako se neki bitovi kodiraju.

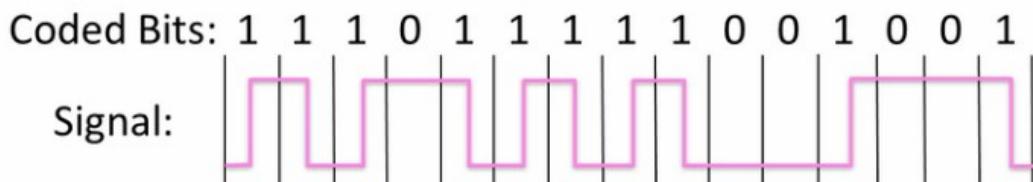
Ako se koristi ovo mapiranje najvise moze biti 3 uzastopne nule u nizu.

Ali sta radimo ukoliko se desavaju dugacki nizovi jedinica?

Inverzija signala na jedinici kako bi se izbegli dugacki nizovi jedinica. Invertuje se nivo signala na 1 i drzi se na toj istoj voltazi signala za 0.

–  $0000 \rightarrow 11110$ ,  $0001 \rightarrow 01001$ ,  $1110 \rightarrow 11100$ , ...  $1111 \rightarrow 11101$

- Message bits: 1 1 1 1 0 0 0 0 0 1



Computer Networks

9

Na slici vidimo kako se signal inverte uvek kada je u pitanju 1, a ako naidje na 0, onda on ostaje isti kakav je bio u tom trenutku.

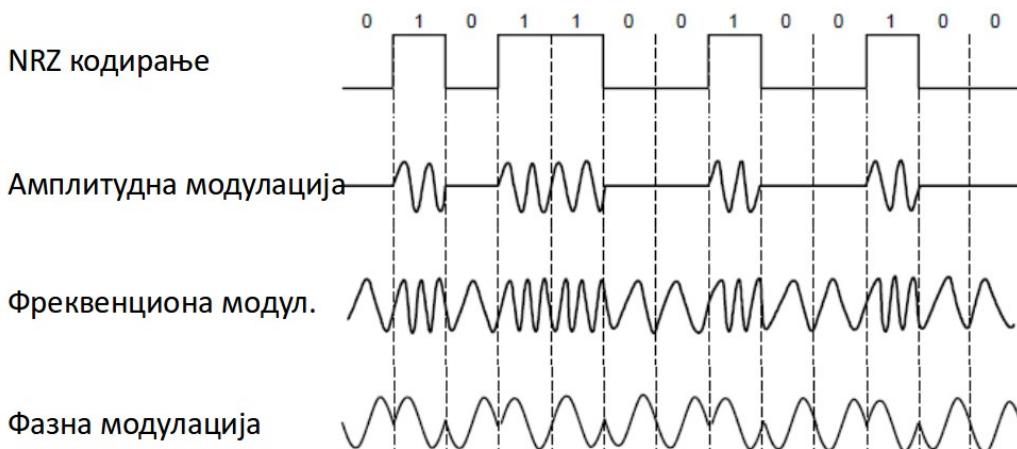
### Modulacija preko nosaca (passband)

Prethodni tipovi modulacije su bili direktni (baseband)

- signal se na zicu salje direktno
- frekvencije digitalnog I analognog signala su iste

Ovo nije moguce kod optike I bezicnih signala, zbog toga sto rade na mnogo visim frekvencijama. Zbog toga se koristi signal nosac. Modulacija preko nosaca koristi drugaciju (indirektnu) reprezentaciju signala.

Signal nosac je signal koji oscilira na zeljenoj frekvenciji. On se moze modulirati promenom amplitudne (visina signala), frekvencije (na primer da li oscilira brzo ili polako, za 1 brzo oscilira, za 0 polako) ili faze (signal ide od gore ka dole za 1, I od dole ka gore za 0).



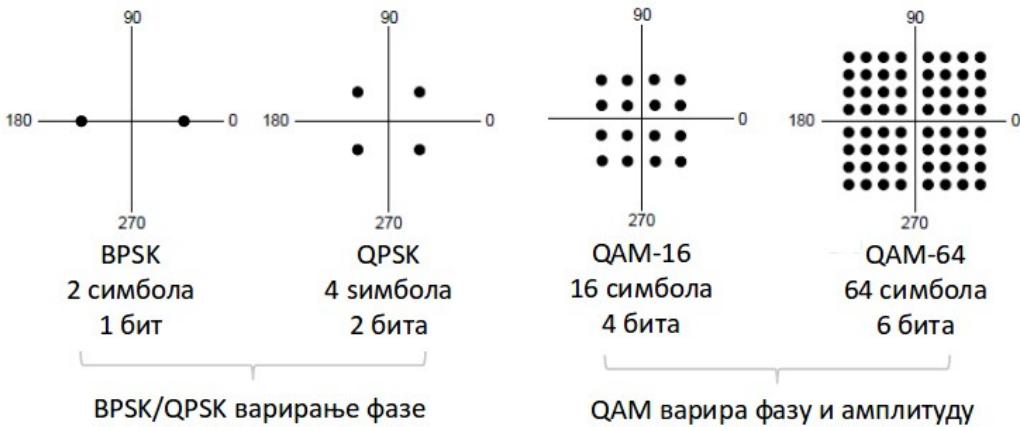
Dakle:

Amplitudna modulacija – koriscene su dve razlicite amplitude za predstavljanje nule I jedinice.

Frekvencionala modulacija – koriste se dva ili vise razlicitih tonova.

Fazna modulacija – nosecom talasu se sistematski obrce faza za 180 stepeni u jednakim vremenskim intervalima.

Tehnika modulisanja odredjuje broj bitova po simbolu. Bodovima se izrazava broj napravljenih uzoraka u sekundi. Svaki uzorak nosi deo informacije tj. Jeden simbol. Broj simbola I broj bodova u sekundi su jednaki. Svi noviji modemi koriste kombinaciju tehnika modulisanja da bi preneli vise bitova po bodu. Cesto se kombinuju visestruka amplitudna I fazna modulacija.



Npr QAM-16 – Kvadraturna amplitudna modulacija ima 4 amplitude I 4 faze, ukupno 16 kombinacija. QAM-64 – ima 64 kombinacije, po simbolu se prenosi 6 bitova.

### Multipleksiranje

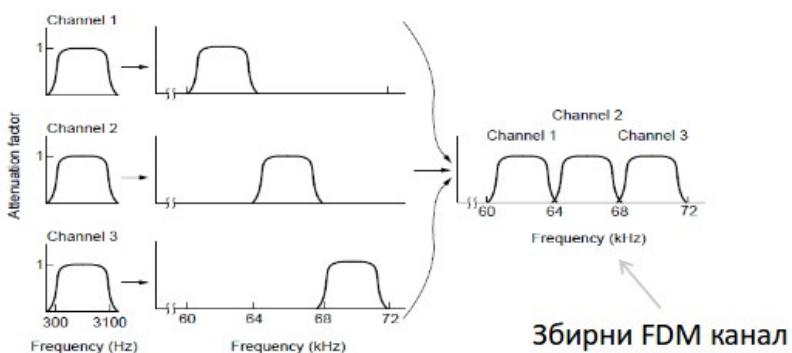
Bavi se deljenjem kanala izmedju vise korisnika. Analogon: Soba puna ljudi, kako neki od njih da komuniciraju?

Tri standardna pristupa:

- Frekvencionalno multipleksiranje
- Vremensko multipleksiranje
- Multipleksiranje zasnovano na kodovima (CDMA)

#### Frekvencionalno multipleksiranje:

Deli kanal tako sto razlicite korisnike postavlja na razlicite frekvencione opsege. U nasem analognom primeru sa sobom punom ljudi, ovo bi znacilo da se fokusiramo na slusanje onih koji pricaju jako brzo, jako sporo, srednje...



Kanal 1 – prvo bitni kanali, U sredini su kanali premesteni u dodeljena frekventna područja, I desno je multipleksiran kanal

#### Vremensko multipleksiranje:

Korisici se smenjuju u kratkim vremenskim intervalima u kojima imaju na raspolaganju ceo propusni opseg. Korisnici se drze fiksnog rasporeda. Upotrebljava se u sistemima fiksne I mobilne telefonije. U nasem primeru, ovo bi znacilo da u sobi punoj ljudi svi ostali cete dok prica neka podgrupa, pa onda prica sledeca grupa, itd.

#### Kodno multipleksiranje (CDMA)

CDMA – Code Division Multiple Access

Korisnicima se dodeljuju kljucevi, koji su medju sobom ortogonalni. Medjusobno ortogonalni kljucevi nemaju medjusobnu interakciju. Na originalni signal se primenjuje kljuc (skalarni proizvod).

U primeru sobe ljudi ovo bi znacilo da neki ljudi pricaju razlicitim jezicima.

## 10. Prirodna ogranicenja prenosa signala.

Koliko cesto se moze slati podatak kroz kanal?

Najkvistov limit (~1924)

Senonov kapacitet (1948)

Realni sistemi su dobro realizovani ako nisu mnogo daleko od ovih ogranicenja. Ogranicenja nam govore koliko smo relativno dobri u necemu.

Klucni koncepti za prenos signala:

Protok (B), jacija signala (S) – meri se kod primaoca, jacija suma (N)

B ogranicava brzinu promena – frekvenciju. Karakteristika kanala

S I N ogranicava broj razlucivih nivoa signala. Karakteristika primaoca

Da vidimo rezultate:

### Najkvistov limit:

Izveo je jednacinu za maksimalnu brzinu prenosa kroz besumni kanal ogranicene propusne moci.

Maksimalan broj promena simbola je  $2B$  (simbol je waveform koji se koristi za donosenje informacija, dakle to je signal sto predstavlja bitove)

Ako postoji  $V$  nivoa signala (nivo singala moze biti broj bitova) onda je maksimalan protok u bitovima:

$$R = 2B \log_2 V \text{ b/s}$$

npr. Besumni kanal opsega 3kHz moze da prenosi binarne signale (2 naponska nivoa) brzinom od maksimalno 6000 b/s.

### Senonov kapacitet:

Senon je prosirio Nikvistovu jednacinu na kanale sa slucajnim sumom.

Situacija se pogorsava ako postoji slucajan sum, a njega uvek ima. Sum se izrzava kao kolicnik snage signala I snage suma I naziva se SNR – signal – sum – odnos (Signal to Noise Ratio).

Broj razlucivih nivoa signala zavisi od odnosa jocene signala I jocene suma S/N. SNR se meri u decibelima:

$$\text{SNR dB} = 10 * \log_{10} (\text{S/N}) \text{ (osnova je 10)}$$

npr. Za vrednost odnosa S/N od 10 odgovara 10 dB.

Koristi se logaritamska skala jer S/N (sum moze mnogo da varira) moze samo da varira.

Formula za Senonov kapacitet:

$$C = B \log_2 (1 + \text{S/N}) \text{ b/s}$$

Broj razlucivih signala se dobija iz odnosa:  $(S + N)/N = 1 + S/N$

Zice I optika:

Mogu se projektovati ciljni SNR I B.

B se moze menjati kvalitetom zice, a SNR se moze menjati ogranicavanjem duzine kabla. Time se popravlja prenos.

Samim tim I ciljni prenos u b/s

Bezicni kanali:

Za dato B, SNR drastично varira, cak i do 60 dB. Nije isplativo projektovati za najgori slucaj, mora se "ziveti" s visokim varijacijama prenosa. Mora se adaptirati prenos za date okolnosti.

## **11. Pregled relevantnijih sistema komunikacija.**

### Sistem fiksne telefonije:

- Hijerarhijski sistem za prenos govora
  - Lokalne konekcije (unutar mesta) najčešće upredene parice (postojeca infrastruktura)
  - Medjumesne konekcije, optički kablovi (novijeg datuma)
  - Centrale, preusmeravanje I održavanje konekcija
- Putem ovog sistema se realizuje I DSL (odnosno ADSL)

### Sistem mobilne telefonije – generacije:

- 1G, analogni glas
  - FM modulacija (kao kod radija)
  - Odvojene frekvencije za slanje I primanje govora - IMTS. Postojaо je takođe I mali broj kanala tako da su korisnici morali dugo da čekaju pre nego što dobiju vezu. Ovo je bilo nepraktično.
  - Zatim je usledio AMPS (dole objasnjen)
- 2G, digitalni glas
  - GSM (Global System for Mobile Communications)
  - QPSK modulacija
- 3G, digitalni glas I podaci
  - UMTS (Universal Mobile Telecommunications System)- omogućeno je da korisnik istupi iz W-CDMA celije I udje u GSM celiju a da se veza pri tome ne prekine.
  - Koristi CDMA – kodirani visestruki pristup – za razliku od prethodnika (kao što je GSM) umesto da dodeljeno frekventno područje deli na vise stotina uskih kanala, CDMA svakoj stanici dozvoljava da sve vreme emituje u celom frekventnom području, a njihove jednovremene emisije razdvaja drugacijim kodiranjem. Ključno je izdvajati samo zeljeni signal I istovremeno sve drugo odbaciti kao pozadinski sum.
- 4G, digitalni glas I podaci
  - LTE (Long Term Evolution)
  - OFDM modulacija – napredna varijanta FDM

### Organizacija baznih stanica: (**AMPS**)

- U svim sistemima mobilne telefonije geografsko područje se deli na celije odnosno bazne stanice
  - Svaki mobilni korisnik koristi celijsku frekvenciju. U svakoj celiji se koristi skup frekvencija razlicit od frekvencija koje koriste susedi.
  - Svaki mobilni telefon se u svakom trenutku logicki nalazi u određenoj celiji, pod kontrolom bazne stanice u toj celiji. Kada mobilni telefon fizicki napusti celiju, njena bazna stanica zapaza da signal slablji I svim okolnim celijama salje upit o snazi signala koji primaju s tog telefona. Baza tada predaje kontrolu celiji koja najjace prima signal tј onoj u kojoj se telefon nalazi. Telefonu se salje obavestenje o "novom sefu", I ako je razgovor u toku, trazi se da predje na drugi kanal posto se stari ne koristi u susednim celijama. Ovaj proces zove se proces predavanja upravljanja (handoff)
  - Iste frekvencije se koriste sa nesusednim celijama
  - Za podrzavanje veceg broja korisnika, obicno se ogranicava geografski prostor celije
  - u područjima gde zbog sve veceg broja korisnika sistem postaje opterecen celije se dele na manje mikrocelije I snaga sistema se smanjuje u odgovarajucoj meri, cim se omogucava cesce koriscenje istih frekvencija.

## Internet preko kablovske

- Internet kabl moze da koristi postojeću infrastrukturu za kablovsku televiziju
  - Drugacija je topologija u odnosu na telefonski sistem
  - Ovde organizacija podseca na toplogiju magistrale (deljenog kanala)

## Podela spektra frekvencija:

Preuzimanje I postavljanje podataka (download I upload) koriste frekvencione opsege koji se ne koriste za gledanje TV programa.

## Kabloska ili (A)DSL

- Kabloska:
  - koristi koaksijalne kableve ka korisnicima (dobar protok)
  - podaci se salju svima, jer je magistrala (manja sigurnost)
  - protok je deljen medju korisnicima, pa moze varirati
- ADSL (asimetrična varijanta DSL-a):
  - protok je posvecen svakom korisniku
  - nema mogucnost emitovanja kao kabloska
  - koristi upredene parice na korisnicima (nizi protok)

12. Слој везе, улога, комуникација са слојем испод и изнад, кратко објашњење списка активности на слоју везе

#### Одговорност слоја везе

- Пренос оквира путем једног или више повезаних комуникационих канала
- Оквири су фиксиране величине
- Наслања се на физички слој

Под суседним рачунарима подразумевамо рачунаре међусобно повезане комуникационим каналом који се теоријски понаша као ћица тј испоруђује битове оним редом којим су послати. Испрва делује као да је ово једноставно и да је само потребно да рачунар A смешта податке на ћицу а да ih рачунар B само приhvата. Најзлост у комуникационим колима nastaju грешке и како она преносе податке ограниченој брзином, постоји увек извесна времанска разлика између trenутка slanja i trenутка prijema podataka. Ova ограничења последићно утичу на ефикасност преноса података и протоколи који се користе за комуникације moraju ih uzeti u obzir.

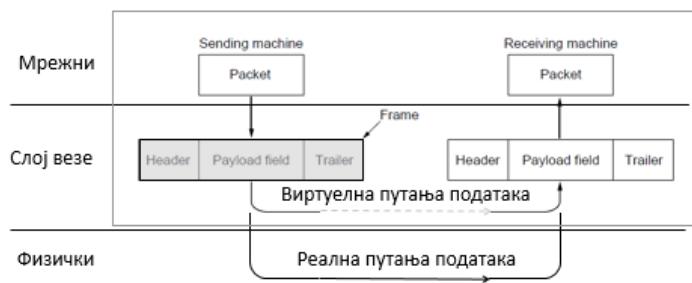
Пројектовање слоја везе података:

Слој везе података има више специфичних функција, које се остварују кроз:

1. Dobro definisan uslužni interfejs ka mrežnom sloju
2. Obradu грешака при преносу
3. Управљање током података тако да рачунар који прима податке не буде нима преплављен

Da bi остварио наведене циљеве, слој везе података преузима пакете које добија од мрежног слоја и капсулира ih у оквире погодне за транспорт. Svaki оквир садржи заглавље, полje за корисниčke податке u kome je paket i završni blok okvira.

#### Расподела активности по слојевима



Usluge које се обезбеђују за мрежни слој:

Osnovna услуга је пренос података из мрежног слоја на изворишном рачунару у мрежни слој одредишног рачунара. На изворишном рачунару постоји део мрежног слоја који испоруђује битове слоју везе података да bi ih ovaj prosledio do odredišta. (Slika) Конкретне понуђене услуге слоја везе података варирају od система до система ali se обично уграджу barem tri sledeće услуге:

1. Prenos podataka bez uspostavljanja direktne veze i bez potvrde o njihovom prijemu
2. Prenos podataka bez uspostavljanja direktne veze sa potvrdom o njihovom prijemu
3. Prenos podataka sa uspostavljanjem direktne veze i sa potvrdom o njihovom prijemu

U prvom slučaju izvođač računar šalje podatke odredišnom računaru ne zahtevajući od njega potvrdu prijema okvira, takođe pre slanja se ne uspostavlja logička veza, niti se nakon prenosa takva veza raskida. Usluga je pogodna kada je učestalost grešaka niska tako da se ispravljanje grešaka prepusta višim slojevima, pogodna za saobraćaj koji se događa u realnom vremenu (npr. govorni). Takođe se ova usluga koristi i u većini lokalnih mreža.

Druga usluga je pouzdanija iako se i dalje ne uspostavlja logička veza između pošiljaoca i primaoca jer se zahteva da pošiljalac dobije potvrdu o ispravnom prijemu okvira. U suprotnom moguće je ponovno slanje. Pogodno za slanje preko bučnih kanala npr. bežični sistemi.

Mrežni sloj uvek može da pošalje paket i da čeka potvrdu o njegovom dolasku, a ukoliko ona ne dođe u predviđenom roku, pošiljalac može da ponovo pošalje čitavu poruku. Problem ovakve strategije je što okviri imaju ograničenu veličinu, za razliku od paketa. Ako se paket izdeli na okvire i potom se za svaki okvir očekuje pojedinačna potvrda prijema, podaci će se prebaciti mnogo brže nego ako se čeka potvrda prijema celog paketa. Kod ove opcije podaci se prenose u tri jasno odeljene faze: U prvoj se inicijalizuju promenljive i brojači koji vode računa o primljenim okvirima. U drugoj fazi se prenosi jedan ili više okvira. U trećoj se veza raskida, oslobađaju se promenljive, bafer i drugi resursi upotrebljeni za održavanje veze.

Treća opcija je najsloženija jer se veza mora uspostaviti pre razmene podataka. Svaki okvir se numeriše, a sloj veze podataka farantuje da je svaki takav okvir i primljen. Štaviše garantuje da je svaki takav okvir primljen samo jednom i da su svi okviri stigli redom kojim su i poslati.

### 13. Уоквиривање у слоју везе

Sloj veze najčešće deli tok podataka u okvire konačne veličine i izračunava kontrolni zbir za svaki okvir. Kada okvir stigne na odredište, kontrolni zbir se računa ponovo, ako se dobijeni zbir razlikuje od zbiru sadržanog u okviru onda se preduzimaju odgovarajuće mere. Deljenje tokova u okvire nije tako jednostavno kao što se čini. Jedan od načina jeste da se između dva okvira ubaci prazan vremenski interval, međutim mreže retko garantuju sinhronizovanost događaja, pa pomenuti intervali između susednih okvira tokom transporta mogu nestati ili se pojaviti tamo gde im nije mesto. Pošto je ovo previše rizično, razvijene su drugačije metode, među kojima su:

1. Prebrojavanje znakova
2. Upotreba indikatorskih bajtova uz umetanje bajtova
3. Upotreba početnih i završnih indikatora uz umetanje bitova
4. Narušavanje kodiranja fizičkog sloja

1. U prvoj metodi u zaglavlju postoji polje sa brojem znakova u okviru. Kada sloj veze pročita taj broj, on zna koliko znakova sledi iza njega i tako utvrđuje kraj okvira. Mana ove metode (i razlog zbog kojeg se danas ne koristi) je ta što ukoliko se broj znakova pogrešno očita zbog greške u prenosu, onda odredišni računar ne može naći početak narednog okvira.

2. Po drugoj metodi se problem ponovnog sinhronizovanja posle greške zaobilazi tako što se početak i kraj svakog okvira obeležavaju posebnim bajtovima. Ranije su se početni i krajnji bajt razlikovali ali u poslednje vreme većina protokola za obeležavanje koristi isti indikatorski bajt označen kao FLAG. Na taj način ukoliko primalac izgubi korak on samo treba da potraži indikatorski bajt da bi pronašao kraj tekućeg okvira. Problemi nastaju prilikom prenosa binarnih podataka kada se lako može dogoditi da se sekvenca bitova indikatorskog bajta nađe u podacima i to obično ometa ispravno učitavanje okvira. Zato se ubacuju specijalni bajtovi (kontrolni znak ESC) ispred svakog indikatorskog bajta koji se slučajno nađe unutar podataka. Ovo nazivamo umetanje bajtova/znakova. Takođe ukoliko se u među podacima nalazi i specijalni bajt njega ćemo takođe morati označiti specijalnim bajtom.

3. Umetanje bitova radi na sledeći način: Svaki okvir počinje i završava se specijalnom sekvencom bitova 01111110 a kad god sloj veze pošiljaoca u podacima nađe na pet uzastopnih jedinica, on automatski umeće nulu u izlazni tok bitova. Kada primalac u dolaznom toku bitova pronađe pet uzastopnih jedinica iza kojih sledi nula, on iz toka izbacuje tu nulu. Ova umetanja i izbacivanja (i bitova i bajtova) su potpuno nevidljiva mrežnom sloju na oba računara. Ova metoda se primenjuje samo na mrežama u kojima je kodiranje na nivou fizičkog medijuma u izvesnoj meri redundantno (npr ukoliko se jedan bit podataka kodira sa dva fizička bita).

Mnogi protokoli veznog sloja zbog veće sigurnosti kombinuju prebrojavanje znakova sa nekom od drugih metoda. Kada okvir stigne, koristi se polje sa brojem znakova za određivanje kraja okvira i okvir se prihvata samo ako se na njegovom kraju nalazi odgovarajući graničnik i ako se kontrolni zbir slaže.

#### 14. Kodирање грешака у слоју везе

Nakon što su početak i kraj svakog okvira označeni postavlja se pitanje kako znamo da su svi okviri isporučeni mrežnom sloju i to ispravnim redom. Pouzdanost isporuke se obezbeđuje time što pošiljalac dobija neku povratnu informaciju o tome šta se dešava na drugom kraju veze. (Protokol najčešće zahteva od primaoca da povratno pošalje okvire sa potvrdom o pristiglim okvirima). Tako pošiljalac zna da li je slanje uspelo ili ga je potrebno ponoviti. Otežavajuća okolnost je mogućnost hardverske greške pri kojima okvir potpuno nestane i tada primalac nema na šta da reaguje dok pošiljalac biva blokiran čekajući potvrdu prijema. Ovo se prevazilazi tajmerima koji su podešeni na vremenski interval koji je potreban da okvir stigne na odredište, tamo se obradi i pošalje potvrda o njegovom prijemu. Prilikom slanja okvira pokreće se i tajmer i ukoliko sve bude ispravno potvrda o prijemu stiže pošiljaocu pre isteka tajmera dok se u suprotnom utvrđuje da je došlo do problema. Najlakše rešenje jeste da se okvir ponovo pošalje međutim onda se rizikuje da primalac dva puta primi okvir pa se okvirima dodeljuju redni brojevi tako da primalac može da razlikuje ponovo poslate okvire od onih koji su stigli uobičajenim redom.

Zbog prirode fizičkih procesa koji ih izazivaju, greške se često ne javljaju pojedinačno, već u rafalima. To ima svoje prednosti i mane u odnosu na izolovane greške. Prednost je što se podaci šalju u blokovima bitova i ako se svaki od njih sadrži od npr. 1000 bitova a učestalost grešaka je 0,001 po bitu onda bi u slučaju izolovanih grešaka skoro svi blokovi imali grešku, dok bi se u slučaju rafala grešaka oni javili u samo dva-tri od sto blokova. Mana rafalnih grešaka jeste to što se teže uklanjuju.

## 15. Детекција грешака у слоју везе

Kodovi za otkrivanje grešaka imaju smisla prilikom prenosa podataka putem bakarnih ili optičkih kablova kao i u drugim slučajevima u kojima je učestalost grešaka mala (za razliku od bežičnog) i gde je lakše samo proveriti da li je do greške došlo i ako jeste poslati ponovo pogrešne podatke. Da bi se greška uklonila u blokovima od 1000 bitova potrebno je 10 kontrolnih bitova sto je za megabit podatka 10000 kontrolnih bitova, dok je za određivanje toga koji blok sadrži jednu jednobitnu grešku potreban jedan bit parnosti po bloku, to znači da na 1000 blokova je potrebno poslati samo jedan kontrolni blok. Međutim dodavanjem samo jednog bita parnosti po bloku se verovatnoća otkrivanja rafalnih grešaka smanjuje na 50%. Ovo se može znatno povećati ukoliko se blok posmatra kao matrica sa n kolona i k redova pa se bit parnosti posmatra za svaku kolonu i zapisuje u poslednjem redu matrice. U tom slučaju se matrica šalje red po red i kada svi stignu na odredište primalac proverava kontrolne bitove i ukoliko su neispravni zahteva ponovno slanje čitavog bloka. Ovo otkriva rafalnu grešku dužine n, ali ne i n+1.

U praksi se primenjuje:

Цикличка провера редундансе (CRC ili polinomski kod)

- Секвенца битова се сматра полиномом чији су коефицијенти само нуле и јединице
  - За датих n битова података, генеришемо k контролних битова тако n+k битова буде број дељив са неким унапред одабраним полиномом односно бројем C
  - Број C се назива генератор(generatorski polinom) и знају га и пошиљалац и прималац (prvi i poslednji bit moraju biti јединице)
- 
- Пример:
    - 10011010 је полином  $x^7 + x^4 + x^3 + x^1$
    - Даље радимо са бинарним вредностима и аритметиком по модулу 2 (игноришемо потенцијалне преносе са највише позиције)
  
  - Слање података:
    1. Проширити n битова података са k нула
    2. Поделити број са одабраним бројем C
    3. Задржимо само остатак, количних нас не занима
    4. Додамо остатак на проширенi број
  
  - Примање података:
    1. Поделити поруку са бројем C и проверити да ли је остатак при дељењу једнак нули

Ostatak otkriva pojedinacne greske (tamo gde se nalazi jedinica, taj bit je invertovan) osim u slučajevima da je greška polinom čiji je činilac C, kada se greška propušta. U slučaju polinoma sa neparnim brojem koeficijenata, svaka greška se može uhvatiti ukoliko uvedemo  $(X+1)$  kao činioca broja C.

Ukoliko kod ima r kontrolnih bitova, mogu biti otkrivene sve greške koje nisu duže od r. Ukoliko je greška duža biće nula samo ako se poklapa sa C (verovatnoća je  $(1/2)^{n(r-1)}$ )

- Од генератора зависи и квалитет детекције
- CRC се често користе у:
  - Ethernet, 802.11, ADSL, Кабловска, ...
- Контролни збир, иако лошији, користи се на вишим слојевима
  - IP, TCP, UDP ...
- Провера парности
  - Слабо се користи

## 16. Корекција грешака у слоју везе

Ovi mehanizmi se koriste na kanalima kod којих је учесталост појаве грешака веома велика (безичне везе)

- Кодна реч се састоји од D битова података и R контролних битова
- Пошиљалац:
  - Израчуна R контролних битова као функцију од D и потом пошаље свих D+R битова
- Прималац:
  - Прихвата D+R са потенцијалним грешкама на било ком од битова
  - Рачуна R' контролних битова на основу D по истом алгоритму као пошиљалац; грешка ако R није исто као R'

Интуиција иза кодова за грешке

- Скуп валидних кодних речи је доста мањи од скупа свих могућих
- Случајно одабрана реч има малу шансу да буде и валидна што значи мала шанса да ће се десити грешка и да ће након тога реч и даље бити валидна.

### Хамингово растојање

- Хамингово растојање је минималан број инверзија битова потребних да се од једне речи добије нека друга реч
- Хамингово растојање кода минимално Хамингово растојање између свих парова валидних кодних речи
- Обично су у делу за податке могуће све комбинације док је скуп контролних битова ограничен
- Детекција грешака:
  - Да би се поуздано открило до  $d$  грешака, Хамингово растојање кода мора бити најмање  $d+1$
  - Тада је немогуће да  $d$  једнобитних грешака промене валидну кодну реч у неку другу валидну кодну реч
- Корекција грешака:
  - За код са Хаминговим растојањем  $2d+1$ , највише  $d$  грешака се увек може исправити до најближе исправне валидне речи

### Интуиција иза кодова за корекцију грешака

- Ако имамо Хамингов код са растојањем најмање 3 ( $HD \geq 3$ )
  - Потребно нам је  $\geq 3$  битова да пређемо из једне валидне речи у другу
  - Једнобитне грешке ће, дакле, бити ближе некој јединственој валидној речи
- Под претпоставком да се може десити само једнобитна грешка, можемо да исправимо реч тако што ћемо је мапирати у најближу валидну
  - Дакле, приступ ради ако  $HD \geq 2d + 1$ , где је  $d$  максимални очекивани број једнобитних грешака

Bitovi kodne reči numerišu se uzastopno, počevši od levog kraja. Bitovi čija pozicija odgovara stepenu dvojke ( $1, 2, 4, \dots$ ) predstavljaju kontrolne bitove, dok se na ostalim pozicijama nalazi m bitova podataka. Svaki kontrolni bit vodi računa da određeni skup bitova (uključujući i njega) ima paran broj jedinica. Za utvrđivanje тога који kontrolni bitovi воде računa о poziciji k потребно је само napisati k kao zbir stepena dvojke. Kada добије кодну реč, primalac inicijalizuje свој бројач на нулу. Potom проверава сваки kontrolni bit у погледу исправне парности. Kad utvrdi neslaganje, primalac помера бројач за k. Ukoliko је nakon провере свих kontrolnih bitova бројач остао нула, онда се реč приhvata. Ako se nakon провере у бројачу налази број različit od nule онда он представља poziciju invertovanog bita.

Hamingovi kodovi mogu ispravljati само pojedinačне greške, међутим могуће је применити ih i za исправљање rafalnih grešaka uz jednu izmenu. Sekvenca od k uzastopnih kodnih reči se poređa u matricu tako да је свака реč засебан red. Potom слати матрице по колонама, jer ће онда у случају rafalnih grešaka дужине do k biti изменjen највише један bit у свакој кодној речи, што Hamingov kod може исправити. Помоћу ovoga se km bitova podataka imunizuje на једну rafalnu grešku највеће дужине k uz korišćenje kr kontrolnih bitova.

## 17. Слој везе, типови сервиса, окружење, утопијски једносмерни протокол

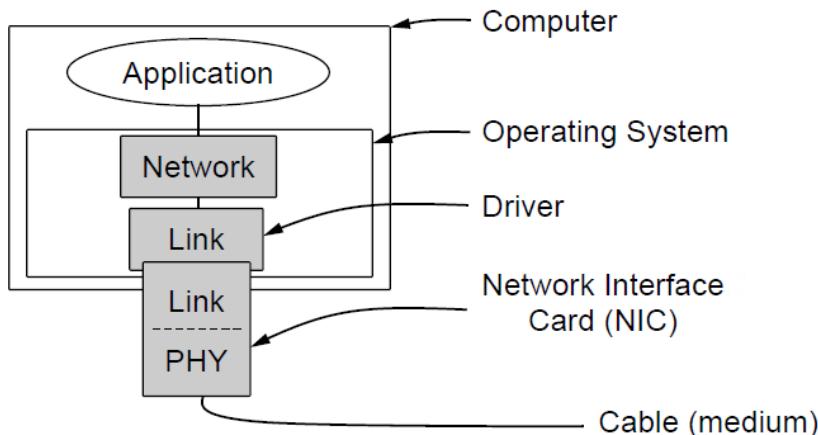
### Типови сервиса

- Сервис без успоставе везе и без потврде пријема
  - Оквир се шаље независно и без ретрансмисије у случају грешке

- Пример је Етернет
- Сервис без успоставе везе са потврдом пријема
  - Ради се ретрансмисија ако се јави потреба
  - Пример је WiFi
- Сервис са успоставом везе и са потврдом пријема
  - Успостава везе омогућава да подаци теку истим редом којим су и послати
  - Ретко се користи

Окружење у слоју везе

- Овај слој је обично реализован делом на мрежној картици, а другим делом на нивоу оперативног система



- Преглед неких функција за интеракцију слоја везе са слојем испод и изнад:

Група	Функција	Опис
Мрежни слој	from_network_layer(&packet) to_network_layer(&packet)	Узима пакет из мрежног слоја Прослеђује пакет мрежном слоју
Физички слој	from_physical_layer(&frame) to_physical_layer(&frame)	Приhvата оквир из физичког слоја Прослеђује оквир ка физичком слоју
Догађаји & тајмери	wait_for_event(&event) start_timer(seq_nr) stop_timer(seq_nr) start_ack_timer() stop_ack_timer()	Чека на пакет/оквир/истек тајмера Покреће тајмер Прекида тајмер Покреће тајмер аз оквир потврде ACK Зауставља тајмер за оквир потврде ACK

Пошиљалац шаље оквире стално

Прималац је у стању да их све прихвати

## Osnovni protokoli sloja veze podataka

Što se tiče sloja veze, paket koji mu kroz interfejs stigne od mrežnog sloja sačinjavaju samo podaci koje do poslednjeg bita treba isporučiti mrežnom sloju na odredištu. Kada sloj veze podataka prihvati paket, on ga kapsulira u okvir dodajući mu zaglavlj i završni blok koji se odnose na sloj veze. Znači, okvir se sastoji od ugrađenog paketa, nešto upravljačkih podataka u zaglavlj i kontrolnog zbiru u završnom bloku. Okvir se prenosi do sloja veze na drugom računaru uz pretpostavku da postoje odgovarajuće bibliotečke procedure za slanje/primanje okvira. Transportni hardver izračunava i priključuje kontrolni zbir (kao završni blok), pa se softver sloja veze ne mora baviti time. Primalac na početku ne treba ništa da radi već samo da čeka da se nešto dogodi (wait\_for\_event) i kada se dogodi da utvrdi tip događaja. Kada okvir stigne primaocu, hardverski se izračunava kontrolni zbir, ako se on ne složi, o tome se obaveštava sloj veze podataka. Ukoliko je je okvir stigao neoštećen, sloj veze se obaveštava i o tome. Čim je sloj veze primaoca preuzeo neoštećen okvir, on proverava upravljačke informacije u zaglavlj i ako je sa njima sve u redu, prosleđuje ostatak paketa mrežnom sloju. Zaglavlj okvira nikada ne stiže u mrežni sloj. Razlog za ovo jeste to što se tako protokoli mrežnog sloja i sloja veze podataka ne mešaju, nego ostaju nezavisni i mogu se menjati bez međusobnog uticaja. Protokoli:

- Utopijski jednosmerni protokol
- "Stani i čekaj" protokol za kanal bez grešaka
- "Stani i čekaj" protokol za kanal sa greškama

Utopijski jednosmerni protokol:

Podaci se prenose samo u jednom smeru, mrežni slojevi su uvek spremni i na izvorištu i na odredištu. Vreme obrade je zanemarljivo. Na raspolaganju su baferi neograničene veličine i komunikacioni kanal između slojeva veze nikada ne izobličava, niti gubi okvire. Protokol se sastoji od dve jasno odjeljene procedure: pošiljaoca i primaoca koje se izvršavaju u sloju veze izvorišnog/odredišnog računara. Okviri se ne numerišu niti se šalju potvrde o prijemu.

17. Контрола тока, ARQ, паузе (тјмаути), дупликати, протокол „стани и чекај“ за савршен и несавршен канал

Kontrola toka podrazumeva usklađivanje prenosa podataka u slučaju kada pošiljalac i primalac nemaju iste brzine slanja/primanja podataka.

ARQ - Automatic Repeat reQuest - protokol kod kojeg pošiljalac čeka pozitivnu potvrdu pre nego što pošalje sledeći paket

- ARQ se obично koristi kada su gрешке uobičajne i kada se moraju исправити
  - Нпр. WiFi, и TCP (касније)
  - Основна идеја:
    - Прималац шаље потврду о пријему исправног оквира ACK
    - ACK је такође оквир
    - Пошиљалац аутоматски шаље поново након временске паузе (тјмаута), осим ако у међувремену пристигне ACK
- ARQ кроз несавршен канал

## Кључна питања у вези ARQ?

- Две битне одлуке:
  - Колика треба да буде пауза (време на тајмеру)?
  - Како да се избегне интерпретирање дуплираних оквира као нових?
- Основни циљ је коректност
  - исправни оквири, без грешака и дупликата
- Али желимо и да комуникација буде ефикасна

## Паузе (тајмаути)

- Пауза треба да буде:
  - Не превише велика (неискоришћеност канала)
  - Не премала (непотребне трансмисије)
- Дужину паузе релативно једноставно одредити за LAN
- Јасна је горња граница, мало одступање
- Тешко за Интернет
- Велико одступање, нема јасне горње границе

## Jednosmerni protokol "stani i чекай" za savršen kanal

Odbacujemo najmanje realističnu pretpostavku protokola - da мrežni sloj primaoca može trenutno da obradi sve dolazne podatke(uz постојање неограниченог бафера у слоју везе primaoca koji može da čuva sve пристигле податке до trenutka njihove обраде). Još uvek važe pretpostavke da канал radi nepogrešivo i da саобраћај иде само у једном смеру. Главни проблем јесте да спречимо пошиљаoca да затрпа primaoca подацима које он не може довољно брзо да обради. Уколико је у време потребно да прихвати оквир и проследи га мреžном слоју онда пошиљалac мора да шalje оквire brzinom koja je manja od jednog okvira tokom vremena t. Уколико prepostavimo da se kod primaoca ne vrši privremeno складиštenje i стављање података у red за чекање онда пошиљалac никада не sme da пошаљe novi оквир dok prethodni не preuzme procedurra from\_physical\_layer() da ga ne bi prebrisao. Jedna od opcija јесте процена понашања primaoca u najnepovoljnijim uslovima i prilagođavanje brzine slanja tome, међутим ово води ka niskom iskorišćenju punog opsega. Opštije rešenje bi bilo da primalac, nakon što проследи paket svom mrežnom слоју, šalje povratno mali prazan оквир koji пошиљаocu dozvoljava da пошаљe sledeći оквир sa подацима. Sa друге стране, пошиљалac шalje paket i потом чека dok ne stigne mali prazan оквир(potvrda o prijemu).

```
void sender2(void)           void receiver2(void)
{
    frame s;                  {
    packet buffer;
    event_type event;
    while (true) {
        from_network_layer(&buffer);
        s.info = buffer;
        to_physical_layer(&s);
        wait_for_event(&event);
    }
}

frame r, s;
event_type event;
while (true) {
    wait_for_event(&event);
    from_physical_layer(&r);
    to_network_layer(&r.info);
    to_physical_layer(&s);
}
```

Iako se prenos podataka odvija u jednom smeru, okviri putuju u oba pa je neophodno da kanal omogući dvosmerni prenos informacija. Međutim, predmetni protokol strogog reguliša saobraćaj, tako da se on u jednom trenutku odvija samo u jednom smeru i taj posao može obaviti poludupleksni fizički kanal.

#### Protokol za jednosmerno slanje podataka bučnim kanalom

Rešavamo se prepostavke da je kanal savršen tj. okviri se mogu oštetiti ili potpuno izgubiti, međutim, prepostavljamo da će hardver primaoca otkriti okvir oštećen u prenosu kada izračuna njegov kontrolni zbir. Kao najjednostavnije rešenje nameće se uvođenje tajmera. Pošiljalac šalje okvir, ali primalac šalje poruku samo ako ga primi u ispravnom stanju, u suprotnom (ako je okvir oštećen) primalac ga odbacuje. Pošiljalac čeka potvrdu i po isteku tajmera ponovo palje paket. Opisana šema međutim ima propust jer mrežni sloj na odredišnom računaru nema informaciju o tome da li je neki paket izgubljen ili je neki paket duplikat, pa sloj veze mora da garantuje isporuku paketa bez duplikata, što u ovom slučaju nije moguće. Primer ukoliko se izgubi okvir sa potvrdom, deo podataka će biti dupliran. Zato je potrebno naći nacin da primalac razlikuje okvir koji vidi prvi put od okvira koje je ponovo poslat. Trivijalno rešenje je da pošiljalac stavi redni broj u zaglavje svakog okvira jer na taj način primalac može proverom rednog broja utvrditi da li je u pitanju novi okvir ili duplikata koji treba odbaciti. Kako je poželjno da zaglavje okvira bude što manje, a i slanje  $m+1$ -og okvira je uslovljeno slanjem  $m$ -tog okvira, za kodiranje rednog broja dovoljan je jedan bit. U svakom trenutku primalac očekuje da usledi određeni redni broj. Svaki pristigli okvir koji ne ispunjava očekivanje odbacuje se kao duplikat. Kada stigne okvir sa očekivanim rednim brojem, on se prihvata i prosleđuje mrežnom sloju, a zatim se redni broj uvećava za 1 po modulu 2.

Pošto pošalje okvir, pošiljalac uključuje tajmer, ukoliko on radi, vraća ga nulu da bi se čekao pun interval. Rok posle kojeg će se tajmer automatski uključiti treba izabrati tako da okvir ima vremena da stigne do primaoca, da tamo bude obrađen u najnepovoljnijim uslovima i da se potvrda o njegovom prijemu vrati do pošiljaoca i samo kad ovaj tajmer istekne treba prepostaviti da su se poslati okvir ili njegova potvrda izgubili u putu i treba poslati duplikat. Ukoliko je rok prekratak, biće nepotrebni slanja koja neće uticati na greške, ali hoće na performanse. Nakon uključivanja tajmera, pošiljalac čeka jedan od tri moguća ishoda: da mu stigne neoštećena potvrda o prijemu, da mu stigne oštećena potvrda o prijemu ili da se tajmer isključi. Ukoliko stigne ispravna potvrda, pošiljalac uzima sledeći paket od svog mrežnog sloja i smešta ga u bafer i uvećava redni broj, u ostala dva slučaja sadržaj bafera i rednog broja ostaju nepromenjeni i šalje se duplikat.

#### 18. Protokoli kliznih prozora

U prethodnim protokolima su se okviri prenosili samo u jednom smeru, međutim, većinom postoji potreba da se okviri istovremeno prenose u oba smera. To se može postići pomoću dva paralelna komunikaciona kanala za jednosmeran saobraćaj, ali time se dobijaju dva zasebna fizička kola, svako sa kanalom za slanje(podataka) i za primanje(potvrda). U oba slučaja se povratni kanal koristi neefikasno, već je bolje koristiti isti kanala za slanje podataka u oba smera. Ideja je da umesto da pošalje zaseban upravljački okvir kada stigne okvir sa podacima, primalac sačeka da mu njegov mrežni sloj prosledi sledeći paket podataka i da potvrdu priključi odlaznom okviru sa podacima(u polju ack u zaglavljtu). Ova tehnika se naziva šlepovanje. Osnovna prednost jeste efikasnije korišćenje propusnog opsega kanala, jer polje je polje ack dugačko samo nekoliko bitova dok se zaseban okvir sastoji od zaglavlja potvrde i kontrolnog zbitka. Ali ovo donosi komplikacije koje nisu postojale prilikom slanja zasebnih potvrda, a to je

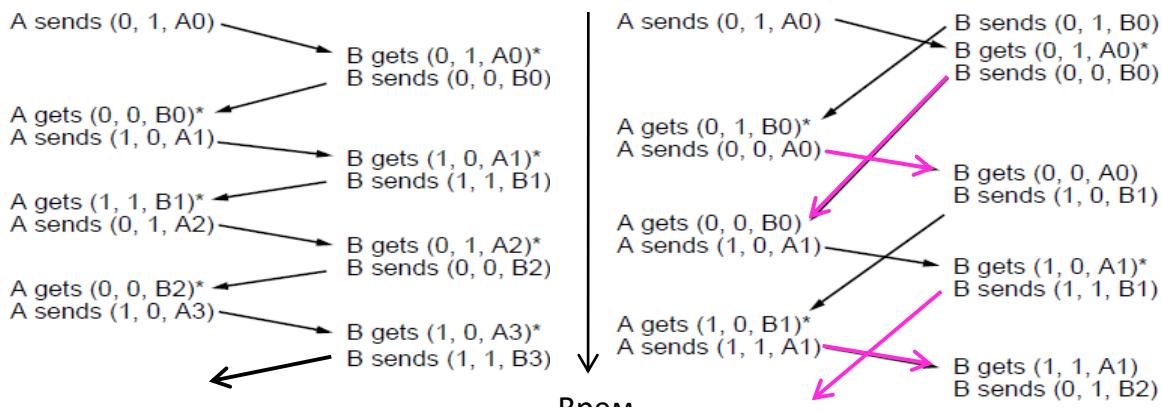
koliko dugo sloj veze treba da čeka paket na koji bi nakačio potvrdu. Ukoliko čeka duže od zadatog roka tajmera, onda će okvir biti ponovo poslat i gubi se poenta sistema potvrđivanja. Ovo se radi tako što sloj veze na računaru primaoca čeka određeni broj milisekundi i ukoliko mu stigne novi paket sa mrežnog sloja, potvrda se kači za njega dok se u suprotnom šalje zaseban okvir sa potvrdom. Suština protokola kliznih prozora jeste što pošiljalac u svakom trenutku čuva skup rednih brojeva okvira koje sme da pošalje i oni predstavljaju prozor za slanje. Slično tome, primalac održava prijemni prozor koji odgovara skupu rednih brojeva okvira koje sme da primi. Prozori pošiljaoca i primaoca ne moraju imati iste granice, niti biti iste velicine. Iako ovi protokoli sloju veze daju više slobode u pogledu redosleda slanja i primanja okvira, nije isključen zahtev da protokol mora da na odredišni mrežni sloj prosledi pakete isti redom kojim su prosleđeni sloju veze na izvoru. Redni brojevi unutar pošiljačevog prozora odgovaraju okvirima koji su poslati ili se mogu poslati, ali čiji prijem još nije potvrđen, kad god nov paket stigne iz mrežnog sloja, dodeljuje mu se prvi najviši redni broj, a gornja granica prozora povećava se za jedan. Kada stigne potvrda, donja granica se poveća za 1. Ako je maksimalna veličina prozora  $n$ , onda pošiljalac mora imati  $n$  bafera za čuvanje nepotvrđenih okvira i ukoliko se dostigne ta maksimalna veličina, sloj veze pošiljaoca mora da nasilno zatvori mrežni sloj dok se ne osloboди neki bafer. Svaki pristigli okvir koji je izvan prozora primaoca se odbacuje. Kada stigne okvir čiji je redni broj jednak donjoj granici prozora, on se prosleđuje mrežnom sloju, generiše se potvrda i prozor rotira za jedinicu.(slika)

#### Jednobitni protokol kliznih prozora

Ne postoji odvojenih algoritama za pošiljaoca i primaoca, jer sada i jedan i drugi mogu da šalju i primaju

- Za ACK koristimo okvir iz suprotног правца („шлепање“)
- Radi nad nesavršenim kanalom
- Omogућава контролу тока

#### Radi po principu "stani i čekaj"

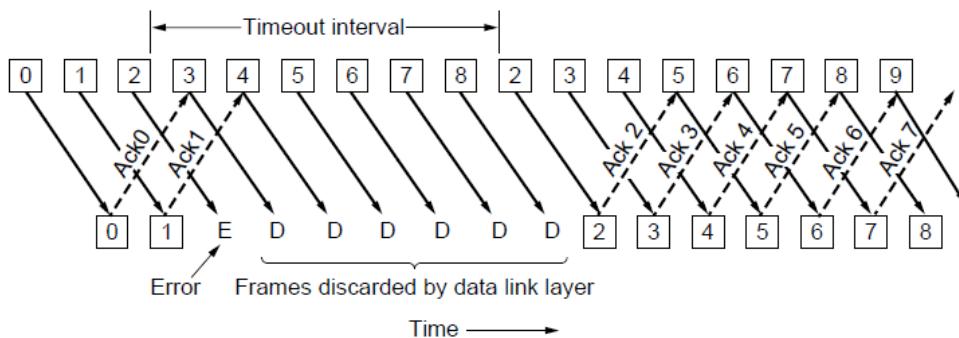


Ponekada je prepostavka da putovanje okvira od primaoca i vraćanje potvrde pošalijocu traje zanemarljivo malo nije održiva. Kombinacija velike propusne moći i male dužine okvira je katastrofalna sa gledišta efikasnosti i to zbog striknog pravila da pošiljalac čeka potvrdu prijema pre nego što pošalje sledeći paket. Ukoliko ovo olabavimo efikasnost se povećava.

U osnovi je rešenje da se pošiljaocu dozvoli da pre blokiranja pošalje više paketa umesto jednog. Uz pogodan izbor vrednosti w pošiljalac bi mogao neprekidno da pošalje okvire tokom vremena potrebnog okviru za obilazak veze a da ne ispuni svoj prozor. Pošiljalac treba da ima veliki prozor ako je velika vrednost propusnog opsega i vremena obilaska veze. Šta raditi ako se desi da se okvir koji se nalazi u sredini rafala ošteti ili izgubi? Dva protokola koji se bave tim problemom su "Vrati se N" i protokol selektivnog ponavljanja.

#### „Врати се N“ протокол

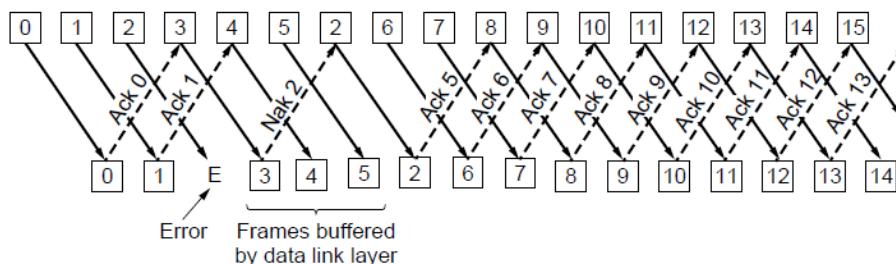
- Прималац приhvata само оквире који стижу редом:
- Притом одбацује све који су уследили након спорног
- Примаоцу истиче у неком моменту тајмовут и он шаље све непотврђене оквире до краја прозора поново



#### Протокол селективног понављања

Ovde se neispravan okvir odbacuje, ali se ispravni okviri koji ga slede smeštaju u bafer. Kada istekne rok tajmera pošiljaoca, on šalje samo najstariji nepotvrđeni okvir. Ako na odredište stigne u ispravnom stanju, primalac može da mrežnom sloju isporuči redom i sve okvire iz bafera.

- Прималац приhvata оквир све док је редни број оквира у опсегу дефинисаном клизним прозором
- NAK (негативни ACK) информише пошиљаоца да поново пошаље проблематични оквир (зарад ефикасности), пре истека тајмаута за цео прозор
- Карактеристике:
  - Сложенији за имплементацију од „Врати се N“ због баферисања при примању и вишеструким тајмерима при слању
  - Ефикаснија употреба протока, јер се сада само изгубљени оквири поново шаљу





## **20. MAC podsloj, uloga, alokacija kanala, ALOHA protokol**

U svakoj mrezi u kojoj se koristi neusmereno emitovanje, glavni problem je kako odrediti ko će koristiti kanal u situaciji kad ima vise korisnika. Protokoli kojima se odredjuje sledeći korisnik takvog kanala pripadaju podsloju sloja veze podataka, poznatom kao podsloj za upravljanje pristupom medijumima (MAC). Podsloj MAC je posebno vezan za lokalne mreze, pogotovo one bezicne koje su po prirodi neusmerene.

Tema – nacin dodeljivanja jedinstvenog kanala za neusmereno emitovanje u situaciji kada ne njega pretenduje vise korisnika.

Staticko dodeljivanje kanala:

Klasican nacin dodeljivanja jedinstvenog kanala, kao sto je telefonski vod, jednom od pretendenata na njega jeste da mu se kapacitet isecka nekim od nacina multipleksiranja, recimo multipleksiranjem podelom frekvencija (FDM). Ako postoji N potencijalnih korisnika, propusni opseg se deli na N jednakih delova po jedan za svakog korisnika. Posto tako svaki korisnik ima svoje privatno frekventno područje, oni se međusobno ne ometaju. Kada je u pitanju mali I nepromenljiv broj korisnika, od kojih svaki ima ujednacen ili gust saobracaj, takva podela je jednostavno i efikasno resenje. Međutim, kad je broj posiljalaca veliki I promenljiv ili se saobracaj odvija u rafalima, tehnika FDM ne zadovoljava. Ako se propusni opseg izdeli na N područja a u nekom trenutku zeli da komunicira manje od N korisnika, veliki deo propusnog opsega ostace neiskoriscen. Ako, pak, zeli da komunicira vise od N korisnika, neki od njih to neće moći zbog nedostatka propusnog opsega, cak i u slučaju da neki od korisnika kojima su frekventna područja vec dodeljenja nista ne emituju niti primaju. Sve sto je receno za FDM, vazi i za ostane nacine staticke podele kanala. Da primenimo multipleksiranje podelom vremena(TDM) i svakom korisniku dodelimo po jedan od N vremenskih intervala, on biva protracen ako ga ne upotrebi korisnik kojem je dodeljen. Ne može dobro da se izbori sa saobracajem koji se odvija u rafalima.

Dinamicko dodeljivanje kanala:

Pet osnovnih prepostavki:

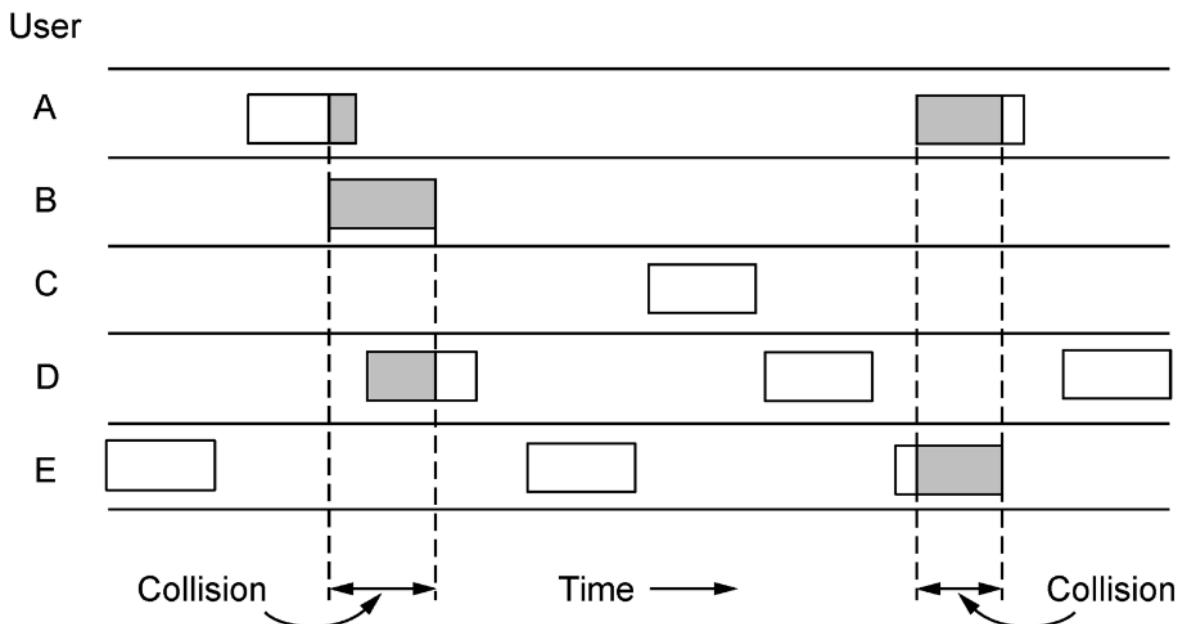
1. Nezavisnost saobracaja. Model sadrži N nezavisnih stanica (računara ili telefona) koje generisu okvire za slanje, bilo da to radi program ili korisnik. Kada generise okvir, stanica se blokira i ne ističe sve dok okvir ne bude uspesno poslat.
2. Jedinstven kanal. Za sve komuniciranje na raspolaganju je samo jedan kanal. Stanice mogu da emituju samo preko njega i samo preko njega da primaju okvire. Prepostavila se da su sve stanice jednakih mogućnosti, iako im protokoli mogu dodeljivati razlike uloge.
3. Sukobljavanja se mogu otkriti. Ako se dva okvira istovremeno emituju, oni se vremenski preklapaju, što rezultuje izobiljenim signalom. Taj dogadjaj se naziva sukobljavanje. Sve stanice mogu otkriti da je doslo do sukobljavanja. Okvir koji se sukobio sa drugim okvirom mora biti ponovo poslat.
4. Neprekidan vremenski tok ili raspodeljeno vreme. Može se prepostaviti da je vremenski tok neprekidan, zbog čega se paket može poslati u bilo kojem trenutku. S druge strane, može se prepostaviti i da je vreme raspodeljeno u intervalima određene veličine. U tom slučaju slanje se uvek podudara s pocetkom intervala.
5. Ima ili nema osluskivanja saobracaja na nosiocu podataka. Uz prepostavku da osluskivanja nema, stanica može da proveri da li je kanal slobodan pre nego što ga sam upotrebi. Nijedna stanica ne pokusava da emituje dokle god cuje tudi saobracaj na kanalu. Ukoliko nema osluskivanja saobracaja na nosiocu podataka, stanice ne proveravaju da li je kanal prazan, već odmah emituju okvire. Tek kasnije mogu da utvrde da li je prenos obavljen uspesno.

## ALOHA:

Racunarska mreza koja je povezivala Havajska ostrva krajem 1960-ih godina.

### Cista ALOHA:

Ideja-dozvoliti korisnicima da emituju uvek kada imaju podatke za slanje. Naravno da ce u tom slučaju biti sukobljavanja, posiljaoci moraju imati neki nacin da ustanove ima li sukobljavanja. U sistemu ALOHA, centralni racunar odmah vraca svaki primljeni okvir kao odgovor, tj. neusmereno ga emituje svim stanicama. Dakle, ako posiljalac osluskuje kanal, uvek moze ustanoviti da li je njegov okvir ispreavno primljen. Kada se poslati okvir osteti, posiljalac ce cekati odgovor tokom slučajno odabranog vremenskog intervala I zatim isti okvir jednostavno poslati ponovo. Vreme cekanja mora biti nasumicno odabрано, inace ce se isti okviri sudarati neprestano na isti nacin.



Svi prikazani okviri su iste velicine jer je to neophodno za postizanje maksimalnog protoka u ovom sistemu. Cak I ako se samo prvi bit novog okvira preklopi sa poslednjim bitnom prethodnjem okvira, oba okvira propadaju I moraju se kasnije ponovo slati.

### Vremenski raspodeljena ALOHA:

Ubrzo nakon pustanja sistema ALOHA u rad, Roberts je objavio metodu dupliranja kapaciteta takvog sistema. On je predlozio da se vreme izdeli u vremenske intervale konacne duzine I da svaki interval odgovara jednom okviru. Takav pristup podrazumeva dogovaranje korisnika oko granica vremenskih intervala. Jedan nacin da se to postigne bio bi da posebna stanica emituje odgovarajuci signal na pocetku svakog intervala, slicno satu. Stanici je zabranjeno da emituje odmah kada korisnik otkuca novi red vec je duzna da saccea pocetak sledeceg vremenskog intervala.

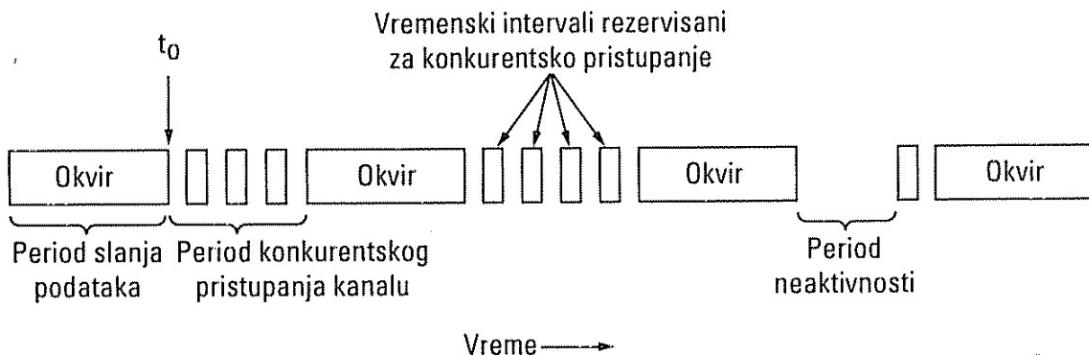
## 21.CSMA, CSMA/CD, BEB (poboljsaj!)

### CSMA:

Prvi protokol za pristup uz osluskivanje nosioca podataka zove se 1-trajni CSMA. Kada stanica ima podatke za slanje, ona prvo oslusne kanal kako bi utvrdila da li je zauze. Ako se na kanalu nista ne cuje, stanica salje svoje podatke. Ukoliko na kanalu ima saobracaja, stanica ceka da on utihne, pa tek onda salje svoj okvir. Kada dodje do sukobljavanja, stanica ceka tokom nasumicno odabranog perioda I sve pocinje od pocetka. Protokol nosi ime 1-trajni, zato sto stanica emituje s verovatnocom 1 kada utvrdi da je kanal prazen.

### CSMA/CD:

Ako dve stanice utvrde da je kanal prazan I pocnu istovremeno da emituju, njihovi signali ce se sukobiti. Jos jedno ponasanje bilo bi da stanice brzo otkrivju sukob I odmah prekidaju emitovanje svojih okvira, koji su ionako nepopravljivo izobliceni. Time se stedi vreme I propustni opseg. Taj protokol poznat je kao CSMA uz otkrivanje sukoba (CSMA/CD). Hardver stanice mora da osluskuje kabl dok ona emituje. Ako se ono sto cuje razlikuje od onoga sto salje, ona zna da je doslo do sukobljavanja.



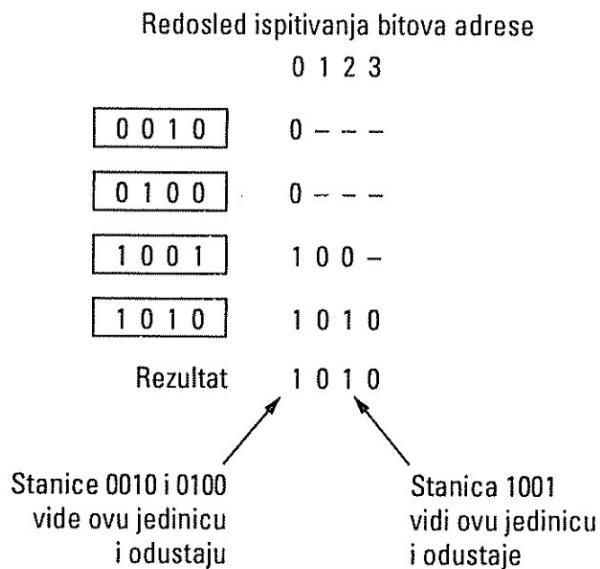
**Slika 4-5.** Protokol CSMA/CD može da bude u jednom od tri stanja: konkurentskog pristupanja, slanja podataka ili neaktivnosti.

U trenutku  $t_0$  stanica je dovršila slanje okvira. Sada bilo koja druga stanica koja ima okvir spreman za slanje može to i da ucini. Ako dve ili više stanica rese da istovremeno pocnu emitovanje, bice sukobljavanja. Kada stanica otkrije sukob, ona prekida emitovanje, ceka tokom proizvoljno odabranog vremenskog intervala I tada pokusava da ponovo emituje. Prema tome protokol CSMA/CD sastoji se od naizmeničnih pokusaja pristupanja kanalu I perioda slanja podataka, uz povremenu neaktivnost kada sve stanice miruju.

### BEB:

Kada stanica zeli da emituje, ona objavljuje svoju adresu u obliku binarnog niza, pocinjujuci od najznačajnijeg bita. Sve adrese imaju istu duzinu. Kada se posalju istovremeno, bitovi na korespondentnim pozicijama adrese razlicitih stanica medusobno se u kanalu podvrgavaju operaciji logičke disjunkcije. Opisani protokol zvacemo binarno odbrojavanje. U protokolu se zanemaruje kasnjenje u prenosu, tako da sve stanice prakticno trenutno vide potvrdjene bitove. Da bi se izbegli sukobi, mora se primeniti neko pravilo odredjivanja prvenstva: cim stanica u cijoj je adresi najznačajniji bit 0 ustanovi da je taj bit prebrisana bitom, ona se povlaci.

Primer, ako stanice sa adresama 0010, 0100, 1001 I 1010 pokusavaju da istovremeno izadju na kanal, u toku intervala predvidjenog za prvi bit one emituju 0, 0, 1, 1. Ti bitovi se povrgavaju operaciji disjunkcije, dajuci rezultat 1. Kada vide rezultat, stanice 0010 I 0100 znaju da na kanal pretenduju stanice s visim adresama I zato odustaju. Stanice 1001 I 1010 nastavljaju da konkurisu jedna drugoj. Rezultat disjunkcije bitova na sledecoj poziciji je 0, tako da su obe stanice I dalje u igri. Treca disjunkcija daje 1, pa stanica 1001 odustaje. U ovoj rundi pobedjuje stanica 1010 jer ima najvisu adresu. Ona tada može da posalje okvir s podacima, posle cega pocinje nova runda takmicenja.



Slika 4-7. Protokol binarnog odbrojavanja. Crtica označava neaktivnost.

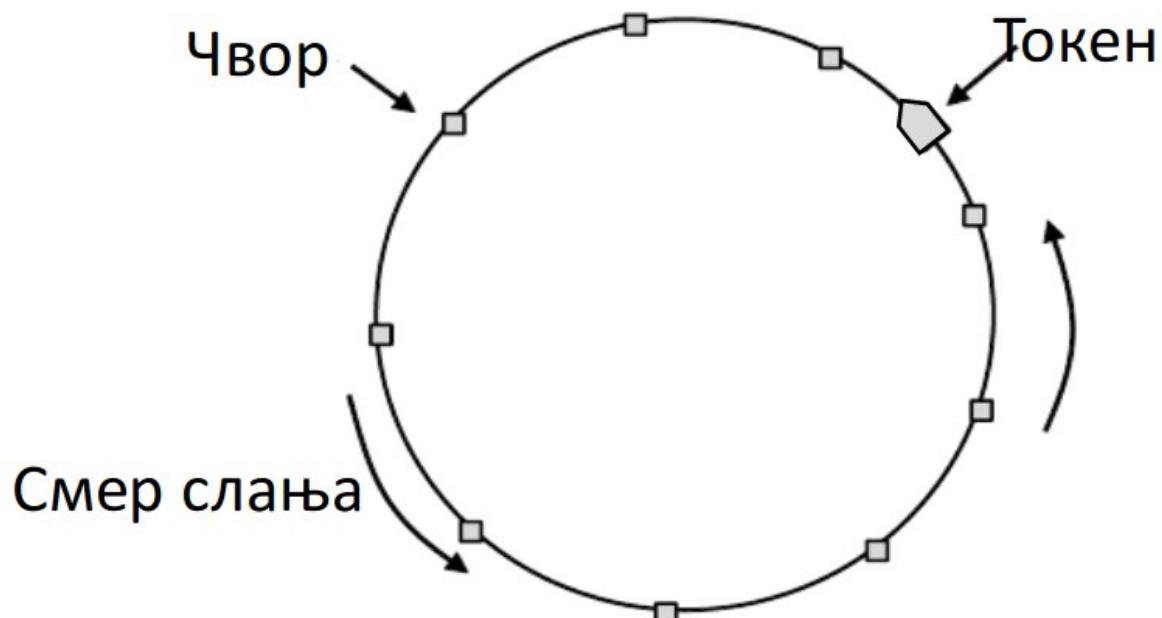
## 22. MAC protokoli zasnovani na redosledu

Definise se uredjenje prema kojem cvorovi salju ako imaju nesto da posalju, ili samo propuste ukoliko nemaju sta da posalju.

Prsten sa tokenom:

Organizujemo cvorove u prsten, token se potom prosledjuje u krug. Ako cvor u trenutku prijema tokena ima okvir u redu cekanja na slanje, moze ga poslati pre nego sto zeton prosledi sledecem cvoru. Ukoliko cvor nema svoj okvir u redu cekanja na slanje, token odmah prosledjuje sledecem cvoru. Na taj nacin okviri kruze po prstenu I dostizu cvor koji im je odrediste. Medjutim, da okvir ne bi beskonacno kruzio, cvor koji ga je prvobitno poslao ga uklanja, nakon sto je okvir obisao pun krug, ili ga uklanja cvor koji je bio predvidjeni primalac okvira.

Nakon slanja okvira, svaki cvor mora sacekati da svih N cvora posalje token susednom cvoru I da ostalih N-1 cvorova posalje po jedan okvir, ako ga imaju.



Prednosti:

Unapred odredjeni dodatni troskovi I nema kolizija, efikasnije pod visokim opterecenjem.

Nema nesredjenih cvorova. Garantovan servis, svi cvorovi mogu da ocekuju da ce biti opsluzeni u nekom unapred definisanom vremenu.

Mane:

Slozenost. Vise stvari moze da podje naopako, sta ako se izgubi token?

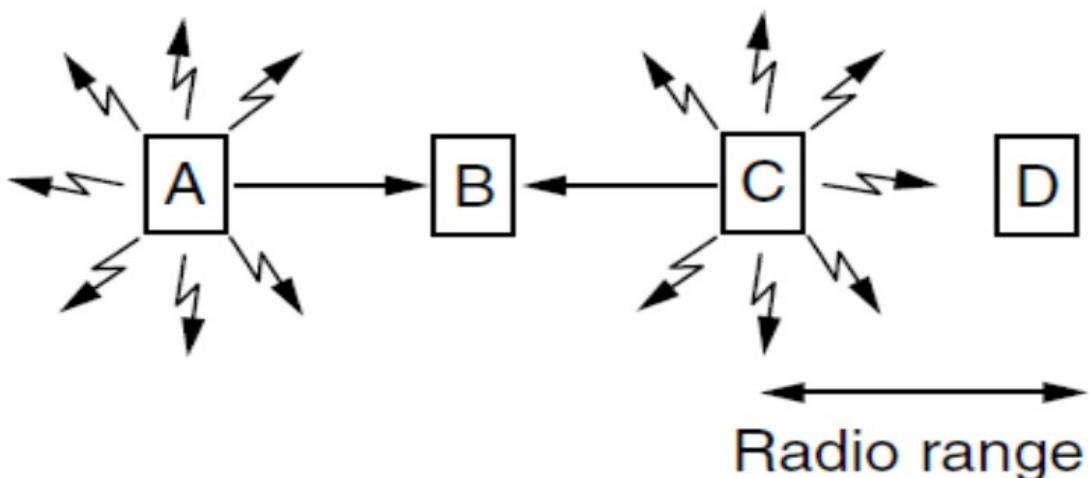
Relativno visok dodatni trosak pri malom opterecenju.

Protokoli sa redosledom u praksi:

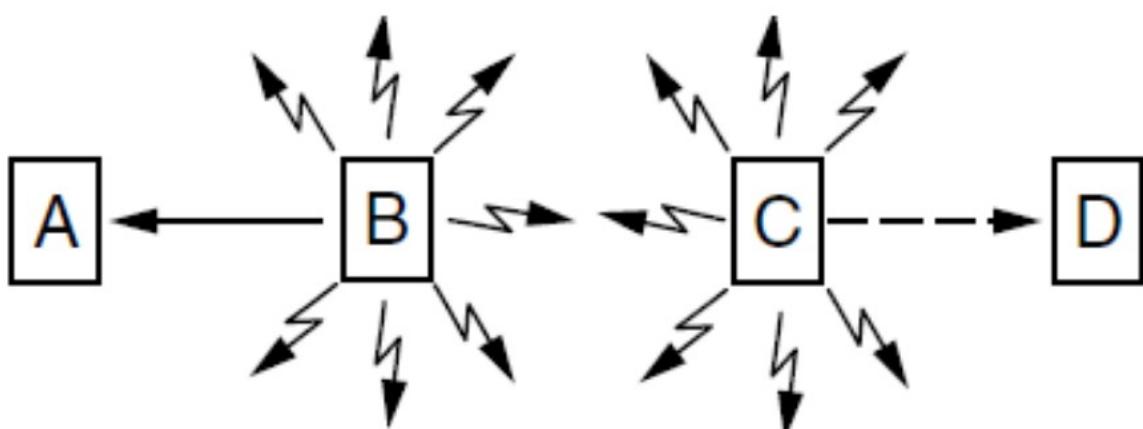
Obicno se isprobaju kao poboljsanje. Medjutim, protokoli sa slucajnoscu se obicno tesko nadmasuju. Jednostavnii I dovoljno dobri najcesce. Dobro se prosiruju na veci broj cvorova (skaliraju).

### 23.MAC protokoli bezicne mreze

CSMA protokol- osluskivacemo emisije drugih stanica I emitovat tek onda kad se sve utisaju. Taj protokol u bezicnim mrezama ipak nece raditi dobro, jer su problemi smetnje kod prijemnika a ne kod predajnika.



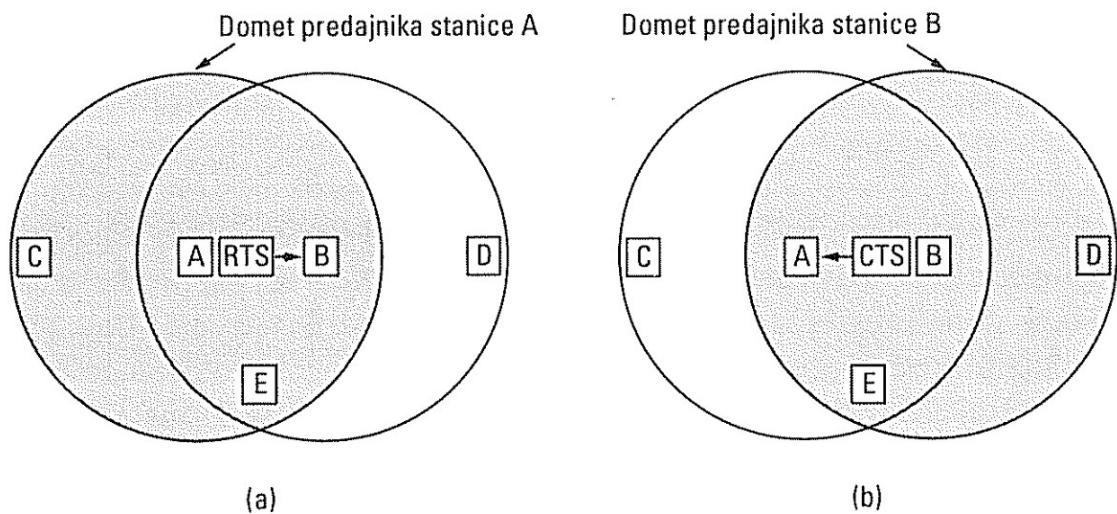
Stanice A I C salju podatke stanici B. Ako stanica A emituje, a C osluškuje medijum, ona nece cuti stanicu A jer se nalazi izvan njenog dometa I zato ce pogresno zakljuciti da moze da posalje podatke stanicu B. Ako stаница C pocne da emituje, ometace stanicu B I upropastiti okvir koji joj salje stаница A. Treba nam MAC protokol koji ce spreciti ovu vrstu sukobljavanja koje arci propusni opseg. To je problem skrivenog terminala.



Stanica B salje podatke stanici A, a C istovremeno hoce da posalje podatke stanici D. Ako stanica C oslususkuje medijum, ona ce cuti tu emisiju I pogresno zaključiti da ne sme da salje podatke stanici D (nacrtano isprekidanom linijom), iako bi takva emisija ometala samo zonu izmedju stanica B I C, u kojoj nema nijednog prijemnika. Treba nam MAC protokol koji ce spreciti ovu vrstu odustajanja kojim se arci prpusni opseg. To je problem iznozenog terminala.

MACA:

MACAkoristi proceduru "rukovanja". Protokol za visekorisnicki pristup uz izbegavanje sukoba. Prvi potez povlaci posiljalac podsticuci primaoca na emitovanje kratkog okvira koji ce ucutkatiobliznje stanice tokom slanja narednog (velikog) okvira s podacima. Ova tehnika se upotrebljava umesto osluskivanja saobracaja.



**Slika 4-12.** Protokol MACA. (a) Stanica A šalje okvir RTS stanici B.  
(b) Stanica B odgovara stanici A okvirom CTS.

Pravila

1. Posiljalac emituje kratki okvir RTS (Request-To-Send, sa informacijom o duzini okvira koji hoce da salje).
2. Primalac emituje kratki okvir CTS (Clear-To-Send, sa informacijom o duzini okvira preuzetoj iz RTS).
3. Posiljalac kada dobije CTS, pocinje slanje, dok drugi okviri koji vide CTS, a nisu slali RTS, cekaju u skladu sa duzinom iz CTS.

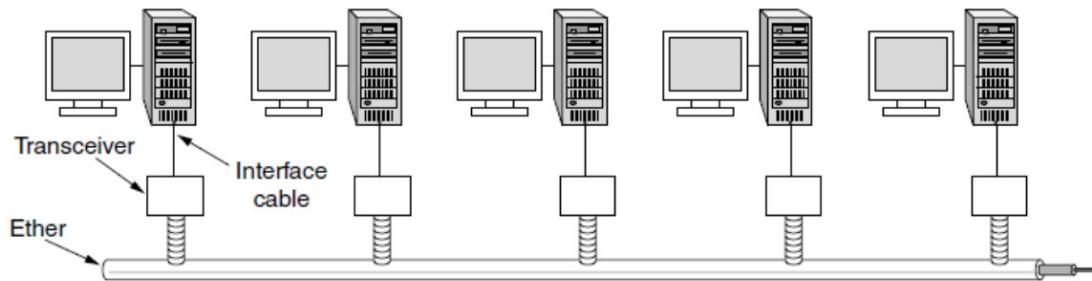
protokola:

Pogledajmo kako reaguju okolne stanice koje slusaju ovaj razgovor. Stanica koja uhvati RTS Okvir izvesno je blizu stanice A I zato mora da cuti dovoljno dugo da CTS okvir stigne do stanice A bez sukobljavanja. Stanica koja uhvati CTS Okvir nalazi se blizu stanice B I mora da cuti tokom narednog prenosa podataka, cije trajanje saznaje isptujuci CTS okvir.

Kolizije su I dalje moguce, ali manje verovatne. Na primer, obe stanice (B I C) mogu da stanici A istovremeno posalju RTS okvire koji ce se sukobiti I propasti. U takvoj situaciji neuspesan predajnik (onaj koji nije registrovao CTS okvir tokom ocekivanog vremenskog intervala), ponovo ce poslati RTS okvir posle proizvoljno izbaranog intervala mirovanja.  
**(POGLEDATI SLAJDOVE!)**

## 24. Klasicki Eternet – IEEE 802.3

Najpopularniji vid organizovanja LAN tokom osamdesetih I devedesetih. Radio je brzinama od 3 do 10 Mb/s. Klasicki Eternet je krivudao po zgradu u obliku jedinstvenog debelog kabla na koji su se prikljucivali svi racunari. Ta arhitektura je prikazan na slici.



Prva varijanta, popularni Eternet, podsecala je na zuto bastensko crevo, sa oznakama na svaka 2,5 metra na mestima gde treba prikljuciti racunare. Nasledio ga je tanki Eternet, koji se mogao lakse saviti. Stanice se na njega povezuju pomocu standardizovanih BNC konektora. Tanki Eternet je mnogo jeftiniji I lakse se instalira.

U svim verzijama Eterneta, segment kabla ima ogranicenu duzinu po kojoj se signal moze prostirati bez pojacivaca. Da bi se ostvarile vece mreze, segmenti kablova mogu se nadovezivati pomocu tzv. repetitora. Repetitor je uredjaj fizickog sloja koji prima, pojacava I ponovo salje signal u oba smera.

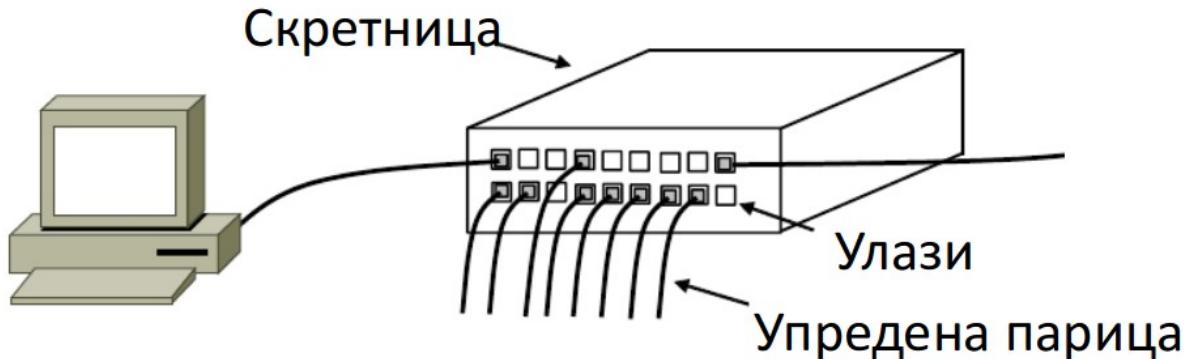
Na slici je prikazan format za slanje okvira. Okvir pocinje Preambulom duzine 8 bajtova, koja je uvek predstavljena nizom bitova 10101010 (uz izuzetak poslednjeg bajta, u kojem su poslednja 2 bita 11). U standardu 802.3, taj osmi bajt je iskoriscen kao granicnik Početka okvira. Poslednja dva bita (dve jedinice) kazuju primaocu da sledi ostatak okvira. Potom dolaze adrese izvorista I odredista dugacke po 6 bajtova. Najznačajniji (prvi) bit odredisne adrese je 0 za obične adrese a 1 za grupne adrese. Grupna adresa omogućava da vise stanica prima poruke preko jedne adrese. Sledeće polje je Tip ili Duzina, zavisno od toga da li je okvir Eternet ili IEEE 802.3. Eternet pomocu polja Tip saopstava primaocu da radi sa okvirom. Ovim poljem specificira se proces kome treba predati okvir. Zatim dolaze Podaci, polje duzine do 1500 bajtova. Ta, pomalo proizvoljna granica, izabrana je zbog toga sto pirmopredajnik mora imati dovoljno radne memorije da prihvati citav okvir. Osim sto postoji njegova maksimalna duzina okvira, postoji I njegova minimalna duzina. Iako je polje s podacima duzine 0 bita ponekad koristno, ono izaziva probleme. Ako je posle s podacima krace od 46 bitova, okvir se u polju Dopuna dovodi do minimalne duzine. Poslednje polje okvira je Kontrolni zbir. To je u stvari 32-bitna ciklicna provera redudanse (CRC). CRC je kod za otkrivanje gresaka, kojim se utvrđuje da li su bitovi okvira ispravno primljeni. Ona samo otkriva greske I tada se okvir odbacuje.

Bajtovi	8	6	6	2	0–1500	0–46	4	
(a)	Preamble	Odredišna adresa	Izvořišna adresa	Tip	Podaci	Dopuna	Kontrolni zbir	
(b)	Preamble	SOF	Odredišna adresa	Izvořišna adresa	Dužina	Podaci	Dopuna	Kontrolni zbir

## 25. Moderni (komutirani) Eternet

Jezgro ovog sistema je skretnica, cija brza osnovna ploca povezuje sve prikljucke. Spolja skretnica izgleda kao razvodnik. Oba uredjaja izgledaju kao kutije. Svaki kabl povezuje skretnicu ili razvodnik po jednim umrezenim racunarom, kao sto je prikazano na slici. Kod klasicnog je

povezivanje bilo nadovezivanjem kabala (izlaz iz jednog računara, ulaz u naredni, itd) topologija magistrale. Dodavanje nove stanice svodi se prosti na to da se zica utakne u kutiju (odnosno izvadi iz nje). Vecinu kvarova je lako otkiri, posto los kabl ili prikljucak obicno utice na samo jednu stanicu. I dalje postoji jedna zajednicka komponenta koja moze da kaze – sama skretnica – ali kada sve stanice “izgube mrezu”, osoblje odrzavanja zna kako da otkloni kvar: treba zameniti skretnicu.



Skretnice salju okvira samo onim prikljucnima kojima su ti okviri upuceni. Kada odredjeni prikljucak skretnice primi Eternet okvir od neke stanice, skretnica procita njegovu Eternet adresu da bi saznala kojem prikljucku je okvir namenjen. To znači da skretnica mora znati koji prikljuci odgovaraju kojim adresama.

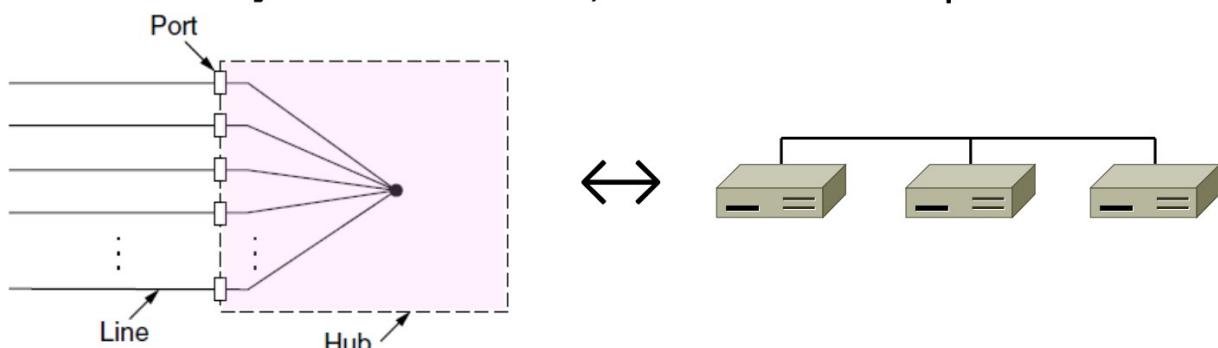
Razvodnik:

Električki povezuje sve na sebe prikljucene zice, kao da su zaledljene.

Ulaz se ponavlja na sve ostale izlaze.

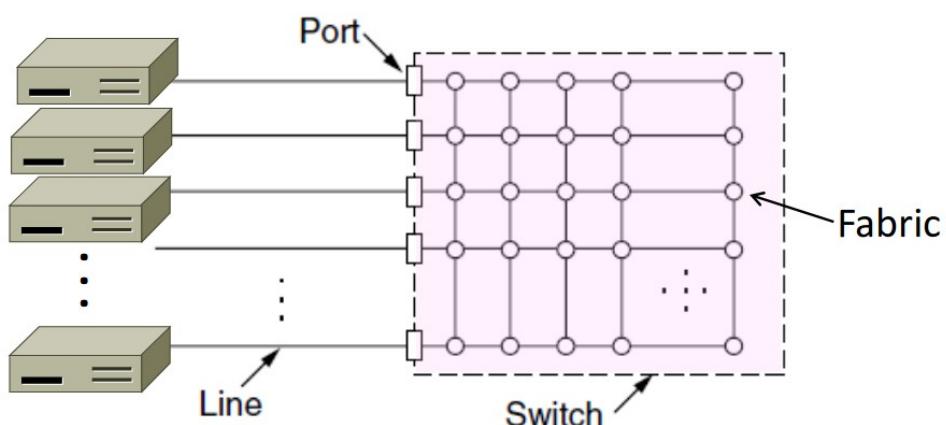
Ulazno/izlazne prikljucke cemo zvati portovi.

Svestan je samo bitova, ne zna za okvire.

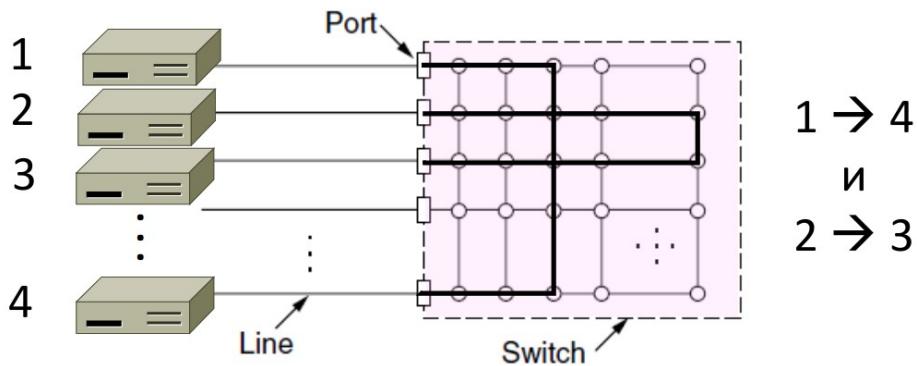


Svic:

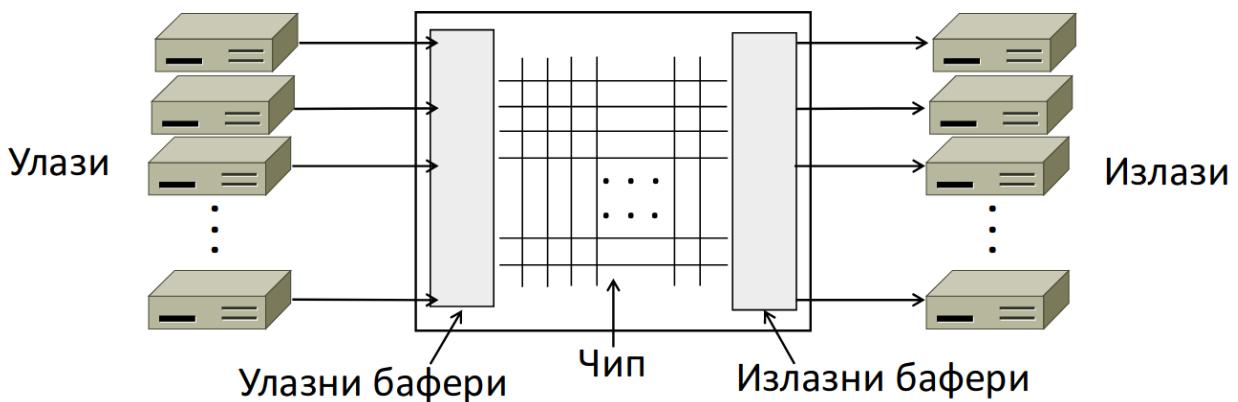
Radi na sloju veze; koristi adrese iz okvira kako bi prosledio ulaz na zeljeni izlazl visestruki okviri se mogu poslati istovremeno.



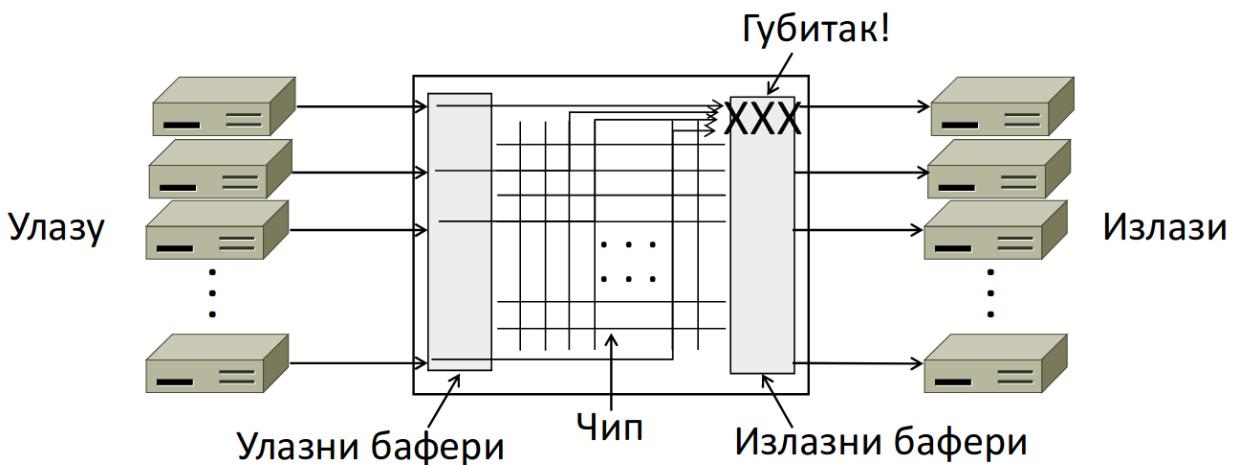
Port je obicno puni dupleks.



Potrebbni su baferi kada npr. Visestruki ulazi ciljaju isti port (crtamo ulaze I izlaze odvojeno zbog preglednosti).



Veliko opterecenje moze da dovede do prelivanja bafera I gubitka okvira.

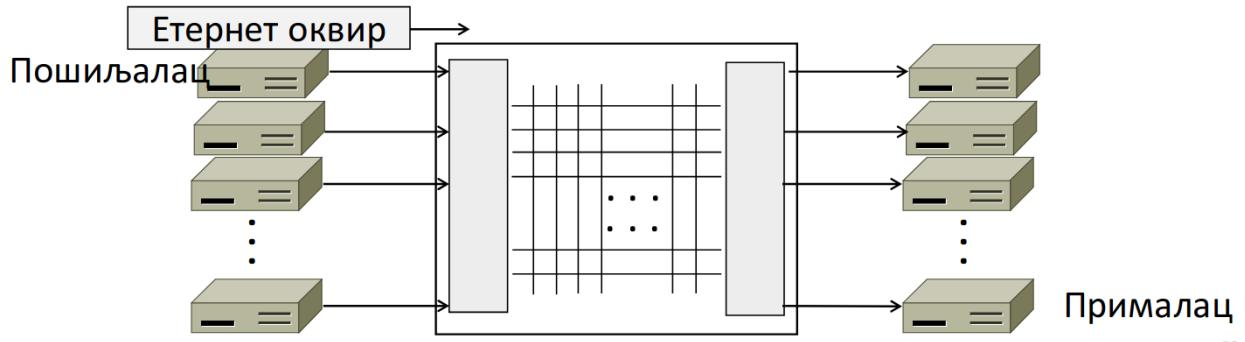


Prednosti skretnica:

Skretnice I razvodnici su zamenili koncept deljenih kablova iz doba klasicnog Etherneta. Prakticnije je dovesti sve zice na jednu lokaciju. Pouzdanije nego klasicni Eternet. Kvar na jednoj zici nema uticaj na veci deo mreze. Kvar se lako pronalazi, ako ne radi cela mreza, znaci da je problem u skretnici odnosno razvodniku. Skretnice omogucavaju poboljsani protok, npr 100 Mb/s po ulazno/izlaznoj liniji, umesto 100Mb/s za celu mrezu (deljeni kabl).

Prosledjivanje podataka:

Skretnica treba da pronađe odgovarajući port na osnovu adrese primaoca iz Eternet okvira. Dodatno zelimo da možemo da pomestamo cvorove (isključujemo i uključujemo u različite portove).



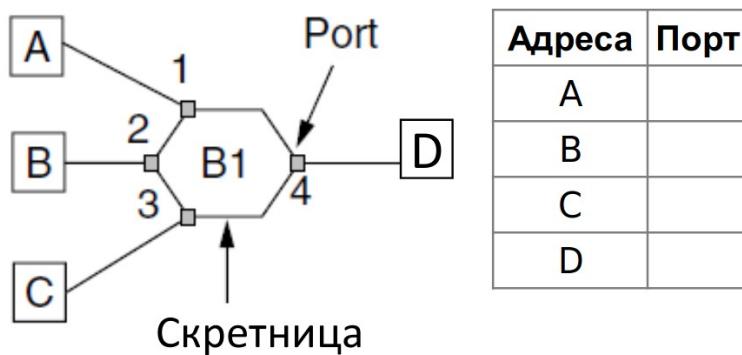
58

Ucenje unazad:

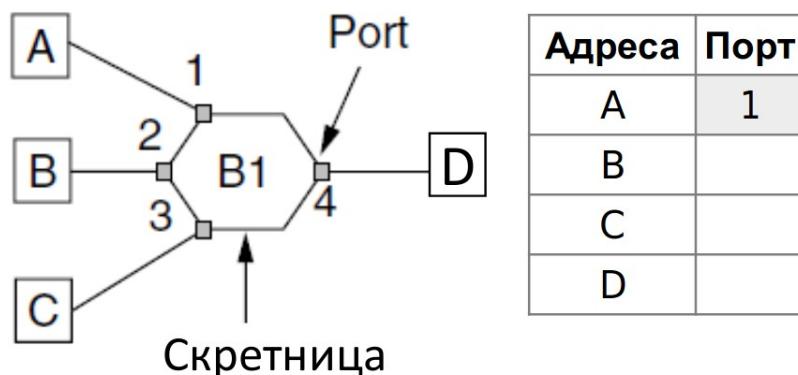
Prosledjivanje okvira na osnovu relacija između broja porta I adrese iz okvira:

1. Da bi se popunila ova tabela, posmatramo adrese I portove cvorova koji salju okvire
2. Ako se za zadatu adresu u tabeli nalazi pridruženi port, onda posalji samo njemu, inace posalji svim portovima

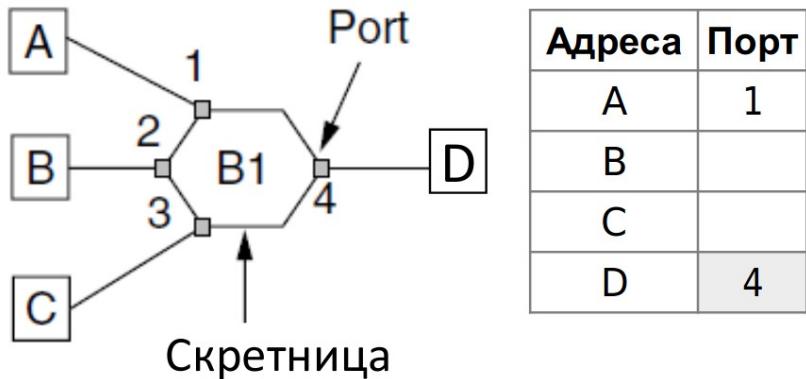
1: A salje ka D



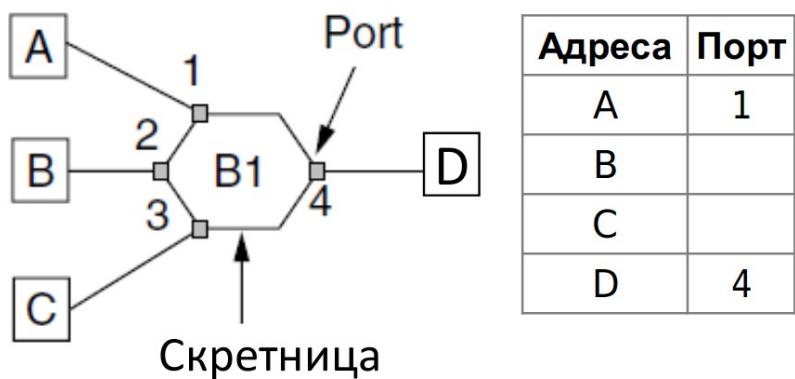
2:D salje ka A



3:A salje ka D

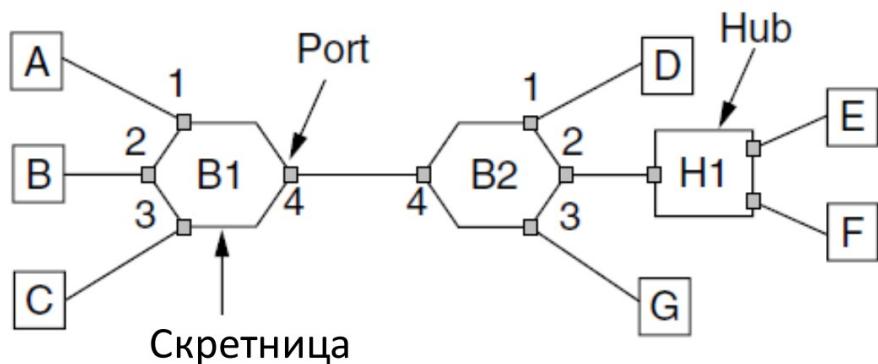


3:A salje ka D

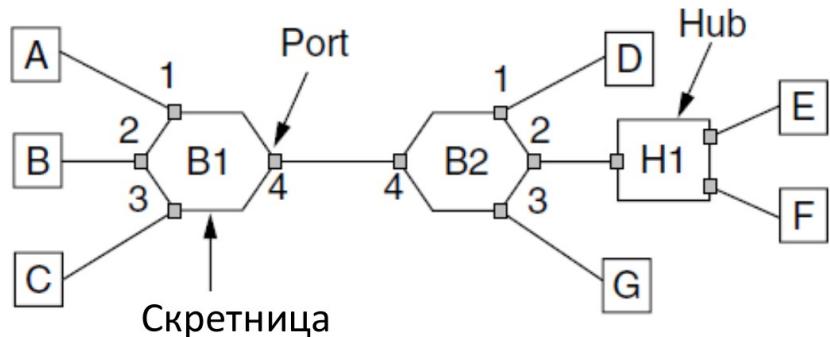


Visestruke skretnice I razvodnici:

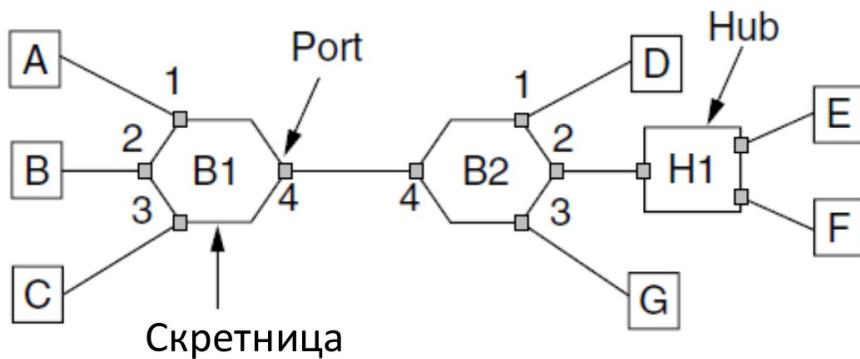
Nije problem, pod pretpostavkom da nema petlji.



Npr. A salje ka D, I posle D salje ka A. Svaka skretnica pravi svoju odvojenu tabelu.



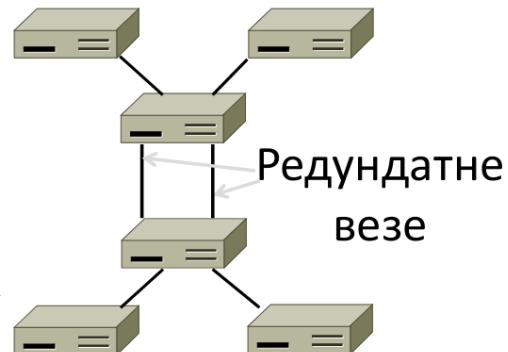
Npr. Skretnica B2 ce portu 4 dodeliti adresu A. Kasnije ako stigne zahtev od B, port 4 ce imati A I B pridruzeno.



#### Petlje u racunarskim mrezama:

Kako da povezemo skretnice na proizvoljan nacin, a da mreza I dalje radi. Problem sa petljama. Mreze se obicno projektuju a namerno imaju redundantnost, tako da stalno nailazimo na petlje.

Konacno resenje-formiranje razapinjujuceg stabla, stablo nema cikluse a ipak izmedju svaka dva cvora postoji put.



## **26. Mrežni sloj, uloga, motivacija, rutiranje i prosleđivanje (ukratko), tipovi servisa na mrežnom sloju, objašnjenja i njihov uporedni odnos.**

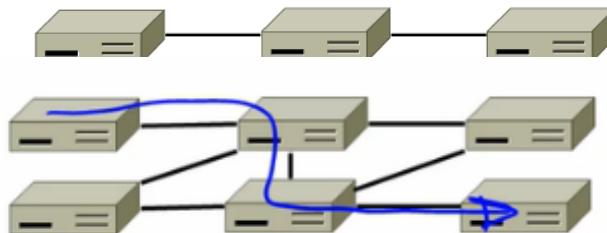
### **Uvod**

Zadatak mrežnog sloja je da pakete od izvorišta sproveđe celim putem do odredišta, što znači da treba da ih provuče kroz brojne usputne usmerivače (rutere). Taj zadatak jasno odudara od funkcije sloja veze – jednostavnog premeštanja paketa s jednog kraja kabla na drugi. Mrežni sloj je najniži sloj koji se bavi prenosom od jednog do drugog kraja mreže.

Da bi ispunio svoj zadatak, mrežni sloj mora da poznaje topologiju mreže (tj. skup svih usmerivača i veza između njih) i da kroz nju bira odgovarajuće putanje, čak i u slučaju velikih mreža. Kada bira putanju, on takođe mora da vodi računa o tome da neke linije i usmerivače ne preoptereti a da drugi ne rade ništa. I na kraju, kada se izvorište i odredište nalaze na različitim mrežama, pojavljuju se nepredviđeni problemi koje mrežni sloj mora da rešava na licu mesta.

### **Šta će nam mrežni sloj?**

Videli smo da već možemo da izgradimo mrežu upotrebom veza i skretnica. I možemo da šaljemo frejmove od jednog računara ka drugom koji su povezani skretnicama.



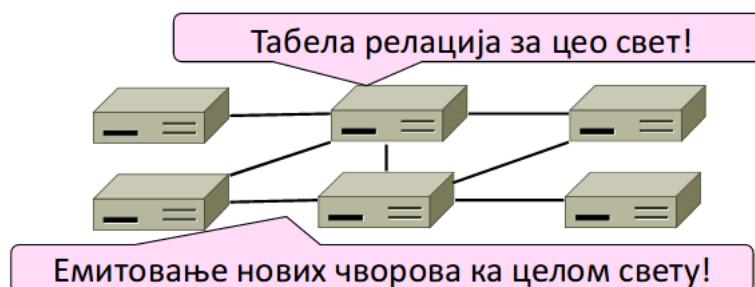
Problemi sa skretnicama:

#### **1. Metode koje smo videli se ne skaliraju dobro**

Kako se mreža povećava, dostiže desetine, stotine, hiljade do milione različitih hostova. Kako bi se onda pristup sa skretnicama skalirao?

**A. Tabela relacija bi postala ogromna.** Svaka skretnica u ovom dijagramu čuva tabelu za svaku moguću destinaciju kuda treba da ide. Tabela će imati milione ulaza. To je mnogo jer svako mora da održava ovu tabelu.

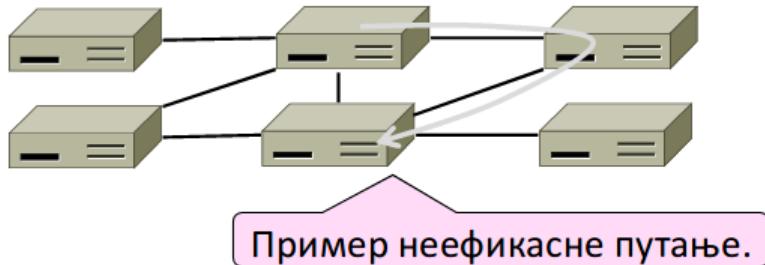
**B. Drugi problem je inicijalno emitovanje celom svetu.** Da bi pristupile novom članu za koga nisu ranije čuli, skretnice emituju. To znači da kada prvi put šaljete nešto na novu destinaciju, to može biti poslatno preko cele mreže. Zamislite da šaljete paket na novu destinaciju i da on prolazi kroz milione drugih hostova u mreži samo zato što ne znate kojim putem treba da idete. Zato ovaj dizajn ne skalira onako kako bismo mi želeli.



- Pristup sa skretnicama koji smo videli **ne radi preko ako su tehnologije sloja veze različite**. To je pristup za sloj veze tako da je dizajniran za određeni sloj veze. Mi smo već videli različite veze. Ethernet – u velikim korporacijama, 802.11 u mnogim kućama, 3G za mobilnu koneksiјu... to su sve različiti tipovi veza. Mi želimo da budemo u mogućnosti da šaljemo pakete od hosta koji je npr. 802.11 do host koji na Ethernet-u. Ali pristup sa skretnicama ne uzima ovo u obzir, nije opremljen da poveže ove različite tehnologije.

### 3. Ne omogućavaju kontrolu saobraćaja

Razapinjuće stablo je veoma impresivno, (???), ali ne garantuje da je put koji je pronašao dobar put. Zato saobraćaj u razapinjućem drvetu može da prati ovaj put:



Iako je postojao direkstan put. To često nije nešto što želimo, jer je najdragocenija stvari u mreži protok i želimo da ga dobro iskoristimo, zato moramo da imamo dobru kontrolu nad putanjom kojom paketi prolaze.

U pristupu mrežnog sloja želimo da rešimo sve probleme sa kojima smo se susreli.

Posmatramo kako ćemo da skaliramo mrežu, umesto da šaljemo na pojedinačnu destinaciju, moći ćemo da odradimo sve rutiranje i prosleđivanje kroz mrežu, da odradimo sav posao ruteru u pogledu blokova IP adresa koji se zovu prefiksi. Primenjujemo hijerarhiju tako da svi različiti ruteri imaju manje tabele koje neće nužno imati milione ulaza za hostove.

Takođe ćemo videti da mrežni sloj ima podršku za heterogenost. IP koristi pristup zvan internet working u kome su različite tehnologije sloja veze spojene zajedno.

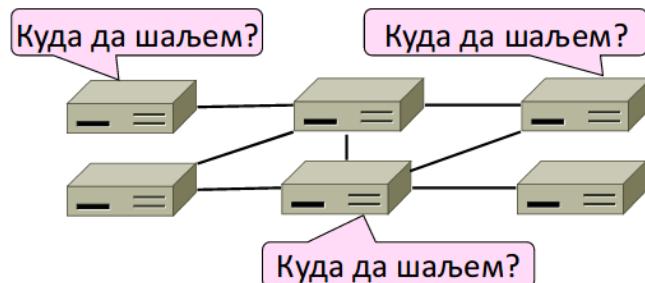
Na kraju, videćemo različite načine da imamo bolju kontrolu nad korišćenjem protoka, umesto razapinjućeg drveta, videćemo rutiranje najmanje ili najkraće cene, što je drugačiji frejmwork za odlučivanje kojim ćemo putem ići. Videćemo I kvalitet usluge. Kvalitet usluge se odnosi na to na koje načine možemo dobro da iskoristimo protok koji imamo.

## Rutiranje i prosleđivanje

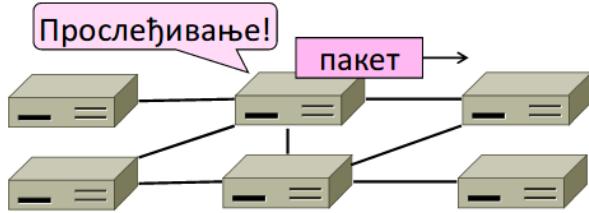
Potrebno je napraviti razliku između rutiranja i prosleđivanja. To su dve različite stvari. Malo je zbumujuće zato što su ruteri na internetu odgovorni za obe ove stvari – i za rutiranje i za prosleđivanje.

**Rutiranje** je proces odlučivanja u kom pravcu treba poslati saobraćaj.

To je proces u kome svi ruteri učestvuju. Uključuje razgovor sa svim susedima radi saznanja ko je gde i generalno traženja najboljeg načina, najbolje veze za korišćenje da bi se dostigle sve različite destinacije. Dakle, to je globalni proces, pa je samim tim i skupo.



Sa druge strane, **prosleđivanje** je proces slanja paketa na osnovu lokalne tabele. Rutiranje zapravo obezbeđuje tu tabelu koja govori na koji način ići do različitih destinacija. A prosleđivanje je korišćenje ove tabele. To znači da je to proces koji se izvršava na jednom čvoru, dakle lokalan je i brz. Zapravo mora da bude brz, jer ruter će možda morati da obradi milione paketa u sekundi, zato ga moramo brzo proslediti.



## Tipovi servisa na mrežnom sloju

Mrežni sloj pruža višem, transportnom sloju mrežni servis.

Postoje dva različita tipa mrežnih servisa.

Jedan tip se zove **Datagrami ili servis bez uspostavljanja veze**. Analogija je pošta. Datagrami su kao slanje pisama preko poštanskog sistema. Napišete pismo, zalepite ga, predate ga u poštu, svi se oni tretiraju individualno. Pošta ih isporiči pomoću adrese. To je veoma slično IP. IP je zasnovan na datagramima.

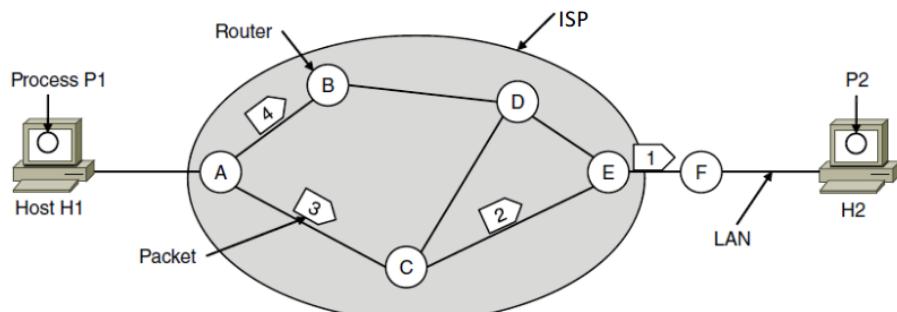
Drugi tip su **virtuelna kola**, odnosno **servis sa uspostavljanjem veze**. Analogija je fiksna telefonija. Morate napraviti neku vezu, kao što je pozvati nekoga preko telefona, pre nego što možete da koristite mrežu. Ali nakon što uspostavite vezu možete slati podatke kroz cev (pipe) i oni će stići do druge strane.

Svaki od ovih tipova ima svoje prednosti i mane. Oba servisa se realizuju tehnikom sačuvaj-i-prosledi. Usmerivači (ruter) dobijaju paket, i privremeno ga čuvaju sve dok ga ne proslede dalje (koriste baferisanje). Moguće je čuvanje paketa ako na primer ima više zainteresovanih za tu destinaciju. Važna osobina je da koristi tehniku statističkog multipleksiranja za upravljanje protokom.

FALI: objasnjenje statističkog multipleksiranja

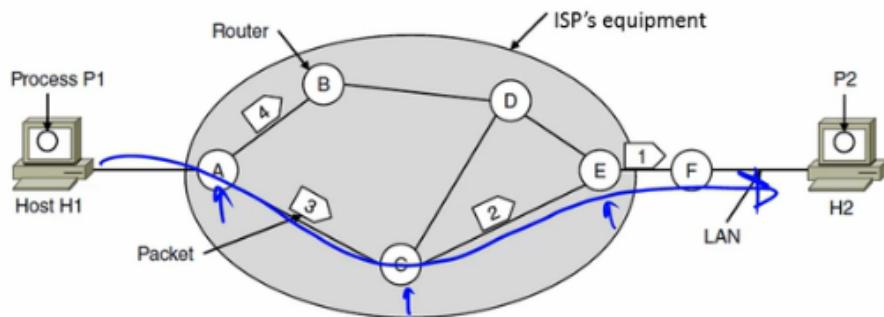
## Datagramski servis

Na slici je prikazana mreža i paket će prolaziti kroz različite hostove na njoj.

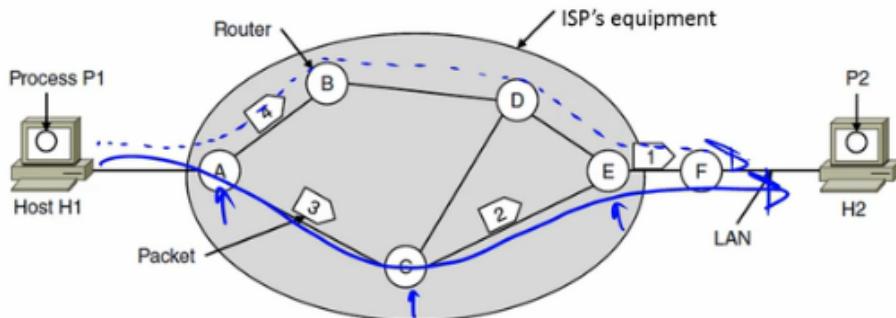


Svaki paket sadrži ciljnu adresu na osnovu koje usmerivač prosleđuje paket dalje. Te adrese koristi ruter da šalje pakete dalje.

Početni put od jednog hosta do drugog može biti sledeći:



Svi ruteri na putanji, A, C i E, kako paket stigne, pogledaju ciljnu adresu i ako vide da je to F pogledaće u tabeli i videti u kom smeru treba da proslede paket. Paket potom stuže u C i prolazi kroz isti proces. Međutim, pošto se svaki paket pojedinačno obrađuje moguće je da će se putanja promeniti dok neko koristi mrežu. Kasnije, A može da “promeni mišljenje” i da paketi prate drugačiju putanju kroz mrežu.



Krajnji host može da regularno primi pakete, ali može doći do čudnih zakašnjenja ili pogrešnog redosleda, zato što su paketi možda išli različitim putevima.

Svaki usmerivač ima tabelu prosleđivanja (forwarding table) – tabela se menja! Red u tabeli za datu ciljnu adresu određuje susedni čvor kojem će se paket proslediti (sledeći hop).

A (иницијално)		A (касније)		С табела		Е табела	
A		A		A	A	A	C
B	B	B	B	B	A	B	D
C	C	C	C	C		C	C
D	B	D	B	D	E	D	D
E	C	E	B	E	E	E	
F	C	F	B	F	E	F	F
Dest. Line							

Tabela A se menja kroz vreme.

Recimo da želimo da posaljemo paket od A do F. A pogleda u tabelu, vidi da mu je destinacija F, a sledeći hop mu je C. C na sličan način ponovi proces, vidi da je sledeći hop E, itd.

Kasnije tokom vremena, tabela A se može promeniti. Možete videti na slici da su se, iz nekog razloga, putevi ka destinacijama E i F promenili. Sad se paketi prosleđuju B umesto C. Zašto? Ne znamo zašto, a i nije ni bitno. Nemamo kontrolu nad tim, ne znamo kad se može desiti, ali se može

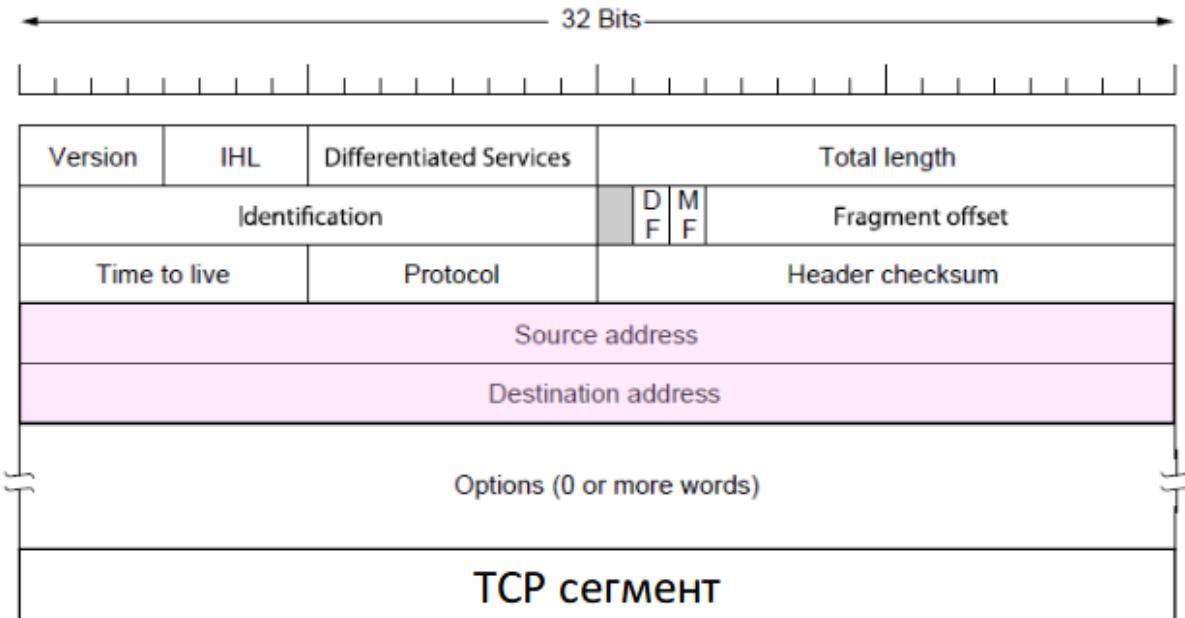
desiti. Kada paketi dođu u A, a destinacija im je F, ovog puta će biti poslati hostu B. B mora da pogleda u svoju tabelu i da odluči šta će dalje i sad pratimo drugi put kroz mrežu.

## IP (Internet Protocol)

Najpopularniji datagramski model je IP – Internet Protocol. IP je mrežni sloj interneta. Mrežni sloj koristi datagramski servis.

IPv4 paket ima 32-bitnu adresu i obično je velik oko 1.5 KB.

U svakom paketu se nalaze početna i krajna adresa. Sledeća slika prikazuje IPv4 paket:



TCP segment sadrži podatke koji se nalaze ispod zaglavlja.

## Virtuelno kolo

Drugi model koji smo spominjali je virtuelno kolo, koje je različito od datagramskog modela. U datagramskom modelu paketi su nezavisni, samostalni, sadrže potpune adrese, što nije slučaj u virtuelnom kolu. U virtuelnom kolu, baš kao kod fiksne telefonije, prolazi se kroz tri faze:

### 1. Uspostavljanje virtuelnog kola

Potrebljeno je uspostaviti neku vezu ili kolo između osobe koja želi da šalje do osobe koja želi da primi informacije. Uspostavljanje veze zapravo podrazumeva odabir putanje i slanje informacija o putanji svim relevantnim usmerivačima. Tako da oni znaju šta da rade kad naiđu na pakete iz ovog virtuelnog kola.

### 2. Prenos podataka

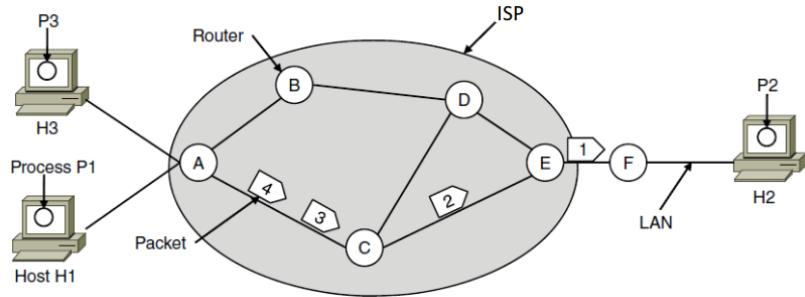
Nakon što je veza uspostavljena sledi prenos podataka. Paketi se prosleđuju duž uspostavljenog virtuelnog kola do destinacije.

### 3. Brisanje virtuelnog kola

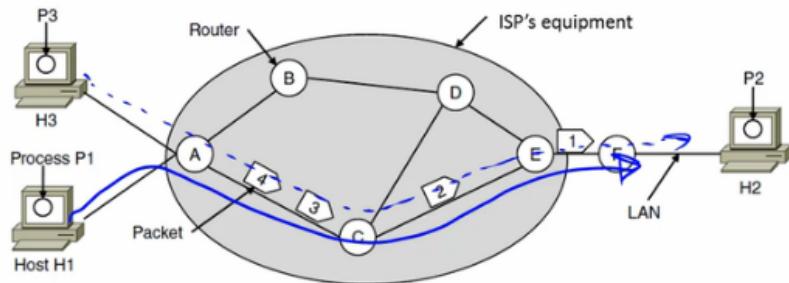
Nakon toga sledi brisanje virtuelnog kola u kome svi relevantni usmerivači brišu informacije o virtuelnom kolu.

Ovaj model je veoma sličan fiksnoj telefoniji, samo što je veza virtuelna. Virtuelna je u smislu da se kanal ne rezerviše samo za tu konekciju, već statičkim multipleksiranjem za veći broj virtuelnih kola i/ili datagramskih servisa. Koliki kanal se dobija za kolo zavisi od toga koliko saobraćaja se šalje i ostalog saobraćaja u mreži koji se takmiče za njega.

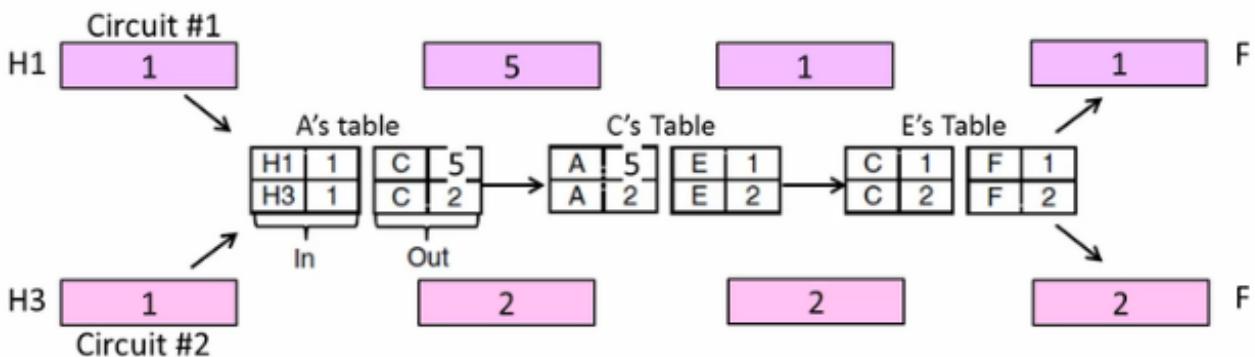
Za razliku od paketa u datagramskom modelu, paketi u modelu virtuelnog kola ne sadrže punu adresu. Sadrže samo kratki identifikator virtuelnog kola. Na taj način ruter zna šta da radi sa njim. U toku uspostavljanja veze je razrešeno šta virtuelno kolo sa određenim identifikatorom znači i kuda treba da se ide. Ovaj identifikator nema globalno značenje kao IP adresa, već samo na nivou nekog dela sloja veze. i mora da bude jedinstven, da bi ruter znao šta da radi.



U ovoj mreži moguće je da imamo dva različita kola koja idu istovremeno i oba ova kola mogu ići kroz neke iste čvorove. Oba ova kola se mogu koristiti istovremeno. Možda i host 1 i host 3 oba žele da pošalju nešto hostu 2.



Videli smo tabelu usmerivača u datagramskom modelu, pogledajmo sada tabelu usmerivača u modelu virtuelnog kola. Potrebna nam je tabela koja kao ulaze ima kola i koja treba da nam kaže šta da radimo sa njima. Na primer, paket koji stiže u A može doći ili od hosta 1 ili od hosta 3 i koji broj kola imaju. Na početku su oba obeležena brojem 1. Tabela izlaza nam kaže šta da radimo. Kolo koje dolazi od hosta 1 sa identifikatorom 1 se prosleđuje čvoru C kome je poznato kao kolo broj 5. To zapravo znači da se identifikator kola menja u paketu kako on putuje kroz mrežu. Paket prolazi vezom od A do C sa identifikatorom 5. Kad dođe u C, došao je od A sa identifikatorom 5, pa pogledamo u izlaznu tabelu i vidimo da treba da se pošalje čvoru E sa identifikatorom 1, 5 menjamo u 1 i prosleđujemo čvoru E. U E se prosleđuje čvoru F sa identifikatorom 1 itd.



Posmatrajmo sada saobraćaj od hosta 3 koji ide do F, pa do hosta 2. Ovo kolo je prolazilo kroz neke linkove koji su isti kao u prethodnom, zato ih moramo razdvojiti na dva različita kola na tim linkovima. Saobraćaj od hosta 3 stiže sa identifikatorom 1 u A. A ima drugačije ulaze za hostove 1 i 3 da bi znao koji je koji. Sledеća instrukcija je da pošalje paket u čvor C sa

identifikatorom 2. U C paket dolazi od A sa identifikatorom 2. Kao što vidimo, u čvoru C su oba paketa došla od čvora A, tako da je veoma bitno da imaju različite identifikatore virtuelnog kola da bismo mogli da ih razlikujemo. Zbog toga iz A nisu oba mogli da izađu sa identifikatorom 1 jer bi došlo do zabune. C prosleđuje paket E sa identifikatorom 2, a E prosleđuje F sa identifikatorom 2, itd. I dalje imaju različite brojeve tako da smo uspeli da razlikujemo ova kola.

## Datagrami ili Virtuelno kolo

**Priprema:** Za datagrame priprema nije neophodna. Napravite paket i ubacite ga u mrežu kad god poželite. Sa druge strane, za virtuelna kola je neophodno uspostaviti vezu.

**Adresiranje:** datagrami nose punu adresu, IPv4 adresu, od 32 bita, a u virtuelnim kolima paketi nose kratku oznaku.

**Usmeravanje:** Za svaki paket je potrebno odraditi posao, tako da je u datagramima usmeravanje po paketu, dok je za virtuelna kola posao odraden kad je veza uspostavljena, tako da nadalje samo šaljemo pakete kroz kolo.

**Otkazi:** U datagramima su lakši za razrešavanje jer ne skladištimo nikakve informacije o kolima u mreži i kako da rukujemo njima. Dok u virtuelnim kolima to radimo tako da ako ruter koji sadrži informaciju otkaže zaglavljeni smo. Potrebno je mnogo posla da se to popravi

**Različiti kvaliteti servisa:** U datagramima se svaki paket obrađuje nezavisno pa je teško dodati kvalitet servisa, jer on obično nije nešto što se primenjuje na pojedinačan paket. Obično se primenjuje na grupe paketa, kao npr. svi paketi u video konferenciji. U virtuelnim kolima je lako za dodavanje zato što imamo identifikator za sve pakete u kolu. Možemo npr. da imamo jedno kolo za video konferenciju.

Карактеристика	Датаграми	Виртуелна кола
Припрема	Није неопходна	Неопходно
Адресирање	Пакети носе пуне адресе	Пакети носе кратку ознаку
Усмеравање	По пакету	По колу
Откази	Лакше за разрешавање	Тешко за разрешавање
Различити квалитети сервиса	Тешко за додавање	Лако за додавање

## 27. IP adrese i prefiksi

### Povezivanje različitih mreža (međumreže)

Kako kombinovati više mreža u jednu veću mrežu? Povezivanje različitih mreža u jednu veću je teško jer možda želimo da povežemo mreže koje se veoma razlikuju po tome kako interno funkcionišu. Mogu biti različite na više načina. Mi zapravo želimo da povežemo hosta na jednoj mreži sa hostom na drugoj mreži kao da je u pitanju jedna mreža. To je teško i ne uspeva uvek.

Razlike među mrežama

Mogu se razlikovati u mnogo aspekata:

- **Tip servisa** (datagrami, virtuelna kola)

Recimo da jedna mreža koristi datagrame, a druga virtuelna kola. Sad ih je potrebno iskombinovati. To je kao kombinovanje poštanskog sistema I fiksne telefonije. Ne deluje kao da to može da funkcioniše baš najbolje.

- **Adresiranje**

Različite mreže mogu da imaju različit tip adresiranja, zato što će ih, u normalnim okolnostima, dizajnirati različiti ljudi.

- **Kvalitet usluge – QOS (Quality of Service)**

Recimo da jedna od mreža ima više različitih vrsta kvaliteta usluge. Možda ima regularne pakete, pakete koji imaju prioritet, koje bi trebalo obrađivati pre regularnih paketa. A druga mreža to nema, ona ima samo klasičnu uslugu. Kako ćemo da kombinujemo te mreže a da opet ispoštujemo dogovore vezane za kvalitet usluge jedne mreže? To nije baš najjasnije.

- **Veličine paketa**

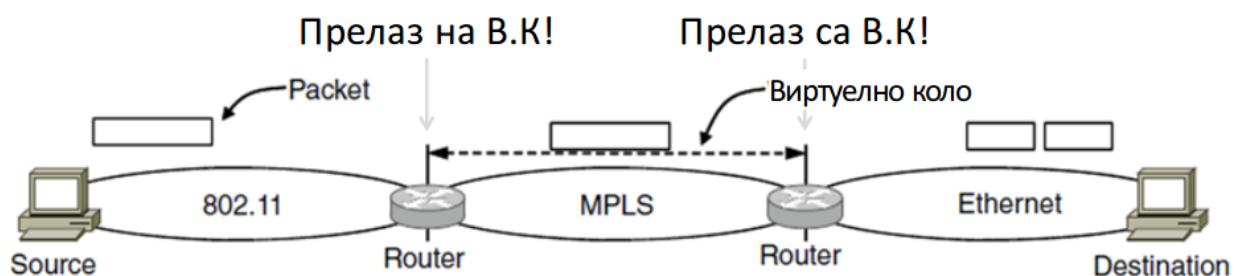
Različite mreže mogu obrađivati pakete različitih veličina. Ovo je svakodnevni slučaj zbog tehnoloških razloga – različite vrste tehnologija na sloju veze mogu da obrađuju pakete različitih maksimalnih veličina.

- **Sigurnost (ima/nema enkripcije)**

Posao povezivanja različitih mreža je da sakrije te razlike koristeći zajednički protokol.

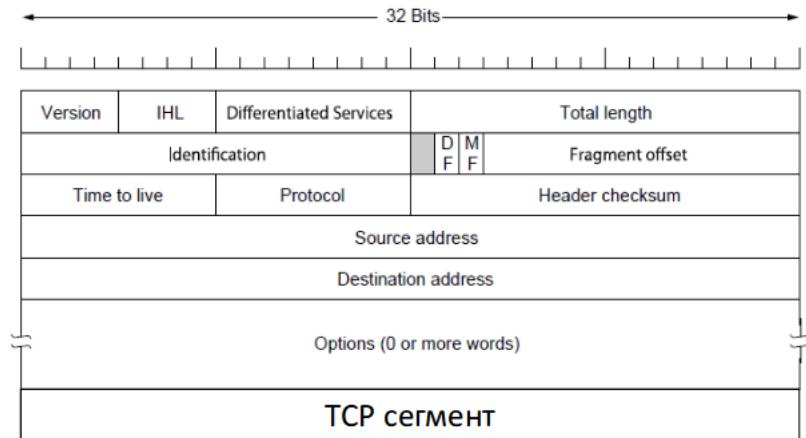
### Povezivanje datagramskog servisa i servisa virtuelnog kola

Prepostavimo da šaljemo podatke od izvorišta ka odredištu. Izvorište je u datagramskoj mreži i odredište je isto u datagramskoj mreži. A u sredini se nalazi mreža virtuelnog kola. Imamo paket sa leve strane i prvo želimo da napišemo adresu odredišta u paket, iako formati mogu biti različiti – jedan je na 802.11 a drugi na Ethernet-u. U datagramskoj mreži paket će se proslediti koristeći adresu odredišta kroz svaki ruter sve dok ne dođe do mreže virtuelnog kola. Ovaj individualni datagram je potrebno preslikati u odgovarajuće kolo, a potom mu je potrebna neka druga identifikaciona informacija koju ranije nije imao, odnosno identifikacija kola, da bi posle mogao da se šalje kroz to kolo. Sa druge strane vraćamo se na datagramsku mrežu tako da možemo da zaboravimo od kog virtuelnog kola je došao. Ovo kolo mora biti postavljeno, da nije bilo postavljeno pre vremena, morali bismo da zadržimo paket neko vreme dok se kolo ne postavi. To može biti problem jer paketi mogu pristići relativno brzo. Potrebno je preslikavanje između IP adresa i oznaka virtuelnog kola. ISP često koriste virtuelna kola kako bi grupisali IP saobraćaj i brže ga preneli.



### IPv4 (Internet protokol)

Ova slika nam pokazuje šta se događa u svim IP paketima I fokusira se na zaglavje paketa. Deo označen kao "TCP segment" (ako je viši sloj TCP) su podaci koje paket prenosi. Dijagram nam pokazuje kojim redosledom su informacije postavljenje u paket. Čita se sleva nadesno I odozgo nadole I odgovara jednoj velikoj liniji. Ovaj dijagram je dužine 32 bita.



Paketi sadrže:

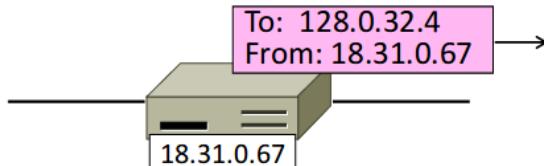
- **Verziju** (Version)  
Broj verzije. Ovde bi trebalo da stoji 4, jer je IPv4 u pitanju,
- **Zaglavlje** (Internet Header Link)  
Govori koliko je dugačko zaglavlje. Posle toga počinju podaci.
- **Dužina paketa** (Total length)  
Govori o tome koliko je dugačak celokupan datagram
- **Kontrolni zbirovi** (Header checksum)  
Za proveru pouzdanosti zaglavlja.
- **Protokol** (Protocol)  
Govori nam koji protokol višeg nivoa u IP-u, na primer, TCP, ako paket prenosi TCP segment. Ovaj protokol je ključ za demultiplexiranje koji omogućava da se sadržaj paketa isporuči ispravnom višem sloju.
- **IP adrese (nije isto I na sloju veze)**  
Adresa izvorišta I adresa odredišta. S obzirom na to da IP koristi datagrame, svaki paket koji je datagram mora da uključuje punu adresu, da bi ruteri znali gde da ga prosledi. Nije isto adresiranje kao na sloju veze, kao što je ethernet adresa, ovo je adresa mrežnog sloja. U IPv4 adrese su dužine 32 bita.
- **Differentiated services**  
Odnosi se na kvalitet usluge.
- **Time to live**  
Koristi se u konjukciji sa protokolom za upravljenje porukama na internetu (ICMP).

U delovima Total length, Identification I Fragment offset se rešava problem sa slanjem paketa različitih veličina. Zahvaljujući tim poljima veći paket se može rastaviti na manje I slati nezavisno, da bi se na kraju sklopio u jedan.

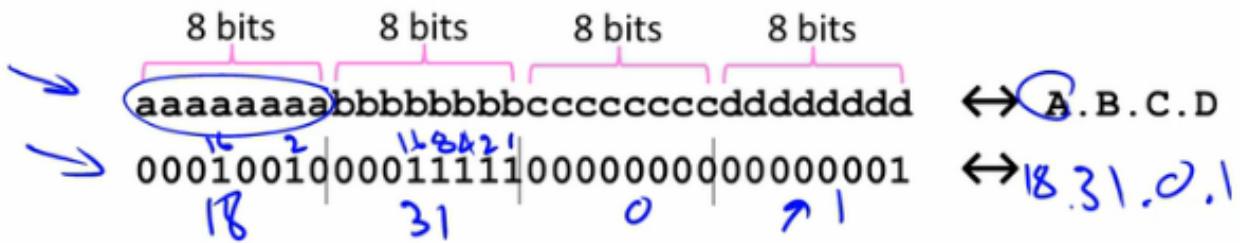
## IP adrese

Kako izgledaju IP adrese?

Ovo se odnosi na IPv4.



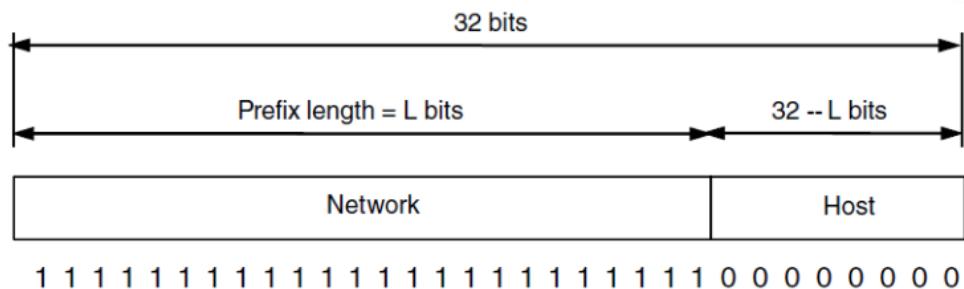
IPv4 koristi 32-bitne adrese. IPv6 koristi 128-bitnu adresu. Razlog zašto su uvedene IPv6 adrese je to što su IPv4 skoro potpuno iskorišćene. Adrese su zapisane u vidu četiri 8-bitna broja odvojena tačkama (decimalna notacija sa tačkom ili notacija u vidu kvarteta dekadnih brojeva). Možemo adrese zapisati binarno, ali takav način zapisivanje je zamoran, pa ćemo ih zapisati na drugačiji način. Grupe od 8 bitova ćemo zapisati njihovim decimalnim ekvivalentom i odvojiti tačkama. Dakle, zapisaćemo ih kao četiri decimalna broja u rasponu od 0 do 255. Tako npr. binarni broj na slici možemo zapisati kao 18.31.0.1.



## IP prefiksi

Adrese se grupišu u blokove koji se nazivaju prefiksi. Dakle, prefiks je blok adresa, ali to je blok koji strukturiran na poseban način.

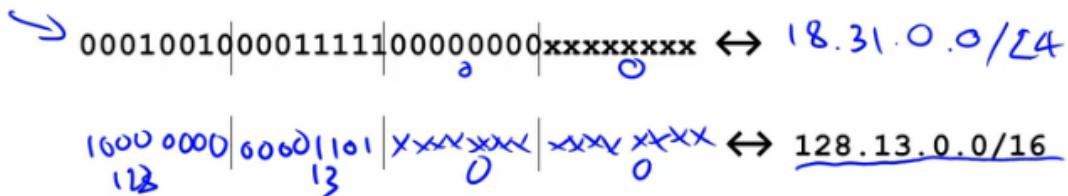
L-bitni prefiks je grupa adresa koje imaju isti prefiks dužine L bita.



L može imati vrednosti od 0 do 32. Ali uobičajeno ima vrednost 16, ili vrednosti između 8 I 24. Ako imamo prefiks dužine L, to znači da će prvih L bitova u adresama imati neku fiksnu vrednost, a ostalih 32-L bitova mogu uzeti bilo koju drugu binarnu vrednost. To znači da L-bitni prefiks ima  $2^{32-L}$  različitih adresa. Na primer, 24-bitni prefiks ima  $2^8$  ili 256 različitih adresa. (Te adrese su poravnate na  $2^{32-L}$  granici).

Potreбна нам је notacija за zapisivanje IP prefiksa. Koristimo notaciju oblika "IP adresa/dužina prefiksa". Adresa će biti najmanja adresa u prefiksu u notaciji sa tačkama, a dužina prefiksa je upravo dužina prefiksa u bitovima. Na primer, 128.13.0.0/16 znači da ima 16-bitni prefiks to znači da je poslednjih  $32-16 = 16$  bitova slobodno, a adrese će biti u rasponu od najmanje (koja je data), a to je 128.13.0.0 do 128.13.255.255 (koja je dobijena okretanjem poslednjih 16 bitova). Dakle, prefiks oblika /24 odgovara opsegu sa 256 adresama, dok /32 odgovara jedinstvenoj adresi.

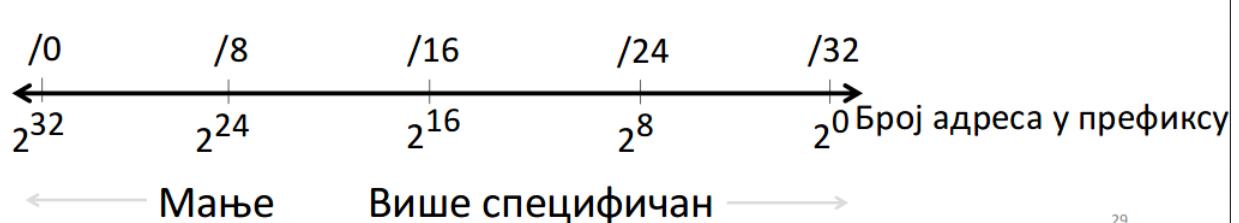
Na slici imamo primer, kako u notaciji zapisatu datu adresu i kako zapisati adresu koja je data u notaciji.



IP adresa pripada različitim prefiksima. Možemo govoriti o više specifičnim i manje specifičnim prefiksima.

Više specifičan prefiks je duži prefiks, ima manji broj IP adresa, dakle bliže određuje opseg oko te adrese.

Manje specifičan prefiks je kraći prefiks, ima veći broj IP adresa.



29

Na dijagramu se može videti, kako idemo ka manje specifičnim prefiksima, prefiksi se smanjuju, ali se broj adresa povećava. Slično, kako idemo ka više specifičnim adresama, prefiksi se povećavaju, ali se broj adresa smanjuje.

## IP klase adresa – stari sistem grupisanja

Danas koristimo IP prefikse, ali ranije su IP adrese bile grupisane u klase adresa fiksne dužine. Tako da je dužina prefiksa bila fiksirana i postavljena kao deo adrese. Adrese klase A su koristile 8 bitova za deo koji opisuje mrežu, a poslednja 24 bita su bila slobodna. Adrese klase A su bile identifikovane početnom nulom, nakon koje je sledilo 7 bitova koji su mogli da se postave na bilo koju vrednost za različite adrese klase A. Dakle, klasa A odgovara /8 adresama, jer su prvih 8 bitova fiksirani. Na sličan način vidimo da su klase B i C ekvivalentni /16 odnosno /24 prefiksima. Mada su početni bitovi ograničeni na 10 i 110, tako da možemo da identifikujemo klase A, B i C gledajući samo u početne bitove.



Danas adrese i dalje imaju početne bitove pomoću kojih se može odrediti da li pripadaju klasi A, B ili C, ali se same klase ignorisu. Prefiksi su zapravo generalizacija ovih klasa. Umesto da imamo samo prefikse dužina /8, /16, /24, možemo imati i prefikse između. Dakle, početni bitovi su ostali isti zbog kompatibilnosti, ali se ignorisu.

## Javne / privatne IP adrese

Postoji još jedno važno razlikovanje adresa, a to je između javnih i privatnih adresa.

Javna adresa je npr. 18.31.0.1.

Javna IP adresa je bilo koja adresa koja je validna adresa destinacije i može da se koristi globalno na internetu i mogu se slati paketi ka toj adresi i od te adrese. Pre upotrebe se mora dodeliti, problem sa tim je što su većim delom potrošene, ponestalo ih je, zato nam je potreban IPv6.

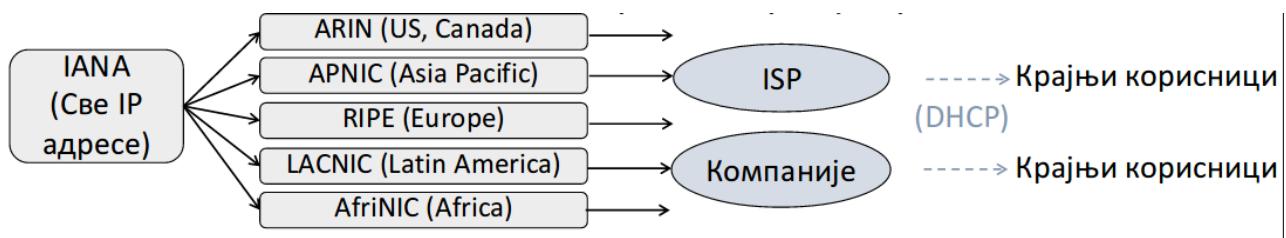
Postoje i privatne IP adrese. To su IP adrese koje se slobodno mogu koristiti, ne mora niko da nam ih dodeli, nisu globalno jedinstvene. Jesu jedinstvene na nivou manjih mreža, npr. U mreži firme, kućne lokalne mreže i slično.

Na primer, 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16

Ali čak i privatnu mrežu ćeleti da povežete na javni internet. Za to vam je potrebna javna IP adresa, koji dobijate od ISP-a i tehnologija NAT (Network Address Translation) da bi se iz ovakvih mreža povezali na internet, tj. da biste preveli privatne adrese koje su validne samo u okviru privatnih mreža u javne koje se mogu koristiti na Internetu.

## Dodeljivanje javnih IP adresa

Dodeljivanje javnih IP adresa je hijerarhijski proces. Na početku sve adrese pripadaju svetskom regulatornom telu pod imenom IANA. Dodeljuje ceo opseg adresa regionalnim telima – RIR (Regional Internet Registries) koja imaju različita imena. Regionalna tala potom delegiraju IP adrese kompanijama u tom regionu, kompanije apliciraju za te adrese i dobijaju ih. Napokon, kad kompanija dobije IP adresu, te adrese dodeli računarima u toj mreži. Ili IP adrese dobiju krajnji korisnici od ISP-a. To se obavlja preko protokola DHCP.



## 28. IP prosleđivanje

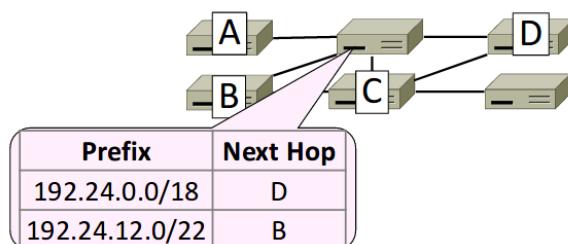
Podsetimo se prvo razlike između rutiranja i prosleđivanja.

**Rutiranje** je proces odlučivanja u kom pravcu treba poslati saobraćaj.

**Prosleđivanje** je proces slanja paketa na osnovu lokalne tabele. Rutiranje zapravo obezbeđuje tu tabelu koja govori na koji način ići do različitih destinacija. A prosleđivanje je korišćenje ove tabele.

Ranije smo spomenuli da je jedan od problema na sloju veze bilo skaliranje na velike mreže. Taj problem se rešava korišćenjem adresa sa hijerarhijama. Hajde da vidimo kako se skalira na veće adrese sa IP prosleđivanjem.

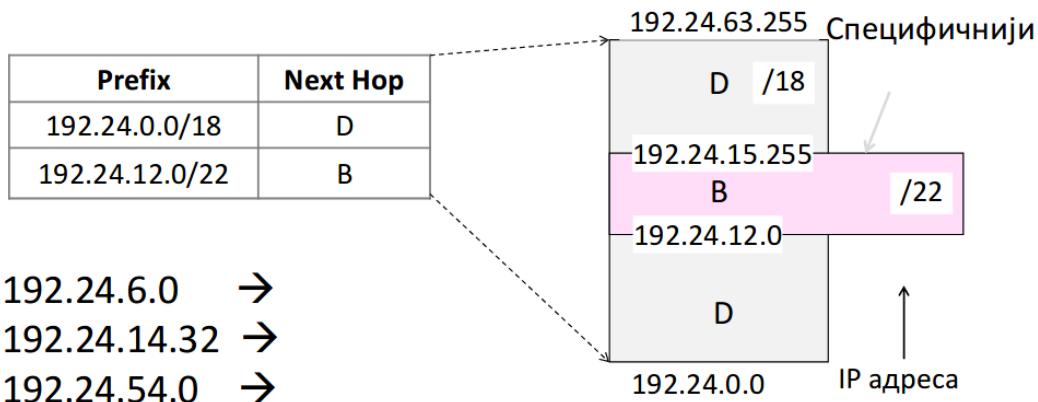
Koristimo IP prefikse da bismo obezbedili skaliranje. Sve IP adrese jedne mreže pripadaju istom prefiksnu. Svaki usmerivač poseduje tabelu uređenih parova **oblika** (prefiks, sledeći čvor – hop). Na taj način ako naiđemo na neke pakete koji su namenjeni za adrese u okviru tog prefiksa, poslaćemo ih na isto mesto, koje god je određeno za taj prefiks. Sledeća slika predstavlja primer toga:



Ovde je prikazana tabela prosleđivanja za ruter i ima samo par ulaza, ali su to prefiksni ulazi. Prvi prefiks je /18, a drugi je /22. Imaju drugačije vrednosti u koloni Next Hop, dakle drugačije postupama sa adresama u tim prefiksima. Oba ova prefiksa obuhvataju mnogo adresa i mnogo destinacija. Bez obzira na veliki broj adresa u tabeli, tabela je relativno mala i (malo se skalira?).

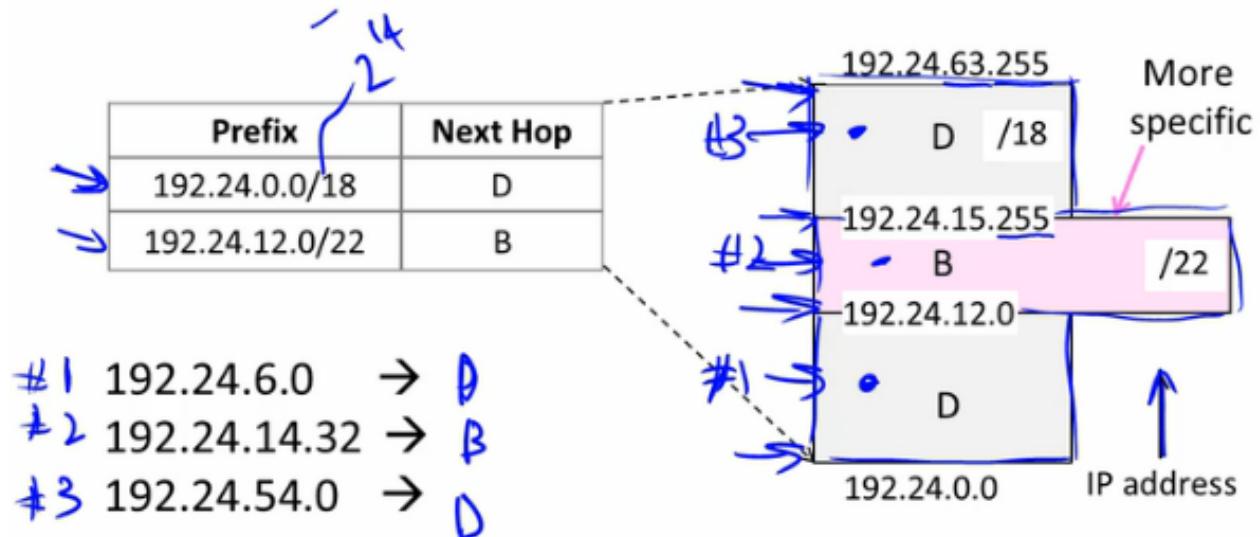
## Najduži odgovarajući prefiks

Prefiski u tabeli se mogu preklapati! Možemo imati prefikse različitih dužina i oni se mogu preklapati. Da bismo rešili ovaj problem, koristimo pravilo prosleđivanja koje se nazina Pravilo najdužeg odgovarajućeg prefiksa. Ono glasi: Za svaki paket, pronaći najduži prefiks koji sadrži ciljnu adresu, odnosno najspecifičniji prefiks. I proslediti paket čvoru koji odgovara tom prefiksu. Pogledajmo to na primeru:



Dakle, prvi prefiks u tabeli kreće od adrese 192.24.0.0, to je najniža adresa na dijagramu koji se nalazi desno na slici. Sledеći prefiks u tabeli kreće od adrese 192.24.12.0, tako da se nalazi iznad. Nemamo gornje adrese pa je potrebno da ih izračunamo. Za prefiks /22, koji fiksira prva 22 bita, poslednjih 10 bitova je slobodno. Računamo gornju adresu tako što uzimamo adresu 192.24.12.0 i okrećemo poslednjih 10 bitova. Okrećemo poslednjih 8 bitova što nam daje 255 kao poslednji broj u notaciji sa tačkama. I okrećemo poslednja 2 bita u sledećem, što je ekvivalent sabiranju sa tri, dakle sa 12 prelazimo na 15 i dobijamo kao gornju adresu 192.24.15.255. Primetimo da je /22 prefiks obojen kao roze dijagram i duži je zato što je specifičniji.

Za drugi prefiks – 192.24.0.0/18 – njemu je poslednjih 14 bitova slobodno. Dakle okrećemo poslednjih 14 bitova. Okrećemo poslednjih 8, dobijamo 255, onda je potrebno okrenuti još 6, čime se dobija 63. Dakle, gornja adresa je 192.24.63.255. Prvom prefiksu u tabeli odgovara sivi blok, koji je mnogo veći i nije širok koliko i roze zato što je manje specifičan.



Pogledajmo sad na primeru kako to radi. Imamo nekoliko adresa, #1, #2, #3. Prva adresa – 192.24.6.0 – se nalazi u bloku D ispod bloka B, zato što je 6.0 manje od 12.0. Dakle, prva adresa se prosleđuje čvoru D. Treća adresa – 192.24.54.0 – se nalazi u bloku D iznad bloka B, zato što je 54.0

veće od 15.255, a menja od 63.255. Dakle, treća adresa se takođe prosleđuje čvoru D. Druga adresa – 192.24.14.32 – sadrži 14.32. koji se nalazi negde između donje i gornje granice za B. Dakle, druga adresa se prosleđuje čvoru B. I ovo je algoritam najdužeg odgovarajućeg prefiksa.

## Fleksibilnost Najdužeg odgovarajućeg prefiksa

Možemo da koristimo najduže odgovarajuće prefikse na par različitih načina.

- **Mogu da pruže podrazumevano ponašanje, sa manje specifičnim prefiksom**

Na primer, imate mrežu kompanije i u ruteri u njoj znaju kako da obrađuju IP adrese koje su u njoj, ali za druge IP adrese koje su van možemo da imamo podrazumevano pravilo koje kaže - pošalji paket na moj ISP ruter. I naš ruter može jednostavno da koristi to podrazumevano pravilo da pošalje pakete van mreže.

- **Mogu da pruže specijalno ponašanje, sa specifičnijim prefiksom**

Na primer, možete uzeti neku specifičnu adresu i neki specifičniji prefiks tako da je ruter posebno obrađuje i možda pošalje na drugačiji način nego što bi to u normalnom slučaju. Ovo se uglavnom radi zbog performansi ili sigurnosti...

Poenta je da dobijamo određeni stepen fleksibilnosti od pravila najdužeg odgovarajućeg prefiksa i to možemo da iskoristimo u našu korist.

## Performanse Najdužeg odgovarajućeg prefiksa

Koliko je dobar ovaj algoritam u smislu performansi?

U slučaju veličine tabele, performanse su veoma dobre. Koristimo hijerarhiju da dobijemo dobru, kompaktnu tabelu. Dakle, ogromna tabela je rasparčana na veliki broj usmerivača. To se najviše ostvaruje ako se koriste manje specifični prefiksi, odnosno prefiksi koji sadrže veliki broj adresa. Ako svi koriste veoma specifične prefikse, nećemo dobiti mnogo kompaktnosti. Kompaktna tabela omogućava podrazumevanje (manje efikasno, sigurno, ...) ponašanje. Dok veće tabele omogućavaju specifičnije (efikasnije, sigurnije, ...) ponašanje.

Ovo je kompromis između vremenske i prostorne složenosti koji stalno srećemo u računarstvu!

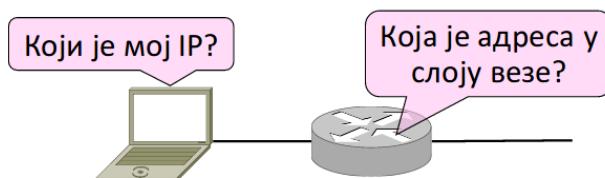
Drugi aspekt performansi je koliko brzo se izvršava operacija pretraživanja u ruterima. Ispostavlja se da je najduži odgovarajući prefiks složeniji od običnog pretraživanja tabele. Ovo je predstavljalo problem kada smo ranije želeli da napravimo ruter da budu brzi, danas to nije toliki problem.

## 29. ARP i DHCP

ARP I DHCP su dva ključna pomoćna protokola koja rade sa IP.

Nailazimo na dva problema:

1. Dodeljivanje IP adrese računaru u mreži (DHCP)
2. Određivanje adrese u sloju veze (MAC) za ciljnu IP adresu (ARP)



ARP i DHCP su neophodni za IP, da bi mogao da funkcioniše u praksi. DHCP pruža malo IT podrške, dok ARP pruža "lepak" da bi spojio mrežni sloj sa slojem veze.

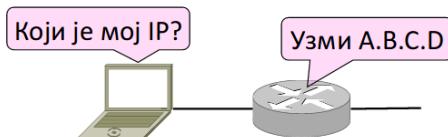
## Dodeljivanje IP adrese

Prvi problem je dobijanje IP adrese. Recimo da se čvor A upravo uključuje, pokreće se po prvi put i ne zna mnogo toga. Pre svega, želi da zna koja je njegova IP adresa, koja je IP adresa novog rutera, na kojoj je mreži, itd...



Jednu stvar koju zna je njegova Ethernet adresa. Razlog za to je što je Ethernet adresa fabrički zadata na mrežnoj kartici. Tako da kad se čvor pokrene, ako je zakačen na Ethernet obično zna svoju Ethernet adresu, ali ne zna svoju IP adresu. Taj problem se može rešiti na par načina:

1. Ručno podešavanje (stari pristup)
2. Protokol za automatsko podešavanje adresa (DHCP – Dynamic Host Configuration Protocol)



Postavlja se pitanje – zašto je ovo neophodno? Za Ethernet jednostavno imamo adresu na mrežnoj kartici. Za IP, međutim, adresa koju imaš zavisi od toga gde se u mreži nalaziš. Da bi se moglo efikasno prosleđivati, zahtevamo da svi čvorovi u jednoj mreži pripadaju istom prefiksnu. Tako da gde se računar nalazi će uticati na IP adresu, zato ona ne može biti zadata fabrički.

Druga alternativa, veoma popularna danas, je da se osmisli protokol za automatsko podešavanje IP adresa novih čvorova kada se pokrenu. Protokol je DHCP (Dynamic Host Configuration Protocol).

## DHCP

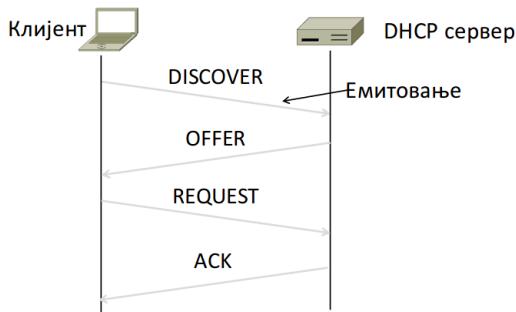
Glavna funkcija mu je da računaru obezbedi IP adresu. Zapravo, adresu ne daje trajno, već je iznajmljuje. Adrese pripadaju mreži tako da ih DHCP server može dati drugim čvorovima tokom vremena. DHCP obezbeđuje i neke druge parametre. Na primer, IP adresu lokalnog rutera, prefiks mreže, da bi znao da li šalje hostu na lokalnoj mreži ili udaljenom hostu, DNS server (prevodi adrese kao što su [www.google.com](http://www.google.com) u IP adresu), time server, itd...

DHCP zapravo radi kao klijent server aplikacija između klijenta koji je zapravo naša mašina i servera koji radi negde na mreži.

Kako DHCP radi?

Ako se čvor pokrenuo i pokušava da kontaktira nekog da bi našao svoju adresu, kako on zna IP adresu onoga od koga treba da traži tu adresu? Tek se pokrenuo i nije konfigurisan. Odgovor na to je da kada se računar pokrene, on ne zna IP adresu DHCP servera. Umesto toga, čvor emituje paket celoj mreži (specijalna adresa je 255.255.255.255, odnosno sve jedinice). Mreža dostavi taj paket svakom hostu na mreži, jedan od njih će biti DHCP server, on će uvideti da je paket namenjen za njega i obradiće ga. Emitovanje je glavna komponenta DHCP-a.

Kako izgled razmena?



Opis protokola:

- Čvor emituje paket celoj mreži (specijalna adresa za emitovanje je 255.255.255.255)**  
Ta poruka se zove "Discover"
- DHCP odgovara čvoru ciljano na osnovu njegove MAC adresе sa predloženom IP adresom**  
Ta poruka se zove "Offer"
- Čvor emituje odgovor da mu odgovara predložena IP adresа (može da bude više DHCP-ova)**  
Ta poruka se zove "Request"
- DHCP potvrđuje I briše adresу iz spiska slobodnih adresа**  
Ta poruka se zove "ACK" (Acknowledgement)

Akronim za ovo je DORA.

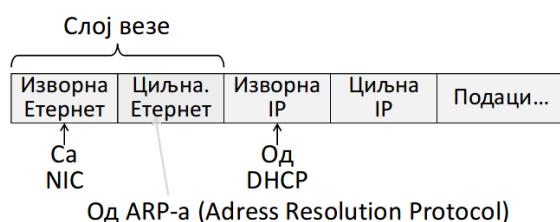
To je bila razmena kad se prvi put dobija IP adresu. Klijent može i da obnovi već dodeljenu IP adresu ako je ranije dobio. Rekli smo da DHCP iznajmljuje IP adrese na određeni vremenski period. Kad taj vremenski period istekne potrebno je nabaviti drugu IP adresu. Obično želimo da zadržimo istu adresu. Protokol u tom slučaju izgleda tako da čvor pošalje REQUEST i DHCP odgovori sa ACK.

Protokol takođe omogućava paralelni rad više repliciranih DHCP servera. U tom slučaju nam posebno dolazi do značaja emitovanje poruke. Na taj način svi DHCP serveri mogu videti poruke koje klijenti šalju. Uglavnom se koristi zarad pouzdanosti i efikasnosti. REQUEST se emituje tako da su sinhronizovani.

## Slanje IP paketa

Drugi problem:

- Čvor mora da sazna ciljnu adresu u sloju veze kako bi poslao okvire na odgovarajući čvor
- Kako na osnovu ciljne IP adrese da sazna adresu u sloju veze?



Dakle, imamo IP adresu i klijent zna da paket šalje ruteru sa određenom IP adresom, ali svejedno mora da sastavi paket, koji u sebi sadrži i adresu u sloju veze, a ne zna koju adresu da koristi.

Na slici iznad imamo dijagram – imamo čvor koji pokušava da pošalje paket drugom čvoru na istoj mreži. Da bi to postigao mora da sastavi paket. Na slici vidimo paket koji se sastoji od podataka,

ispred kojih se nalazi IP zaglavlje koje sadrži izvornu i ciljnu adresu, a ispred njega imamo zaglavlje sloja veze. Ta zaglavlja u sebi sadrže adrese. Čvor ima ciljnu IP adresu, to je poznato, jer mora znati tu adresu ako namerava da pošalje paket na nju. Izvorna IP adresa je sama adresa čvora koju dobija od DHCP-a. Sada je potrebno razrešiti adrese sloja veze. Izvorna Ethernet adresa se nalazi na mrežnoj kartici, tako da je lako dobijamo. Ali potrebna nam je i ciljna Ethernet adresa. Nju dobijamo koristeći ARP (Address Resolution Protocol) protokol.

## ARP

ARP se nalazi na vrhu sloja veze. Za razliku od DHCP, u slučaju ARP-a nemamo server. Pošto nam ARP pomaže da pošaljemo paket čvoru na lokalnoj mreži, ispitaćemo sve čvorove na lokalnoj mreži koima odgovarajuću IP adresu da nam kaže koja je njegova Ethernet adresa.

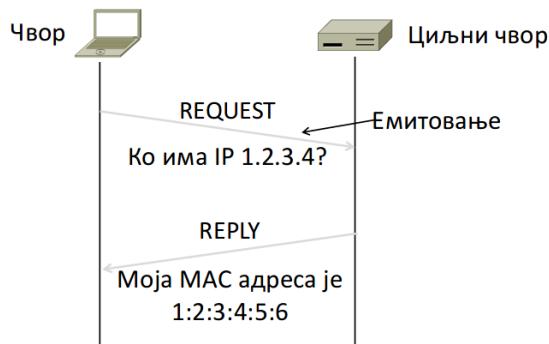
Opis protokola:

**1. Čvor koji hoće da sazna emituje ciljnu IP adresu**

Šalje "REQUEST" koji se emituje kroz celu mrežu, svaki čvor će primiti tu poruku, on traži Ethernet adresu koja odgovara IP adresu koju on šalje.

**2. Onaj koji ima tu adresu za svoju izvornu, vraća mu odgovor sa svojom adresom u sloju veze**

Čvor sa odgovarajućom IP adresom šalje svoju Ethernet adresu nazad u vidu poruke "REPLY".

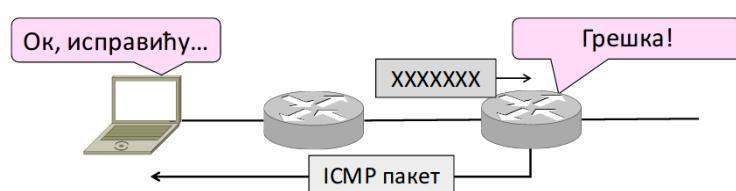


Nakon toga čvor koji šalje može da popuni paket i da ga pošalje.

## 30. ICMP I NAT

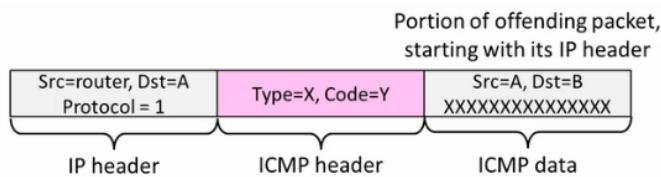
### ICMP

Šta se radi kada se desi greška prilikom prosleđivanja? Potrebno je da se to nekako javi čvoru koji je poslao paket. IP rešava te probleme protokolom koji se zove ICMP (Internet Control Message Protocol). U suštini, IP i ICMP su uvek implementirani zajedno. ICMP poruke se zapravo prenose unutar IP paketa. Ako pogledate IP paket i u delu Protocol taj broj je postavljen na 1, onda IP paket prenosi ICMP poruku. ICMP pruža više funkcionalnosti, najveći deo te funkcionalnosti se odnosi na izveštavanje o grešci. Pod greškom se smatra kad se nađe na problem pri prosleđivanju paketa u ruteru. I potrebno je tu grešku prijaviti onome ko je poslao paket da bi mogao da preduzme nešto po tom pitanju.



Pogledajmo na slici kako to funkcioniše. Prvi korak je da neko pošalje paket, u ovom slučaju je to host (izvor) sa leve strane. Drugi korak je da se u nekom ruteru desi da on ima problem da prosledi taj paket dalje, dakle on detektuje grešku pri prosleđivanju, kao što je npr. preveliki paket, maksimalan broj hopova i slično. Treći korak je da taj ruter pošalje ICMP paket pošiljaocu, a onda da odbaci taj paket jer ne zna šta treba da radi sa njim. Četvrti korak je da izvor dobije tu ICMP poruku, razume gde je došlo do problema i da preduzme neke korake da reši taj problem.

Svaki ICMP paket ima ICMP zaglavje koje sadrži tip greške, kod i kontrolni zbir. Takođe sadrži i početak paketa u kome se nalazi greška, sadrži onoliko tog paketa koliko može da stane. Na taj način vraća taj paket pošiljaocu tako da on može da utvrdi gde je došlo do greške. ICMP paket je isti kao IP paket, ima samo indikatorsko polje koje omogućava razlikovanje. Pošto ICMP paket u sebi sadrži ICMP data (podatke), a to je početak problematičnog paketa, znači i zaglavje tog paketa je tu a i izvorna adresa, tako da će ruter znati kome da vrati paket. Zahvaljujući tome znamo da napravimo IP zaglavje nove poruke kojoj je ciljna adresa zapravo izvorna adresa stare poruke. Izvor nove poruke je upravo ruter koji je naišao na grešku, on stavlja svoju IP adresu. I protokol je postavljen na 1 da bi se znalo da je u pitanju ICMP poruka.



Na sledećoj slici vidimo primer ICMP poruka:

Назив	Тип/ Код	Употреба
Dest. Unreachable (Net or Host)	3 / 0 or 1	Недоступност циља
Dest. Unreachable (Fragment)	3 / 4	Пакет превелик
Time Exceeded (Transit)	11 / 0	Traceroute
Echo Request or Reply	8 or 0 / 0	Ping (тестирање циља)

**Destination Unreachable (Network/Host)** – ICMP poruka sa ovim nazivom se dobija kada smo pokušali da posaljemo paket na destinaciju i mreža nije uspela da odredi gde da ga posalje. Biće tipa 3 i imaće različite kodove u zavisnosti od toga što je nedostižno.

**Destination Unreachable (Fragment)** – U ovom slučaju destinacija je nedostizna zbog fragmentacije. Mreži je bilo potrebno da rasparča paket, ali nije mogla pa se vraća ova greška.

**Time Exceeded (Transit)** – Označava da je paket u mreži previše dugo, a još nije stigao do svoje destinacije.

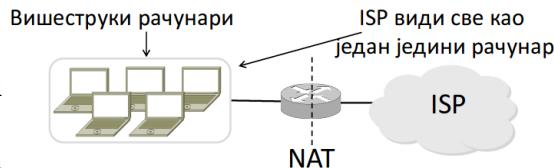
**Echo Request or Reply** – Koristi se za program Ping, koji se koristi da se proveri da li je host živ i dostižan na mreži. Ova poslednja kategorija je drugačija od ostalih zato što nije u pitanju greška koju je generisao ruter pri prosleđivanju. Echo request je nešto što host šalje drugom hostu, a ciljni host zna da je u pitanju echo request i onda šalje echo reply.

## NAT

NAT (Network Address Translation) povezuje računare iz lokalne mreže na spoljnu mrežu npr. Internet. NAT zapravo prevodi adrese tako da povezuje unutarnju mrežu sa spoljašnjom mrežom. IPv4 omogućava samo par milijardi dostupnih javnih adresa. Međutim, računara koji se povezuju na Internet je mnogo više. NAT je motivisan upravo nestasicom IPv4 adresa.

Standardni scenario:

- Kućni računari koriste “private” IP adrese
- NAT (u okviru kućnog usmerivača – AP) povezuje više kućnih računara na jednu javnu adresu dodeljenu od strane ISP



Dakle, recimo da u kući imamo više računara. Računari su povezani na internet preko ISP. ISP će

nam dodeliti samo jednu javnu IP adresu. Računari se nalaze u kućnoj mreži i svakom od tih računara se može dodeliti privatna IP adresa, postoje rezervisani prefiksi za privatne adrese, primer dve takve je 192.168.0.0/16 i 10.0.0.0/8. To su privatne adrese i mogu se koristiti u bilo čijoj privatnoj mreži, samo se ne mogu povezati na neku javnu mrežu jer nemaju jedinstveno značenje. NAT će onda prevesti ove privatne adrese u jednu jedinstvenu javnu IP adresu u ISP. Računarima u kući izgleda kao da svaki od njih ima svoju adresu, a ISP-u ili nekom hostu na internetu izgleda kao da postoji samo jedna mašina u kući. NAT je integriran u bežičnom ruteru koji je deo našeg AP (Access Point) i često je deo Firewall-a.

### Kako NAT radi?

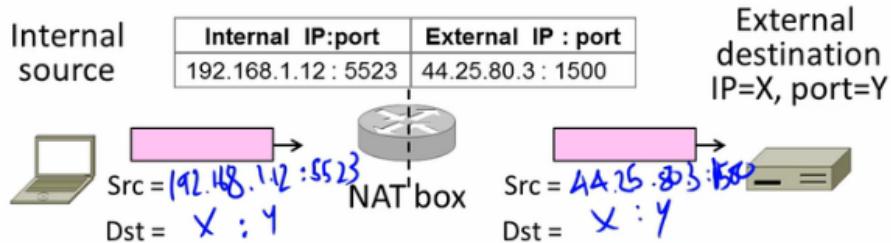
On održava tabelu (preslikavanje) unutrašnjih/spoljnih adresa. Unutrašnja strana odgovara kućnoj mreži, a spoljna strana mreži interneta. U tabeli je prikazano mapiranje između jedne IP adrese i porta i druge IP adrese i porta. Ovo je tipičan način kako se NAT koristi u kućama, zapravo je to preslikavanje IP+TCP port informacije. Razlog za to je da, ako imamo mnogo računara kod kuće a svi oni istovremeno koriste internet ne možemo ih sve mapirati u jednu IP adresu a da se ne zbumimo. Želimo da se ulazi u ovaj tabeli jedinstveno mapiraju u izlaze na tabeli. Da bismo to ostvarili koristimo brojeve portova, tako da mapiramo kombinaciju IP adrese i porta u drugu kombinaciju IP adrese i porta na jedinstven način. Portovi su neophodni kako bi mapiranje bilo 1-1, jer je spoljih adresa manje (obično samo jedna).

Шта рачунар мисли	Шта ISP мисли
Унутрашњи IP:port	Спољни IP : port
192.168.1.12 : 5523	44.25.80.3 : 1500
192.168.1.13 : 1234	44.25.80.3 : 1501
192.168.2.20 : 1234	44.25.80.3 : 1502

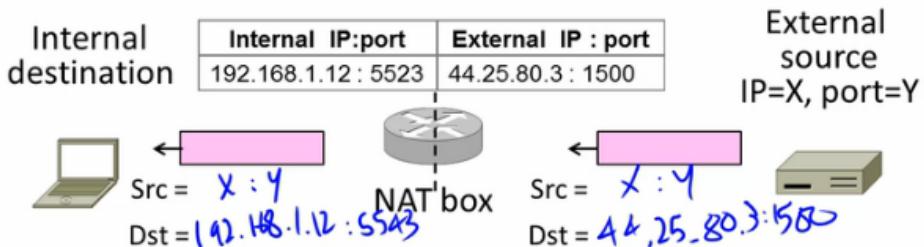
Na slici imamo primer jedne takve tabele. Sa leve strane se nalaze IP adrese u kućnoj mreži, to su privatne IP adrese. One se mapiraju u jednu javnu IP adresu na spoljnoj mreži. Način na koji ih razlikujemo je pomoću portova. Kada smo se konektovali, ako je u pitanju TCP konekcija imamo TCP port. Može se desiti da su dva računara u kućnoj mreži konektovana na isti port (poslednja dva su konektovana na 1234), ali pošto ih mapiramo u jednu IP adresu portovi moraju da budu različiti. Dakle, svi portovi u desnoj koloni moraju biti različiti i uopšte ne moraju odgovarati portovima sa leve strane. NAT će uglavnom portove sa desne strane birati nasumično, jer je bolje zbog sigurnosti.

**Prilikom slanja podataka iz lokalne mreže svakom IP paketu se menja adresa pošiljaoca u skladu sa zadatim preslikavanjem (s leva na desno).** To se može videti na slici.

Kada imamo host koji se nalazi u unutrušnjoj mreži on ima privatnu IP adresu 192.168.1.12 i konektovan je na port 5523. On šalje paket u spoljašnju mrežu u čijem zaglavlju je napisana IP adresa izvora što je 192.168.1.12 : 5523, i IP adresa cilja što je neko X : Y. Međutim, NAT prepravlja to zaglavlje tako da adresa izvora bude 44.25.80.3 : 1500, jer je pročitao iz tabele da se u tu javnu adresu slika privatna. IP adresa cilja ostaje ista.



**Prilikom prihvatanja podataka iz spoljne mreže svakom IP paketu se menja adresa primaoca u skladu sa zadatim preslikavanjem (s desna na levo).** To se može videti na slici. Situacija je analogna prethodnoj.



## NAT loše strane

- **Narušena je “čistoća” slojevitosti**  
Radi na mrežnom sloju, a barata TCP portovima.
- **Paketi mogu da se primaju samo ako je prethodno bilo poslatih paketa.**  
To je ranije odgovaralo za korišćenje veba jer klijent kod kuće uspostavi vezu sa veb serverom i onda je server mogao da odgovori. Ali to ne može da funkcioniše ako imate server kod kuće i želite da vas ljudi kontaktiraju, ili ako koristiti peer-to-peer tehnologiju.
- **Teško je, gotovo nemoguće, korisiti servere preko NAT-a.**

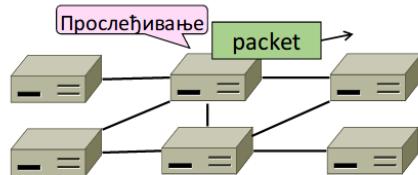
## NAT dobre strane

- **Smanjuje potrebe za javnim IP adresama**  
Dovoljna jedna po domaćinstvu. Mnoga domaćinstva koriste NAT.
- **Lako se instalira**  
Možete sami da instalirate kod kuće i to brzo.
- **Često ima u sebi neki vid zaštite od upada (firewall)**  
Zato što se spoljni hostovi ne mogu lako povezati sa unutrašnjim hostovima. Pošto NAT ne može da ih odgovarajuće mapira, odbaciće ih.
- **Pomaže i po pitanju privatnosti**  
Zato što je možda 7 mašina koristilo jednu IP adresu, pa izgleda kao da je jedna mašina sve vreme. Tako da se ne zna ko je šta radio.

**31. Rutiranje, mehanizmi alokacije protoka, modeli isporuke, ciljevi rutiranja, principi dizajna algoritama rutiranja, rutiranje sa najkraćim putevima (najmanjim troškom), Dijkstrin algoritam.**

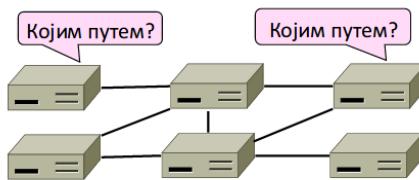
Prosleđivanje je proces slanja paketa susednim čvorovima.

To je proces koji se odvija lokalno u čvoru. Čvor dobije paket I prosledi ga.



**Rutiranje** je proces određivanja putanja kojima će se prosleđivanje vršiti.

To je proces oko koga su se dogovorili svi čvorovi da bi znali gde treba da šalju kroz mrežu. Dakle, rutiranje je proces koji uključuje sve čvorove u mreži. Već smo videli veoma jednostavnu vrstu rutiranja u vidu razapinjućeg stabla. Želimo da iskoristimo sve veze koje imamo između rutera. To nismo mogli da postignemo sa razapinjućim stablom, jer ono ne može da ima sve veze u isto vreme inače bi došlo do stvaranja ciklusa, pa samim tim to više ne bi bilo razapinjuće drvo.



## Iz perspektive alokacije protoka

Alokacija protoka je ključni aspekt rutiranja i želimo da ga dobro iskoristimo.

Rutiranje alocira protok tako da ima na umu otkaze čvorova (postoje i drugi mehanizmi alokacije). To znači da će putanje ponovo biti izračunate ako veza otkaže ili ako čvor otkaže. Ukoliko dođe do otkaza veze ili čvorova vreme reakcije, odnosno vreme alociranja protoka se izražava u sekundama.

Механизам	Време реакције/ Адаптација ка
Рутирање осетљиво на оптерећење	Секунде / Критични чворови оптерећења
Рутирање	Минути / Откази чворова
Обликовање протока	Сати / Оптерећење мреже
Резервација протока	Месеци / Корисници мреже

Postoje i druge vremenske skale u kojima možemo alocirati protok. Pogledajmo poslednji red. Najveća vremenska skala se odnosi na rezervaciju protoka. Rezervacija protoka se odnosi na to kako sagraditi mrežu tako da odgovara korisnicima i onome za šta će je oni koristiti. To može da traje mesecima, čak i godinama. Primer je ako imamo mnogo mušterija koji šalju podatke između Sijetla i San Franciska. Logično je napraviti direktni put između te dve lokacije. Zato je rezervacija protoka zadužena za taj problem.

Oblikovanje protoka se odnosi na prilagođavanje rute u vremenskom periodu od nekoliko sati, minuta do sati do čak nekoliko dana, u zavisnosti od opterećenja mreže. Ako opet uzmemosao primer mrežu SAD-a, opterećenje može biti drugačije u zavisnost koje je doba dana. Možda imamo mnogo saobraćaja na Istočnoj i Zapadnoj obali u različito doba dana. Zato možda želite da podesite

rute kada je gužva na Zapadnoj obali i kada je gužva na Istočnoj obali na drugaćiji način. Time se bavi oblikovanje protoka – prilagođava opterećenje mreže polako preko dana i preko nedelje.

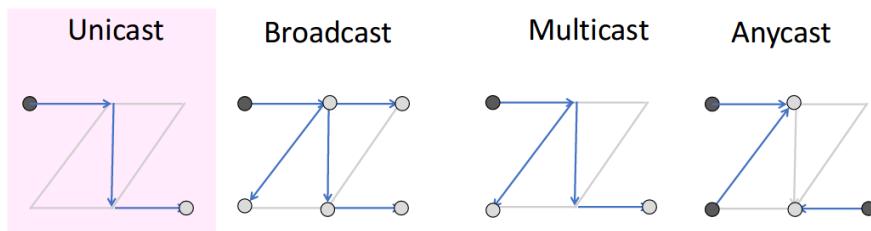
Kao što smo rekli, rutiranje je prilagođavanje otkazima opreme, pronalaženje dobrih okolnih puteva korišćenjem samo opreme koja radi. Tako da će se rute menjati na nekoliko minuta u najgorem slučaju (ako imamo mnogo otkaza). Dinamičniji oblik alokacije protoka je rutiranje osetljivo na opterećenje. Ovo je prilagođavanje u vremenskom periodu od nekoliko sekundi oko hot spot-ova u saobraćaju. Na primer, ako ima mnogo saobraćaja između Sijetla i San Franciska, može doći do nagomilavanja i zagušenja u Oregonu, u tom slučaju ima smisla preusmeriti neki saobraćaj preko Čikaga. To je duži put, ali ako ima zagušenja u saobraćaju, prednost je ići dužim putem i zaobići gužvu.

Mi se fokusiramo na rutiranje.

## Modeli isporuke

Rutiranje je zapravo veoma širok pojam. Mi ćemo se fokusirati samo na jednu vrstu rutiranja. Na slici imamo četiri različita modela isporuke odnosno kako mreža isporučuje saobraćaj. Nama je od interesa Unicast model, sa leve strane.

Različiti algoritmi rutiranja za različite modele.



**Unicast** je vrsta rutiranja koja isporučuje saobraćaj od izvora do destinacije. Na slici imamo rutu između jednog izvora i jedne destinacije.

**Broadcast** pokušava da isporuči saobraćaj od jednog čvora ka svim ostalim čvorovima u mreži. Na slici vidimo da, ako pratimo strelice, isporučićemo kopije paketa svakom čvoru u mreži.

**Multicast** je kao podskup od Broadcast. Isporučujemo paket od jednog čvora ka grupi čvorova, ali ne svima na mreži. Na slici se paket isporučuje dvema razim tačkama, ali postoje još tri čvora kojima nije poslat paket.

**Anycast** je veoma interesantan model. Kada se paket pošalje adresi koja je anycast, mreža će ga isporučiti najbližem članu u grupi. Ako pošaljemo od gornjeg levog crnog čvora, paket će stići roze čvoru koji je desno od njega, jer mu je on najbliži član grupe. Ako pošaljemo sa druge destinacije, na primer od donjeg desnog crnog čvora na istu anycast adresu, paket će biti poslat na drugu destinaciju. Ima istu adresu, a to je najbliža instanca iz te grupe čvoru koji je pošiljalac. Deluje čudno, ali je ovaj model veoma koristan u mreži. Ako, na primer, želimo da pošaljemo paket najbližem servisu.

Mi se fokusiramo na Unicast.

## Ciljevi rutiranja

Ova svojstva u tabeli želimo da zadovoljavaju naši algoritmi rutiranja.

Својство	Значење
Тачност	Пronalazi путању која ради
Ефикасност	Рационално троши проток
Равноправност	Подједнака права чворова, не изгладњује неке чворове
Брза конвергенција	Брз опоравак након промена (отказа, нових чворова)
Скалабилност	Ради добро и када мрежа (брз чворова, веза, ...) расте

**Tačnost** – želimo da algoritam pronalazi putanju koja radi. Deluje očigledno, ali nije tako lako kao na mapi uočiti kojim putem treba da se ide da bi se stiglo od jedne do druge tačke, jer mnogi algoritmi ne vide širu sliku.

**Efikasnost** – želimo da algoritam pronađe efikasne putanje, odnosno one koje racionalno troše protok.

**Ravnopravnost** – želimo da putevi budu ravnopravni, tj. da se neki čvorovi ne izgladnjuju, da čvorovi imaju podjednaka prava. Na primer, ne bi bilo razumno da pronađemo dobre putanje koje rade samo za neke čvorove, ali da neki drugi čvorovi uopšte ne dobijaju isti saobraćaj. Svaki čvor na mreži treba da ima mogućnost da razumno šalje pakete kroz mrežu.

**Brza konvergencija** – ako se desi neki otkaz, potrebno je prilagoditi se toj promeni, želeli bismo da se ruteri brzo oporave.

**Skalabilnost** – znamo da mnoge mreže koje sagradimo rastu veoma brzo. Želimo da šeme rutiranja rade dobro čak i kada mreža raste. To se zovu skalabilni algoritmi.

## Principi za dizajn algoritama rutiranja

- **Decentralizovani i distribuirani** (rade u decentralizovanom i distribuiranom okruženju)

1. **Čvorovi su ruteri, ne razmatramo korisničke računare**

Kada podatak dođe do poslednjeg ruteru, prosleđivanje ka korisničkom računaru razmatra sloj veze

2. **Svi čvorovi su ravnopravni, nema bitnijih čvorova**

Nemamo neki kontroler, tako da se čvorovi moraju dogovoriti šta će da rade.

3. **Čvorovi saznaju ukupno stanje mreže tako što razmenjuju poruke sa susedima**

Ne mogu svi čvorovi razgovarati direktno među sobom, jedino mogu da razgovaraju jedni sa drugima preko svojih suseda. Nemaju načina da dopru do bilo koga drugog sem svojih suseda. Čak ni ne znaju kako izgleda ostatak mreže.

4. **Čvorovi rade konkurentno**

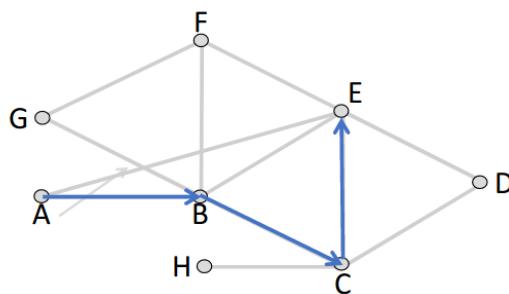
Rade paralelno i to otežava koordinaciju

5. **Mogu se desiti otkazi čvorova i veza ili gubljenja poruka**

Želimo da budu u stanju da odrede ispravne rute čak i kada neki od čvorova ili veza otkazu. Koji god čvorovi i veze rade ispravno, želimo njih da to srede i da nađu dobre putanje.

## Koji put je najbolji?

Mora se prvo definisati prema čemu najbolji: dužini, ceni, kašnjenju ili kombinaciji istih...



## Mere troškova:

Mere troškova zavise od toga šta operator mreže želi.

Veliki broj mogućnosti:

- **Kašnjenje:**

Želimo da dobra putanja ima malo kašnjenje, da izbegava zaobilazne puteve.

- **Protok:**

Želimo putanje koji imaju veliki protoki, da zbegavaju spore veze. Na taj način u našu mrežu može da stane više saobraćaja.

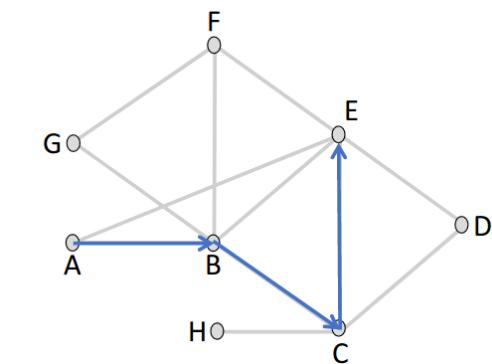
- **Novac:**

Želimo da putanja izbegava skupe veze.

- **Broj hopova:**

Smanjuje iskorišćenost komunikacione opreme.

Mrežu ćemo prikazivati kao graf.



## Najkraći putevi

Napravićemo funkciju cene tako da možemo da dodelimo cenu različitim putanjama, a onda ćemo izabrati putanju koja ima najmanju ukupnu cenu. I tu putanju ćemo smatrati najboljom. Često put sa najmanjom cenom nazivamo najkraćim putem. To se dobija kada se kao cena puta postavi razdaljina.

Dodeljivanjem cena na različite načine moći ćemo da obuhvatimo mnogo različitih faktora.

Najkraći putevi rade na sledeći način:

1. **Svakoj vezi ćemo dodati cenu ili razdaljinu.**

To ćemo uraditi tako da obuhvatimo faktore koji su nam od interesa. Ukoliko želimo da smanjimo kašnjenje kroz mrežu govorićemo o konkretnoj razdaljini. Ako je svaka veza 1 mogli bismo da minimizujemo broj hopova. Ako nam je ključan protok, možemo brzim vezama da dodelimo manju cenu, a sporijim veću cenu. Na taj način će nam putanja manje cene birati brže veze.

2. **Definišemo najbolju put između svaka dva čvora u mreži kao najkraći put između njih.**

3. **Ako neke dve putanje imaju istu cenu, nasumično izaberemo jednu.**

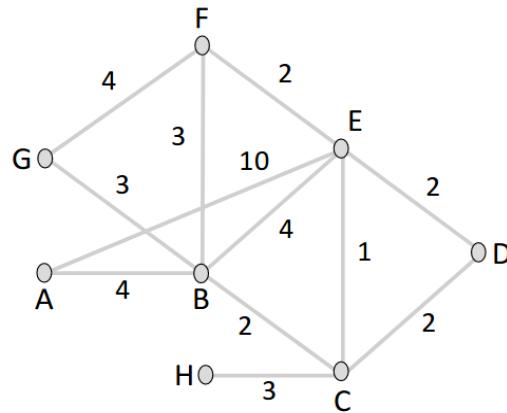
Pogledajmo jedan primer. Na slici dole imamo primer jedne mreže gde je svakoj vezi dodeljen broj, koji predstavlja cenu korišćenja te veze.

Prepostavimo da je graf:

1. Neusmeren (paketi u oba smera)
2. I da ima simetrične troškove

Moguće je modelovati algoritme i za neusmerene, asimetrične grafove.

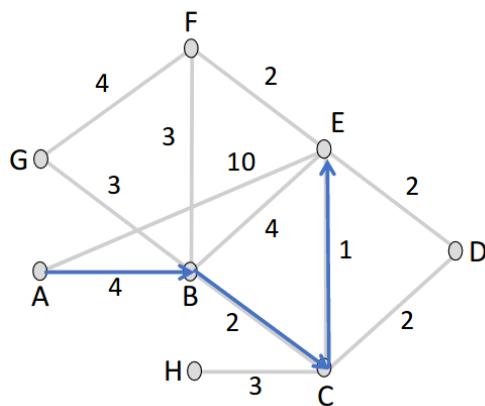
Pronaći najkraći put za A -> E



Rešićemo problem čisto gledajući graf. Imamo više mogućnosti. Imamo direktnu vezu između A i E, koja ima cenu 10. Možemo i bolje od toga, ako idemo A -> B -> E, cena veze je 8, tako da je to kraći put. Da li je to najkraći mogući put? Ne. Postoji još bolji put kroz mrežu, a to je A -> B -> C -> E.

ABCE je najkraći put

$$\text{dist (ABCE)} = 4 + 2 + 1 = 7$$



Ona ima trošak manji od ostalih puteva:

$$\text{dist (ABE)} = 8$$

$$\text{dist (ABFE)} = 9$$

$$\text{dist (AE)} = 10$$

$$\text{dist (ABCDE)} = 10$$

### Princip optimalnosti:

Putevi imaju princip optimalnosti.

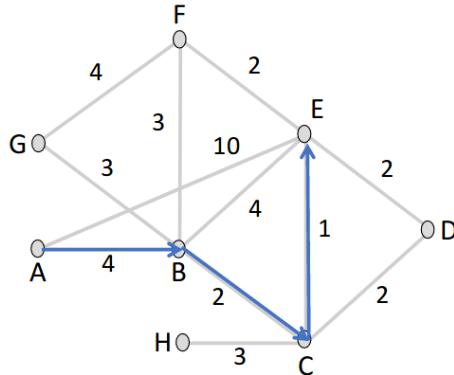
Primetiti da su segmenti optimalnih (najkraćih puteva) takođe najkraći putevi.

Dakle, u jednom najkraćem putu imamo više različitih najkraćih puteva.

ABCE je najkraći put između A i E -> ali su i ABC, AB, BCE, CE najkraći putevi između A i C, A i B, itd...

Zašto je to tako?

BCE je najkraći put ako je ABCE najkraći put jer, da smo mogli da nađemo bolji put od BCE onda bismo taj bolji put korstili da dođemo do E. Mi znamo da je ovaj put od A do E najkraći tako da nije moguće naći bolji put od BCE. To bi bila kontradikcija.



## Dijkstrin algoritam

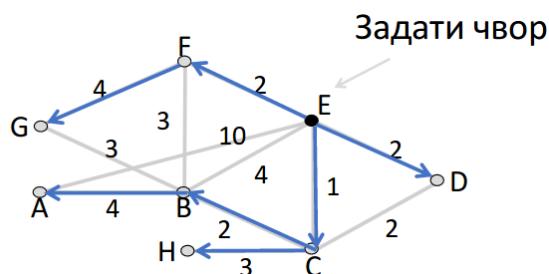
Dijkstrin algoritam:

- Računa najkraće puteve između zadatog čvora i svih ostalih čvorova**

Da bi to postigao mora da zna topologiju mreže I cene između čvorova moraju biti nenegativne.

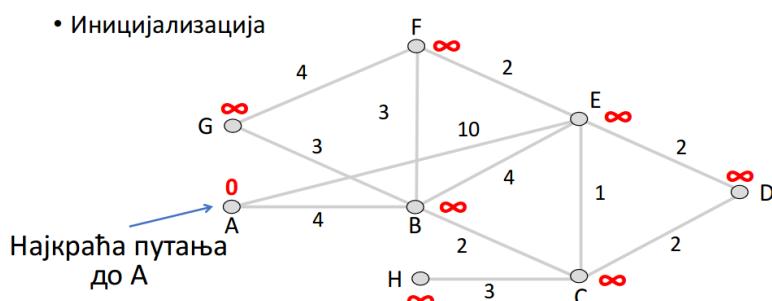
- Rezultat je drvo**

Na ovaj način ćemo moći da napravimo tabelu prosleđivanja za zadati čvor.



- Postavimo sve čvorove kao privremene** (inicijalizacija)
- Postavimo aktuelne vrednosti između zadatog i svih čvorova** (inicijalizacija):
  - Na vrednost 0 ako je udaljenost do samog sebe
  - Na vrednost  $\infty$  za sve ostale čvorove (jer još uvek ne znamo kako da dođemo do njih)
- Dok ima privremenih čvorova** (petlja):
  - Uzmi privremeni čvor X, koji ima najmanju udaljenost od zadatog čvora
  - Izbaci X iz skupa privremenih čvorova i dodaj odgovarajuću vezu ka njemu u drvo
  - Umanji udaljenost čvorova susednih sa X u skladu sa novododatkom udaljenošću

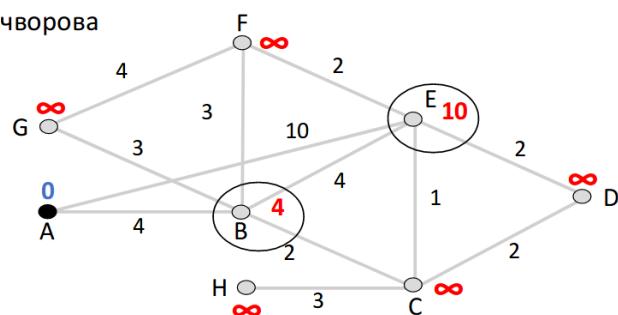
Primer: Inicijalizacija: Zadati čvor je A. Njegova početna cena je 0, a cena svih ostalih je beskonačno.



## Petlja:

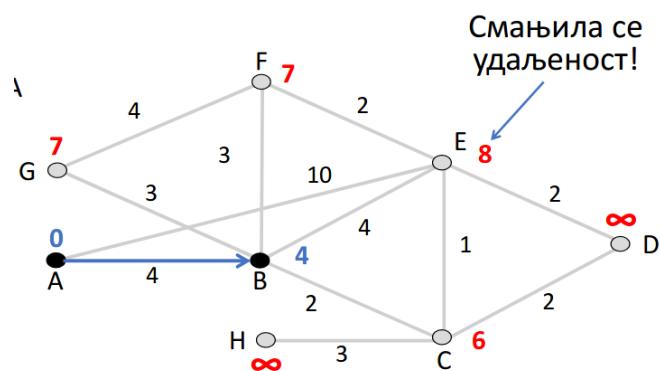
Prvi korak u petlji je da uzmemo čvor sa najmanjom cenom, a to je A. Na slici je njegova cena obojena u plavo, jer je fiksna, neće se više menjati, a čvor je obojen u crno. Sada ćemo posetit susede čvora A i videti mogu li se njihove cene smanjiti. Jedan sused je B, drugi sused je E. Oni su zaokruženi na slici. B je udaljeno od čvora A vezom čija je cena 4, to je manje od beskonačnosti tako da ćemo smanjiti cenu čvora B sa beskonačnosti na 4. Isto ćemo uraditi sa E. Cena veze između A i E je 10, to je manje od beskonačnosti, pa smanjujemo cenu E na 10.

- Умањи удаљености чворова суседних са A



- Biramo B, jer je on najbliži A
- Takođe ažuriramo čvorove koji su susedni sa B
- U sledećem koraku biramo C, jer je on novi najbliži

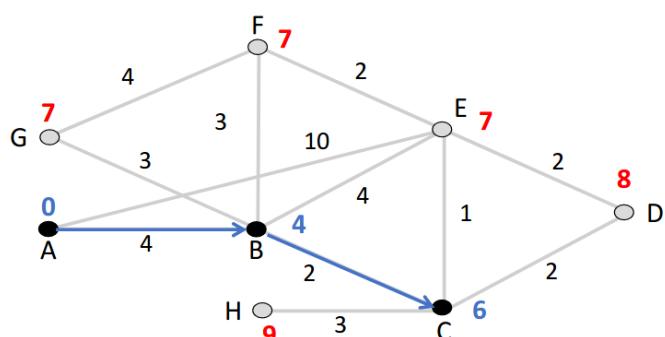
Šta smo uradili u sledećoj iteraciji? Biramo sledeći čvor sa najmanjom cenom. To je B, čija je cena 4. Dodamo B u drvo najkraćih puteva, na slici je dodata grana od A do B. Gledamo sve čvorove povezane sa B i gledamo možemo li da iskoristimo veze sa B da im smanjimo cenu. Prvo gledamo C, njegova cena je bila beskonačnost, ali cena veze između B i C je 2, a cena B je 4, tako da je nova cena C sada 6. E je udaljeno od B vezom cene 4, tako da se sada može dostići putem cene 8, to je manje od 10 (njegove prethodne cene), tako da sada cenu čvora E postavljamo na 8. Tokom vremena pronalazimo bolje puteve. Na sličan način smo smanjili cene čvorova F i G (sad su oba po 7).



- Biramo C

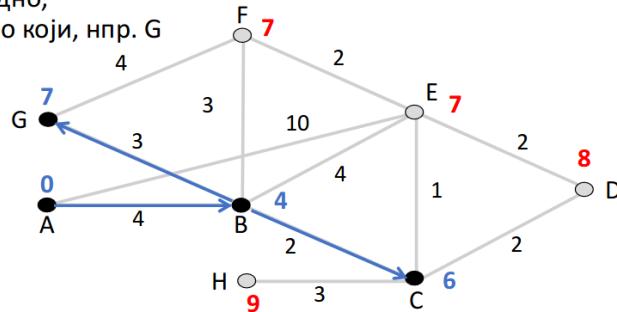
Opet biramo čvor sa najmanjom cenom od preostalih čvorova i to je C, C je dodato u drvo. Cena čvora H se smanjila na 9, D na 8. Ako

idemo preko C, opet se smanjuje cena čvora E, koji je sada 7.



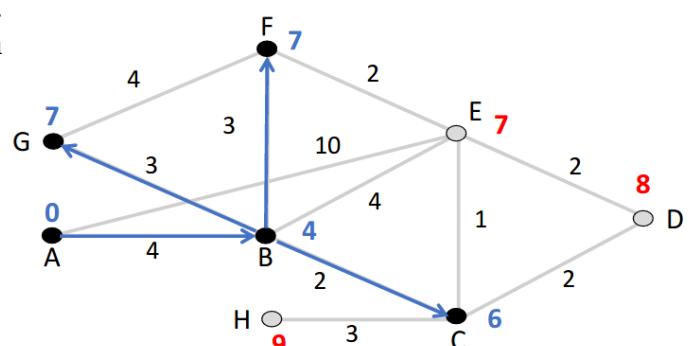
Opet biramo sledeći čvor sa najmanjom cenom, ali pošto ih ima tri sa najmanjom cenom 7, biramo bilo koji. U ovom slučaju smo izabrali G. Do G se može doći preko B, putem cene 7, tako da smo ga dodali u drvo. G ima jednog suseda, to je F. Cena da se dođe do F preko G je 11, a to nije veće od 7, što znači da je dolazak do F preko G loš potez.

- Кад је свеједно,  
бирамо било који, нпр. G



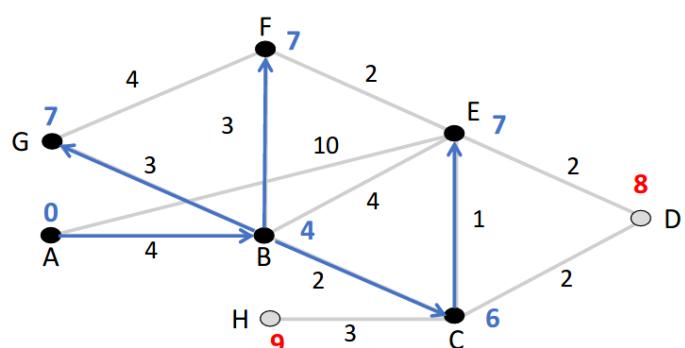
#### • Biramo F

U sledećem koraku biramo F, dodajemo ga u drvo. Pogledamo njegove susede i vidimo da putanja preko F nijednom susedu ne smanjuje cenu.



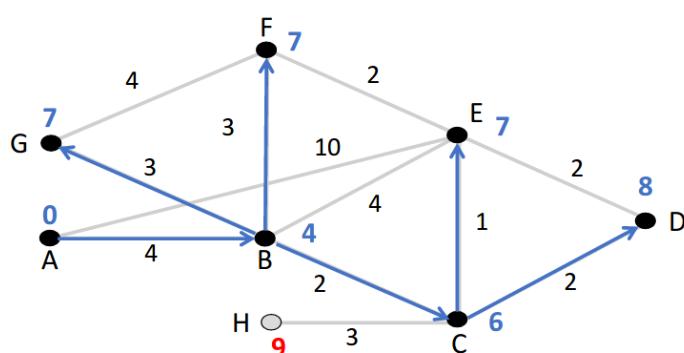
Biramo E, dodajemo ga u drvo. Nijednom njegovom neposećenom susedu se ne smanjuje cena.

#### • Бирали Е

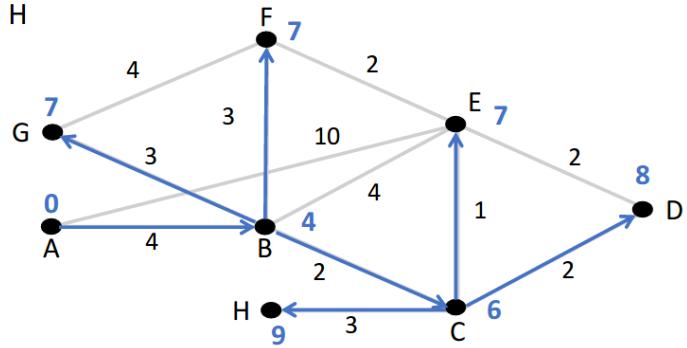


Biramo D, dodajemo ga u drvo. Nijednom njegovom neposećenom susedu se ne smanjuje cena. Mada D ni nema neposećenih suseda.

#### • Бирали D



Biramo H, dodajemo ga u drvo. Nijednom • И коначно H njegovom neposećenom susedu se ne smanjuje cena. Mada H ni nema neposećenih suseda.



Sada smo završili. Našli smo drvo od čvora A ka svim ostalim čvorovima.

## Karakteristike algoritma

- **Pronalazi puteve ka čvorovima prema rastućem poretku dužina**  
Koristi svojstvo dekompozicije optimalnosti. Pošto u svakom koraku biramo čvor X do kojeg je najkraći put, ne može se desiti da put do X preko nekog kasnije dodatog čvora Y bude kraći!
- **Vreme izvršavanja zavisi od efikasnosti pronalaženja najboljeg privremenog čvora (onog do koga postoji najkraći put)**  
Može se koristiti npr. hip struktura. Složenost je veća od linearne. Kako se povećava mreža vreme izvršavanja se još više povećava.

## 32. Rutiranje zasnovano na vektoru razdaljine

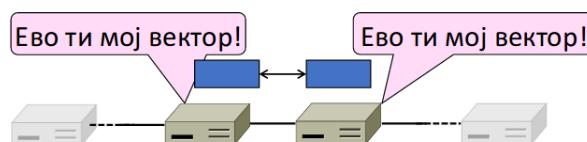
### DV rutiranje (Distance vector routing)

Videli smo kako koristimo Dijkstrin algoritam da odredimo najkraće puteve kada vidimo topologiju mreže. Sada ćemo videti kako da odredimo najkraće puteve u ditribuiranom okruženju mreže. I za to ćemo koristiti rutiranje zasnovano na vektoru razdaljine. Algoritam radi dobro, tačan je, ali slabo konvergira u slučaju nekih otkaza.

Šta je distribuirano okruženje?

Potpuno je drugačije od centralizovanog okrženja u kome imamo celu toplogiju mreže. Čvor u ovom okruženju samo zna svoje susede i možda cenu veze do svojih suseda. Dakle, čvor ima samo lokalno znanje, ne zna celokupnu topologiju. Čvor može da komunicira samo sa svojim susedima koristeći poruke. Svi čvorovi koriste isti algoritam konkurentno, ne postoji nikakav šef, vođa ili centralizovano mesto koje kontroliše. Svaki čvor je jednak i među sobom oni nekako moraju da odluče koji su dobri putevi. Čvorovi i veze mogu da otkažu, a poruke se mogu izgubiti. To je problematično jer je možda čvor otkazao, a ostali čvorovi moraju da budu u stanju da pronađu dobre puteve.

- Ovaj pristup se zasniva na razmeni vektora (tabela) razdaljine između susednih vektora
- Jedan od prvih pristupa (Arpanet)  
U praksi se sada retko koristi



Svaki čvor održava vektor udaljenosti i sledećih hopova za sve ciljne čvorove.

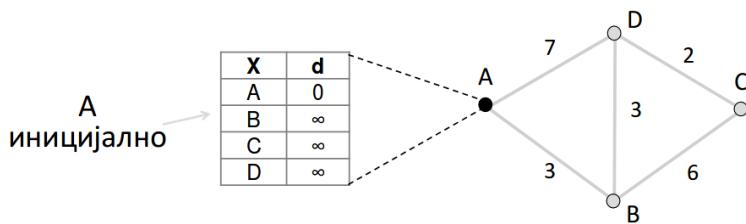
Svaki čvor radi sledeće:

1. Inicijalizuje udaljenost do samog sebe na 0, i udaljenost ka svim ostalim ciljnim čvorovima na  $\infty$  (beskonačno).
2. Periodično šalje svoj vektor ka susedima.
3. Preračunava svoj vektor udaljenosti na osnovu vektora dobijenih od svojih suseda

## DV primer

Imamo jednostavnu mrežu sa 4 čvora – A, B, C i D. Svaki čvor u ovoj mreži će imati svoju instancu DV algoritma i čuvaće svoj vektor udaljenosti.

Na slici imamo primer vektora udaljenosti koji čuva čvor A na početku. A iz čvora A se može dostići vezom cene 0, što je logično jer je to on sam. Do B, C i D je beskonačnost, jer još uvek ne zna kako da dođe do njih. Na slici vidimo da A može da komunicira jedino sa B i D, ne može direktno da dostigne C. Dakle, čvor A razmenjuje vektore samo sa B i D.



A šalje svoj vektor udaljenosti čvorovima B i D. B, D i ostali čvorovi šalju svoje vektore udaljenosti ostalim čvorovima. Dakle, A će primiti dva vektora udaljenosti – jedan od B, drugi od D. To je prikazano na donjoj slici u prvoj tabeli. Da vidimo šta je B rekao A. B je rekao da može samo da dostigne B, a sve ostalo je beskonačnost. Slična situacija je i sa D. A mora da izabere najbolje ulaze nakon što ažurira cene veza. Za B posmatramo  $B+3$ , jer je 3 cena veze između A i B. A za D posmatramo  $D+7$ , jer je 7 cena veze između A i D. O čvoru C ne znamo ništa. To se može videti u drugoj tabeli. Mi želimo da izaberemo najmanju cenu. U trećoj tabeli se vidi vektor udaljenosti čvora A nakon prve iteracije, A je zapravo naučio najbolje 1-hop puteve koji su mogući u toj mreži. Jednim hopom se samo može dostići B ili D. A zapravo uvek ažurira vektor korišćenjem vektora  $\min(B+3, D+7)$ .

Nakon prvog koraka, svako nauči puteve dužine 1

X	B какже	D какже
A	$\infty$	$\infty$
B	0	$\infty$
C	$\infty$	$\infty$
D	$\infty$	0

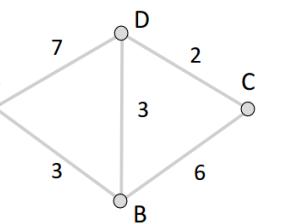
 $\rightarrow$ 

B	D
+3	+7
$\infty$	$\infty$
3	$\infty$
$\infty$	$\infty$
$\infty$	7

 $\rightarrow$ 

A научи	
d	Y
0	--
3	B
$\infty$	--
7	D

= научен бољи пут

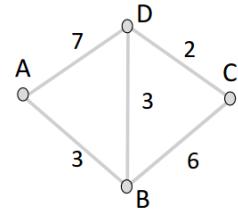


Ovaj proces koji smo opisali se odvija u svim čvorovima, ne samo u A. Na donjoj slici, prva tabela nam pokazuje sve vektore udaljenosti koje B, C i D razmenjuju sa svojim susedima. B uči na osnovu vektora  $\min(A+3, C+6, D+3)$ . C uči na osnovu vektora  $\min(B+6, D+2)$ . Ako pratimo ovaj proces možemo da izračunamo 1-hop najbolje puteve koje B, C i D uče.

Nakon svih prvih razmena, svi uče tabelu 1-hop puteva.

X	A какже	B какже	C какже	D какже		A научи d Y	B научи d Y	C научи d Y	D научи d Y
A	0	$\infty$	$\infty$	$\infty$		0 --	3 A	$\infty$ --	7 A
B	$\infty$	0	$\infty$	$\infty$		3 B	0 --	6 B	3 B
C	$\infty$	$\infty$	0	$\infty$		$\infty$ --	6 C	0 --	2 C
D	$\infty$	$\infty$	$\infty$	0		7 D	3 D	2 D	0 --

A научи d Y	B научи d Y	C научи d Y	D научи d Y
0 --	3 A	$\infty$ --	7 A
3 B	0 --	6 B	3 B
$\infty$ --	6 C	0 --	2 C
7 D	3 D	2 D	0 --

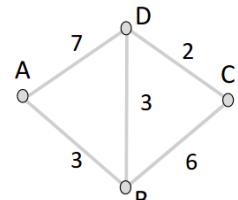


= научен болни путь

Tu se proces ne završava. U drugoj razmeni prva tabela pokazuje vektore udaljenosti koji će biti ažurirani, a to će biti najbolji putevi koji su naučeni nakon druge razmene, pa će oni dalje biti prosleđivani. Nakon drugog kruga razmena, naučeni su svi najbolji putevi dužine 2 (2-hop putevi). Rekli smo da je  $A = \min(B+3, D+7)$ . Za A naravno ostaje 0, to je specijalan slučaj. Za B  $0+3=3$ , to ostaje isto. Za C preko B je  $6+3=9$ , a preko D je  $2+7=9$ , tako da je cena veze 9. A pošto je preko oba čvora ista cena, biramo bilo koji u ovom slučaju D. Za D preko B je  $3+3=6$ , a  $0+7=7$ , tako da A može da dođe do D preko B uz manju cenu.

X	A какже	B какже	C какже	D какже		A научи d Y	B научи d Y	C научи d Y	D научи d Y
A	0	3	$\infty$	7		0 --	3 A	9 B	6 B
B	3	0	6	3		3 B	0 --	5 D	3 B
C	$\infty$	6	0	2		9 D	5 D	0 --	2 C
D	7	3	2	0		6 B	3 D	2 D	0 --

A научи d Y	B научи d Y	C научи d Y	D научи d Y
0 --	3 A	9 B	6 B
3 B	0 --	5 D	3 B
9 D	5 D	0 --	2 C
6 B	3 D	2 D	0 --



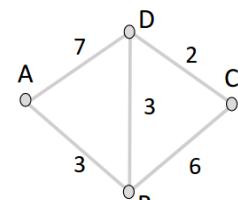
= научен болни путь

Процес се nastavlja.

Najbolji putevi до дужине 3. I postoje 3-hop putevi који су бољи. A може да дође до C преко B уз цену 8. Исто тако и у suprotnom smjeru.

X	A какже	B какже	C какже	D какже		A научи d Y	B научи d Y	C научи d Y	D научи d Y
A	0	3	9	6		0 --	3 A	8 D	6 B
B	3	0	5	3		3 B	0 --	5 D	3 B
C	9	5	0	2		8 B	5 D	0 --	2 C
D	6	3	2	0		6 B	3 D	2 D	0 --

A научи d Y	B научи d Y	C научи d Y	D научи d Y
0 --	3 A	8 D	6 B
3 B	0 --	5 D	3 B
8 B	5 D	0 --	2 C
6 B	3 D	2 D	0 --



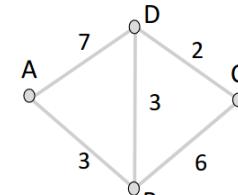
= научен болни путь

Ne postoje 4-hop putevi. Jer nema puteva u grafu koji su duži od 3.

Poslednji krug, nakon ovoga tabela konvergira (konvergencija naravno zavisi od grafa)

X	A какже	B какже	C какже	D какже		A научи d Y	B научи d Y	C научи d Y	D научи d Y
A	0	3	8	6		0 --	3 A	8 D	6 B
B	3	0	5	3		3 B	0 --	5 D	3 B
C	8	5	0	2		8 B	5 D	0 --	2 C
D	6	3	2	0		6 B	3 D	2 D	0 --

A научи d Y	B научи d Y	C научи d Y	D научи d Y
0 --	3 A	8 D	6 B
3 B	0 --	5 D	3 B
8 B	5 D	0 --	2 C
6 B	3 D	2 D	0 --



= научен болни путь

## DV rutiranje – robustnost

- Dodavanje veza ili čvorova:**

Vest putuje jedan hop po razmeni. Ovo nije preterano brzo.

- **Uklanjanje veza ili čvorova:**

Ako čvor otkaže on to ne javlja ostalima, nego susedi ne obavljaju više razmenu sa njim pa posle nekog vremena zaborave da uopšte postoji. Može se uvesti maksimalna starost razmenjenog vektora.

- **Razbijanje mreže na dva dela je ozbiljan problem!**

Razbijanje mreže na dva dela je kada se ona podeli i ne može se doći od jednog dela do drugog. To dovodi do problema koji se zove brojanje do beskonačnosti (count to infinity scenario). Postoje načini da se ovo razreši, ali ćemo preskočiti taj deo...

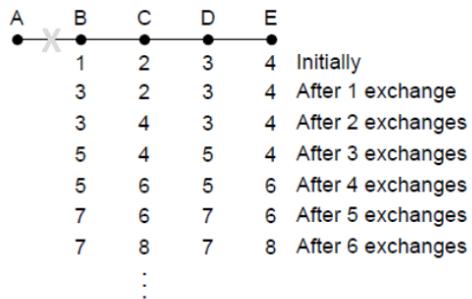
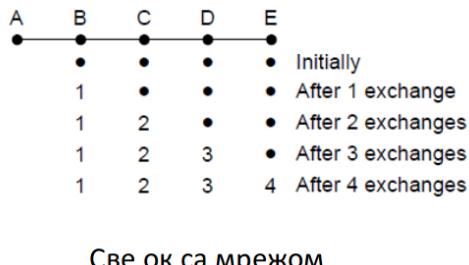
## Problem brojanja do beskonačnosti (count to infinity scenario)

Kako izgleda taj problem?

Na levoj slici je poželjan scenario. Pokazuje kako se informacija od A propagira kroz razmene. Vidimo kako vest putuje brzo. Nakon 1. razmene B zna da može da pristupi A, nakon 2. C zna da može da pristupi A itd.

Šta ako imamo otkaz između A i B, kao npr. na desnoj slici, i A više ne može da javi kako se može stići do njega. B dobija obaveštenje, ne od A, nego od C koji mu kaže: "Ja sam mogao da dostignem A po ceni 2", onda će B da kaže: "Ja ga mogu dostići po ceni 3", jer je to cena koju je dobio od C + 1. U sledećoj iteraciji C uči od B kako da dođe do A pa povećava cenu na 4. U sledećoj iteraciji B povećava cenu na 5 i D isto, a u sledećoj opet C povećava cenu itd. Sve zato što zavise jedni od drugih. To je nepoželjno ponašanje.

- Добра вест путује брзо, а лоша споро...

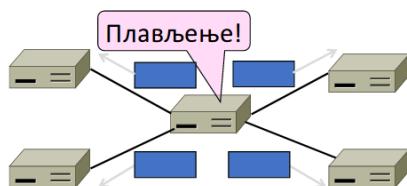


38

## 33. Plavljenje

### Plavljenje

Emitovanje poruke svim čvorovima pomoću tehnike plavljenja. Jednostavan mehanizam, ne preterano efikasan. Zove se plavljenje jer radi tako što obezbeđuje da se poruka pojavi svuda.



Izvršava se u distribuiranom okruženju tako da svi čvorovi izvršavaju ovo pravi konkurentno i nijedan čvor ne zna topologiju mreže.

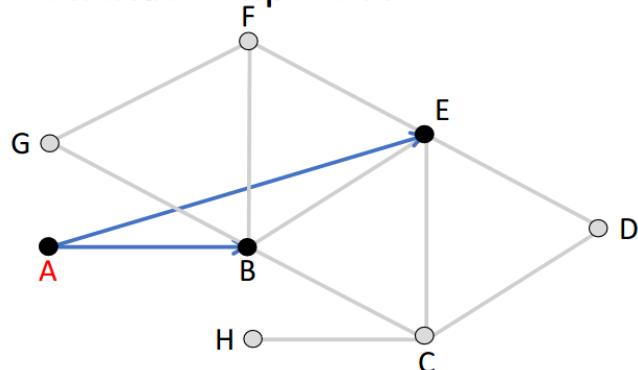
Pravilo na svakom čvoru:

- Nakon pristizanja poruke, prosledi je svim ostalim susedima.  
Sem onim susedima od kojih si je dobio.
- Zapamti nekako poruku kako je ne bi ponovo prosleđivao ako ti opet stigne.

Neefikasno, jer jedan čvor može da dobije višesruke kopije iste poruke.

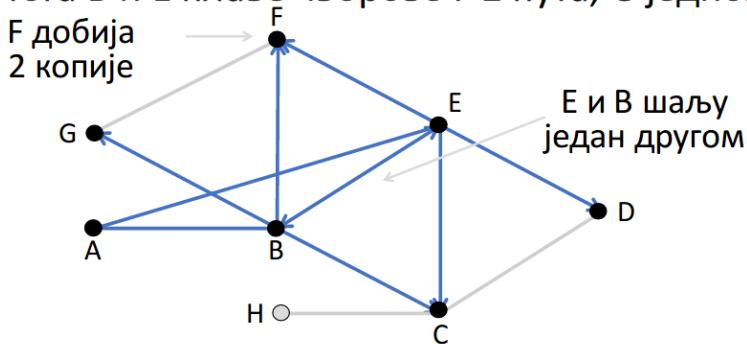
Pogledajmo primer. Plavljenje polazi iz A tako što A to šalje svojim susedima. A šalje poruku B preko AB i šalje poruku E preko AE.

- Плављење полази нпр. из А

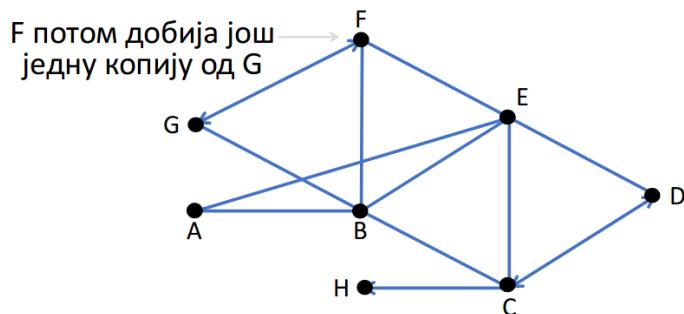


Poruke su stigle u B i E, sad oba ova čvora treba da proslede svojim susedima. B šalje C, E, F i G. E šalje D, C, B i F. Poruka se stvarno kopira po celoj mreži. F je dobio dve kopije – jednu od B i jednu od D. B i D su vršili plavljenje istovremeno pa su oboje poslali jedno drugome istu poruku, tako da je poruka prešla preko veze između njih u oba smera.

- Након тога В и Е плаве чворове F 2 пута, G једном итд.

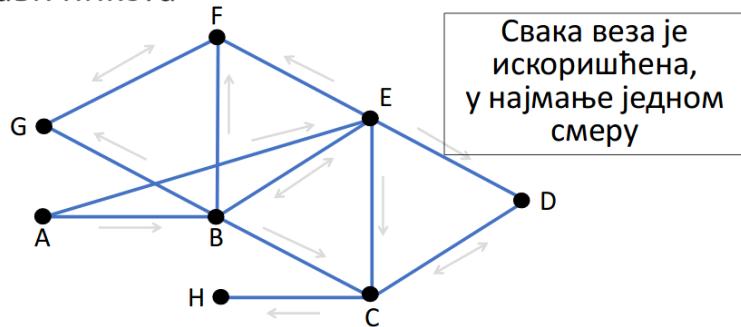


Prošli smo kroz A, B i E. Ostali čvorovi koji su skoro primili poruku, a nisu je prosledili su C, D, F i G. C je dobio dve kopije – jednu od B, jednu od E. Šalje drugim čvorovima – H i D. D je dobio poruku od E, pa šalje svom drugom susedu – C. Poruka je prešla kroz vezu između C i D dva puta. F je dobio dve kopije – od B i E, pa šalje G. G je dobio od B pa šalje F. F je dobio poruku preko svih svojih veza, a poslao ju je preko jedne.



Stigli smo do H, ali on nema koga da plavi.

- H не плави никога



Neefikasno je, jer da smo hteli da poruka dođe do F nije bilo potrebno da se šalje preko svih veza.

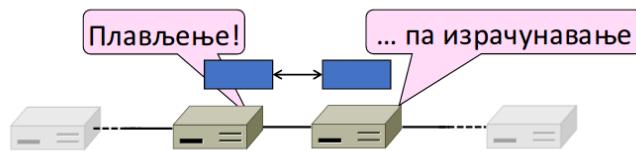
## Plavljenje – ostali aspekti

- **Prilikom pamćenja, dovoljno je zapamtitи само izvor i redni broj poruke.**  
Sledeća poruka se prihvata samo ako ima viši redni broj. Može se samo zapamtiti da je A poslednji plavio porukom sa rednim brojem 7. Ako se dobija poruka od A sa brojem 7 ili manje znamo da smo je već dobili.
- **Moguće je omogućiti i ARQ kako bi plavljenje bilo pouzdaniјe.**  
Kad plavimo i jedan čvor pošalje svom susedu, sused može da pošalje acknowledgement. Ako čvor ne dobije acknowledgement može da ponovo pošalje poruke koristeći time out.

## 34. Rutiranje zasnovano na stanju veza

### LS rutiranje (Link state routing)

- **Drugачије од DV rutiranja**  
DV rutiranje je raspodelilo posao izačunavanja puteva svim čvorovima u mreži. LS rutiranje daje svima kopiju topologije i svako izračuna svoje putanje. U LS se dosta posla ponavlja, za razliku od DV gde su čvorovi sarađivali. LS je slično tome da date mapu svima i da svako pronađe svoje puteve. LS se široko koristi u praksi.
- **Koristi prethodno uvedene koncepte**



Podsetimo se prvo u kakvom okruženju radimo. Okruženje je distribuirano što otežava rutiranje.

1. Čvorovi samo znaju sa kim su direktno povezani, dakle, samo znaju svoje susede i cenu veze do svojih suseda.
2. Čvorovi mogu da komuniciraju samo sa svojim susedima koristeći poruke.
3. Svi čvorovi pokreću isti algoritam konkurentno. Nijedan čvor nije poseban, ne postoji centralizovani kontroler. Moraće da sarađuju i da usklade svoje akcije.

- Čvorovi/veze mogu da otkaži, želimo da ostatak mreže pronađe korektne putanje. Poruke se mogu izgubiti.

Dve faze:

- Čvorovi plave mrežu informacijama o svojoj lokalnoj topologiji (susedima)**

Koriste link state packets. Svaki čvor je tako u stanju da rekonstruiše celokupnu topologiju.

Ovo ponavljamo svaki put kada se topologija promeni i kada se počinje.

- Svaki čvor računa svoju tabelu prosleđivanja**

Ovo se postiže npr. Dijkstrinim algoritmom

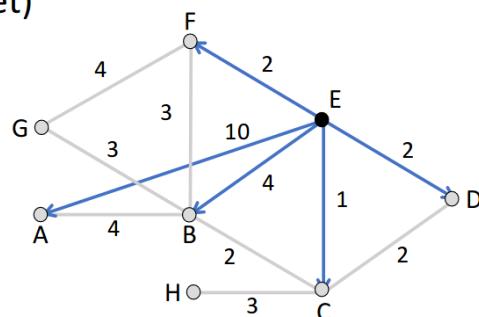
## Faza 1: Rekonstrukcija topologije

Svaki čvor plavi mrežu svojim LS paketom. Na slici vidimo LS paket za čvor E. Njegov paket sadži vezu ka A sa cenom 10, vezu ka B sa cenom 4, vezu ka C sa cenom 1, vezu kad D sa cenom 2 i vezu ka F sa cenom 2. To je sadržaj paketa koji će čvor E da plavi ostalim čvorovima. I svi ostali čvorovi će slično uraditi sa svojim delom topologije.

- Сваки чвор плави мрежу својим LS пакетом (link state packet)

LSP čvora E

Seq. #
A 10
B 4
C 1
D 2
F 2



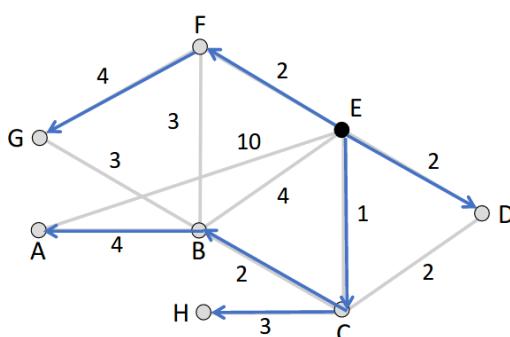
## Faza 2: Računanje najkraćih puteva

- Svaki čvor ima punu topologiju**  
Dobija je tako što kombinuje sve LSP.
- Svaki čvor izvrši Dijkstrin algoritam**  
Deluje da ima suvišnog izračunavanja ako se gleda cela mreža.

## Tabela prosleđivanja

Na slici vidimo najkraće puteve od čvora E do svakog drugog čvora u mreži, to je rezultat Dijkstrinog algoritma. Sa desne strane imamo tabelu prosleđivanja čvora E. Da bi E došao do A, treba prvo da ode u C i to je sve što E treba da zna, nakon toga C preuzima brigu o paketu. Na sličan način i za ostale čvorove.

Након Дијкстриног алгоритма



Табела чвора E

X	Y
A	C
B	C
C	C
D	D
E	--
F	F
G	F
H	C

## Promene u mreži

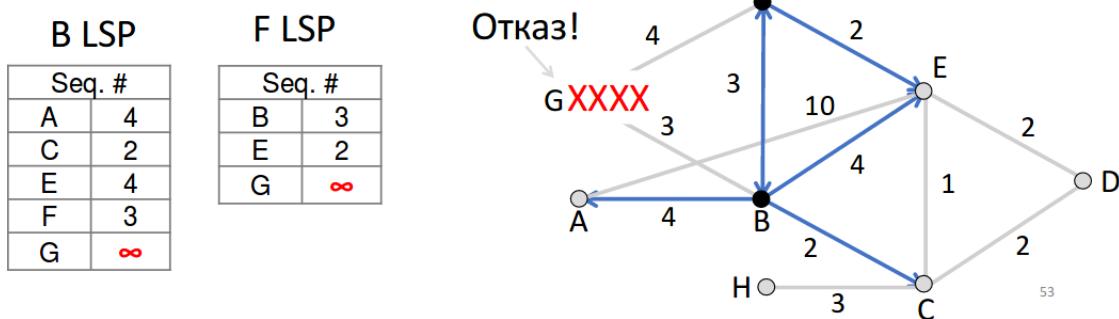
Svrha protokola za rutiranje je da se prilagode promenama. Kada se desi neka promena u topologiji, npr. neka komponenta je otkazala ili je neki novi ruter dodat želimo da ponovimo ovaj proces, tj. da ponovo prođemo kroz faze. Plaviće se ažurirana verzija LSP-a i ponovno se računaju putanje.

Kada neki čvor otkaže njegovi susedi primete da se nešto promenilo. Poslaće ažuriranu informaciju i svako ko je primi će ponoviti izračunavanje.

Da vidimo šta se dešava na slici.

Čvor G je otkazao, njegovi susedi B i F primećuju da ga više nema. S vremena na vreme šalju neke "Hello" poruke da provere da li "živ", ali kad ne bude bilo odgovora primetiće da ga više nema. Onda šalju ažurirane LSP. U novim LSP stoje isti susedi kao i ranije, samo je za G promenjena cena na beskonačno, što znači da ta veza ne radi. Kada se ta informacija raširi po svim čvorovima oni će ponovo izračunati puteve tako da nijedan put ne ide kroz G.

- Када неки чвор детектује локалну промену, он плavi са новим LSP
  - Нпр. суседи од G су открили промену, па шаљу LSP



- **Otkaz veze**

Čvorovi na krajevima pokvarene veze vrše plavljenje sa ažuriranim LSP. U LSP je postavljeno da je cena pokvarene veze beskonačno.

- **Otkaz čvora**

Svi susedi primete da veza ne radi. Ne znaju da li je u pitanju otkaz veze ili čvora, ali nije ni bitno, rade isto kao i u slučaju otkaza veze.

- **Pokvareni čvor ne može da ažurira sopstveni LSP**

Ali to nije ni bitno, on svakako ne radi!

- **Dodavanje čvora ili veze**

Susedni čvorovi će detektovati i ažurirati svoje LSP. Nakon plavljenja, ubrzo će svi znati novu topologiju.

## DV – LS poređenje

- **Konvergencija:**

Spora kod DV, najbolji putevi na daljini k tek u k-toj razmeni. A i videli smo da može doći i do problema brojanja do beskonačnosti. Brza kod LS, jer je plavljenje brzo.

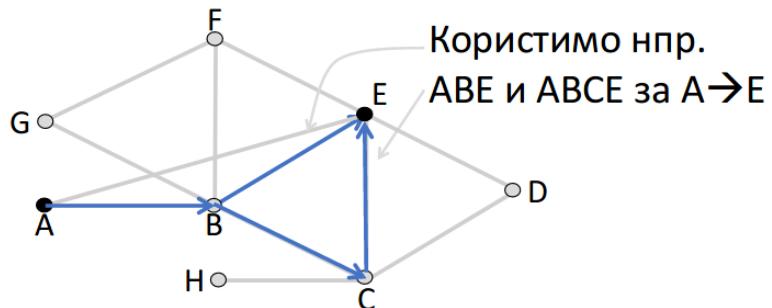
- **Skalabilnost:**

Odlična za DV, čvorovi računaju rešenje potproblema. Korektna kod LS, svaki čvor računa rešenje celog problema. Količina izračunavanja koje je potrebno izvršiti da bi se zapamtio ceo graf raste veoma brzo kako raste mreža. Može se nadomestiti zasad boljom opremom.

## 35. Višeciljno rutiranje sa najkraćim putevima (ECMP)

Dozvoljavamo višestruke puteve. Redudantnost poboljšava pouzdanost, ali pomaže i kod smanjenja zagušenja, pa poboljšava performanse.

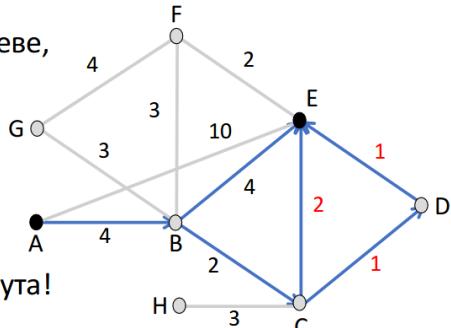
Na slici vidimo da od A do E možemo doći koristeći puteve ABE i ABCE.



### ECMP rutiranje (Equal-cost multipath routing)

To je nadogradnja modela sa najkraćim putevima. Zadržavamo sve najbolje puteve ne samo jedan. Ranije, ako bismo imali više puteva iste cene izabrali bismo jedan od njih nasumično. Na slici imamo primer nekih puteva koji imaju iste cene.

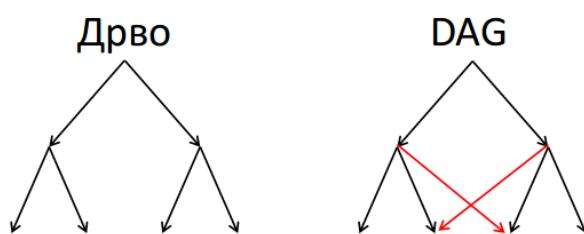
- Тип вишециљног рутирања
  - Задржавамо све најбоље путеве, а не само један
- Нпр.  $A \rightarrow E$ 
  - $ABE = 4 + 4 = 8$
  - $ABCE = 4 + 2 + 2 = 8$
  - $ABCDE = 4 + 2 + 1 + 1 = 8$
  - Користимо сва три најбоља пута!



### Topologija

Kod ECMP rutiranja, najkraći putevi od zadatog čvora ne formiraju drvo kao kod jednocijljnog. Formira se takozvani usmereni aciklički graf (DAG – directed acyclic graph). Dijkstrin algoritam, samo ubacujemo sve najbolje.

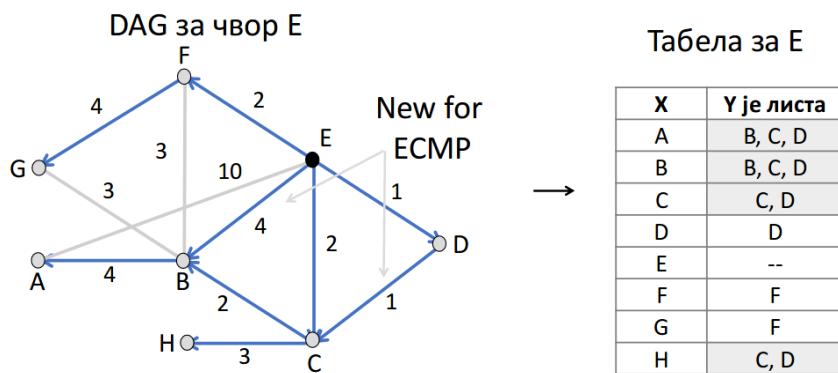
Na levoj slici imamo obično drvo, gde postoji samo jedan put da se dođe do određene destinacije. Na desnoj slici imamo više puteva do određene destinacije. Graf je aciklički tako da nemamo petlji. Svaki čvor nema više jedan hop, neko skup hopova kako do dođe do destinacije.



## ECMP

Sledeći primer pokazuje kako se određuje DAG za čvor E. Koristimo Dijkstrin algoritam, ali zadržavamo sve najbolje puteve. Kad dobijemo DAG odredićemo tabelu prosleđivanja.

Od E možemo doći do D sa cenom 1, do F sa cenom 2, do C možemo doći na dva načina – direktno sa cenom 2 ili preko D isto sa cenom 2, do B možemo doći direkto sa cenom 4 ili preko C isto sa cenom 4 ili preko D isto sa cenom 4, itd.



## ECMP prosleđivanje

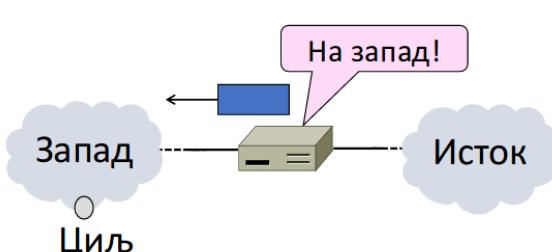
Nakon što je određena tabela prosleđivanja kako se prosleđuje:

- **Nasumičnim odabirom jedne od putanja?**  
Balansira opterećenje. Paketi mogu imati različito kašnjenje, loše za prenos u realnom vremenu. Na primer, želimo da pošaljemo pakete 1,2,3 i 4 od istog čvora. Recimo da 1,3 i 4 idu jednim – kraćim – putem, a 2 nekim drugim, dužim. 1, 3 i 4 će doći pre 2. To bi trebalo izbegavati.
- **Bolje je fiksirati izbor na osnovu izvora i cilja**  
Dakle, kontrolisano nasumično (nasumično, ali isto na nivou izvora i cilja). Opterećenje se i dalje balansira na taj način. Eliminiše se upotreba različitih putanja za pakete iz istih logičkih celina, npr. datoteka, videa itd.

## 36. Hijerarhijsko rutiranje

Rutiranje po svakom pojedinačnom (čvoru) ruteru se ne skalira dobro!

Zato čvorove možemo da grupišemo u regione. Potom unutar regiona vršimo specifičnije rutiranje.



## Rast broja rutera (IP prefiksa) na Internetu

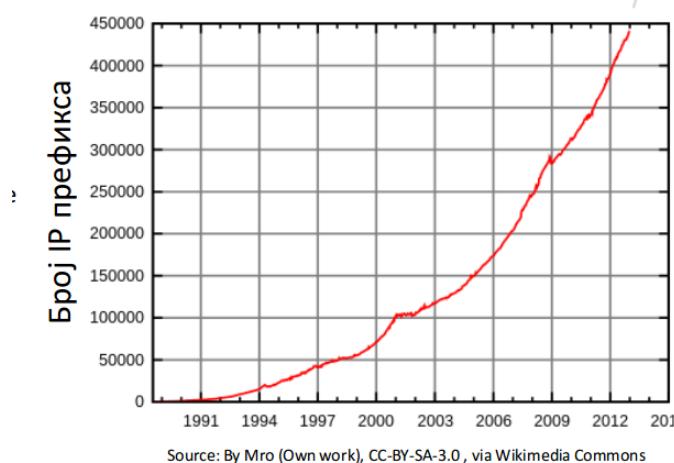
Nema smisla praviti unos u tabeli prosleđivanja za svaki ruter u svetu!

Napomena: ovde se misli na rutere koji se pojavljuju u tabelama rutiranja globalno, postoje i prikriveni ruteri za podmreže (poslednji slajdovi).

Rastu tabele prosleđivanja, raste i broj poruka, i samo izračunavanje pitanja raste.

Na slici vidimo otprilike eksponencijalni rast.

(IP префикса) на Интернету

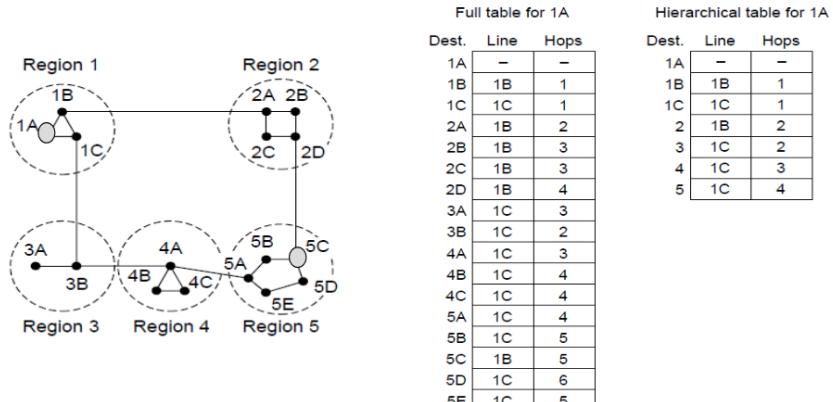


Source: By Mro (Own work), CC-BY-SA-3.0, via Wikimedia Commons

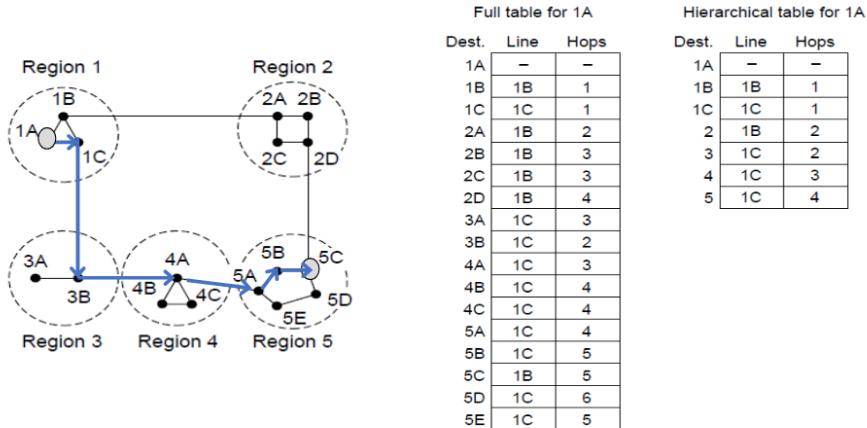
## Hijerarhijsko rutiranje

- **Uvodimo veće jedinice rutiranja, npr. region neke države**  
Za te veće jedinice vršimo rutiranje, posmatrajući ih kao da su pojedinačni čvorovi
- **Mogu da postoje i podregioni unutar regiona**  
I za njih primenjujemo isti pristup, npr. ISP mreža
- **Konačnu putanju dobijamo tako što:**
  - Najpre rutiramo paket u najmanjem regionu, npr. u okviru ISP mreže
  - Potom po izlasku iz ISP mreže, koristi se rutiranje na nivou države
  - Potom se ponovo ulazi u npr. ciljnu ISP mrežu

На крајне левој slici imamo mrežu sa 5 različitih regiona. U svakom regionu imamo skup povezanih čvorova i regioni su međusobno povezani. Svaki čvor u ovoj mreži ima tabelu prosleđivanja. Na slici u sredini vidimo tabelu prosleđivanja za čvor 1A. To je velika tabela, jer vidimo sve ostale čvorove u njoj. Kolone su destinacija, veza i broj hopova. Na desnoj slici imamo hijerarhijsku tabelu za 1A. Ta tabela ima ulaze za čvorove u istom regionu – 1A, 1B i 1C. Ostali ulazi su sažeti. U punoj tabeli smo imali 4 ulaza za region 2, sad imamo samo jedan – zanima nas samo kako da dođemo do regiona 2. Slično je i sa regionima 3, 4 i 5. Sad imamo mnogo manju tabelu.



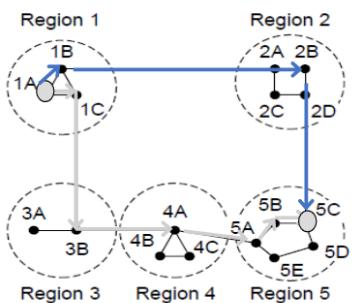
Recimo da želimo da dođemo od 1A do 5C. U tabeli nam samo kaže da odemo do 1C, nema u tabeli direktnе veze sa 5C. Mi nemamo ovde prikazanu tabelu za 1C i ostale čvorove na putanji, ali prepostavljamo da će putanja izgledati kao na slici.



Ovaj gornji primer nam pokazuje kakvu kaznu možemo da platimo ako koristimo hijerarhijsko rutiranje. Za većinu puteva, kada koristimo hijerarhijsko rutiranje, razdaljina koju koristimo u hopovima je ista čak i da nismo koristili hijerarhijsko rutiranje, samo smo dobili manju tabelu. Ali za neke putanje može da se desi da smo izabrali duže putanje umesto najkraćih mogućih.

U velikoj tabeli nam kaže da ako želimo da od 1A dođemo do 5C to možemo postići sa 5 hopovi idući preko 1B. Ali u hijerarhijskoj tabeli nas od čvora 1A do regiona 5 vodi preko druge putanje – preko 1C. To je zbog toga što smo zaboravili na sve detalje u regionu 5 i jednostavno smo želeli da dođemo do regiona 5 što je brže moguće.

- Не даје увек најкраћу путању, али је то компромис зарад ефикасности



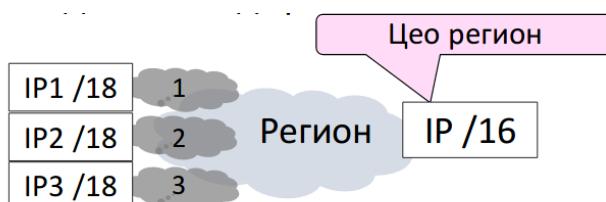
Full table for 1A		
Dest.	Line	Hops
1A	-	-
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

Hierarchical table for 1A		
Dest.	Line	Hops
1A	-	-
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

1C је најбоља генерално ако шаљемо у регион 5.  
Али конкретно, не мора бити најбоља за све чворове унутар њега, нпр. за 5C

69

Rutiranje po regionima je i dalje teško za izračunavanje. Uvodimo dodatni podnivo u vidu IP prefiksa. Podela na podmreže i njihovo kasnije sažimanje. Na slici imamo 3 IP prefiksa i njih spajamo u jedan veći npr. /16 prefiks koji će predstavljati ceo region.

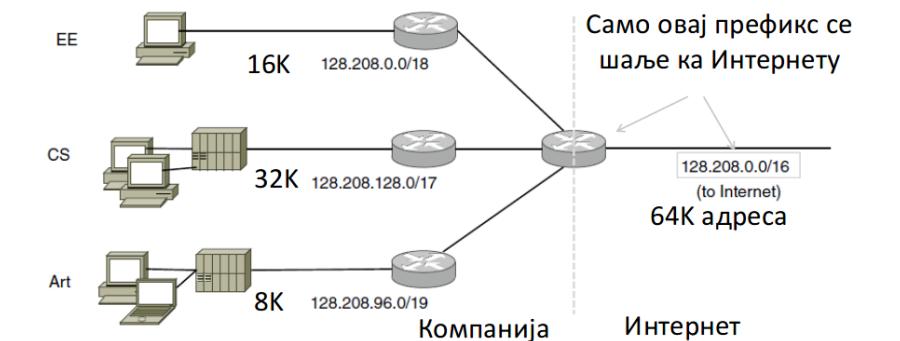


Imamo dva različita slučaja korišćenja. Oba smanjuju veličinu tabele prosleđivanja.

## Podmreže

Interna podela IP prefiksa npr. unutar kompanije. IP prefiksi (EE, CS, Art) ruteru nisu vidljivi spolja. Ti više specifičniji prefiksi se zovu podmreže.

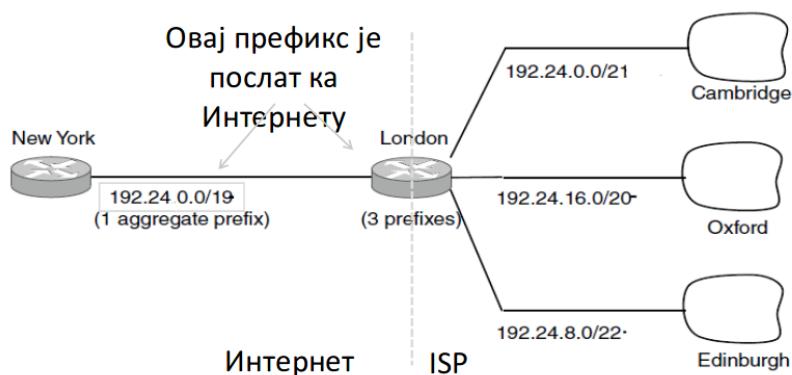
Na slici, vidimo da kompanija ima jedan IP prefiks za sve svoje računare – 128.208.0.0/16. Znači da kompanija ima 64 000 različitih adresa. Interno, kompanija možda želi da ima različite strukture. Imamo EE, CS i Art odeljenja. Interno /16 se može podeliti na /17, /18 i /19. /18 kreće od dna raspona adresa, /17 negde od sredine, a /19 je negde iznad /18.



## Sažimanje

Moguće je i spoljne sažimanje nepovezanih ustanova. Ovo obično rade ISP. Više specifične mreže spajamo u jednu manje specifičnu.

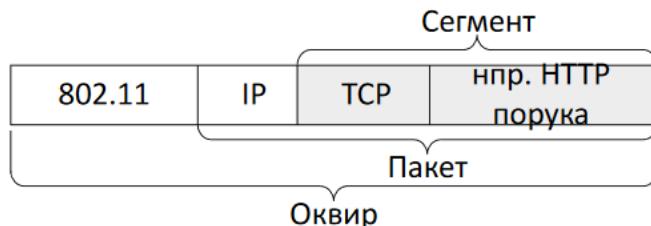
Imamo tri univerziteta – Kembirdž, Oksford i Edinburg. imamo /21, /20 i /22. Ovi prefiksi su izabrani iz adresnog prostora koji je srođan. Zapravo svi oni formiraju veći blok. Možemo ih sažeti u /19 koji je dovoljno veliki da sadrži sve te adrese.



### **37. Transportni sloj, uloga, tipovi servisa I njihovo poredjenje.**

Transportni sloj je srz hijerarhije protokola. Njegov zadatak je da obezbedi pouzdan, isplativ prenos podataka, bez obzira na fizicku mrezu ili mreze koje se trenutno nalaze izmedju izvorisnog i odredisnog racunara.

Jedinica informacije u transportnom sloju se zove segment. Segment sadrzi informacije o kontoli transporta (npr. TCP zaglavlje) i podatke aplikacije koji se prenose preko mreze. Segment je obavljen paketom o kome brine sloj mreze.



Transportni sloj moze da pruzi pouzdan i nepouzdan transport. Pouzdan transport podrazumeva da ce poslat paket stici do primalaca, dok nepouzdani transtport podrazumeva da se paket moze izgubiti u mrezi. Podaci se mogu izgubiti i ako se koristi puzdan transport, ali transportni sloj resava ovaj problem tako da se ne vidi da je paket izgubljen.

Internet pruza dve vrste servisa aplikacija na aplikativnom sloju. TCP je protokol koji implementira strim servis. To je pouzdan strim bajtova izmedju aplikacija. UDP je protokol koji implementira drugi tip servisa, a to je datagram servis kojim se salju poruke. Ovaj servis spada u nepouzdan transport.

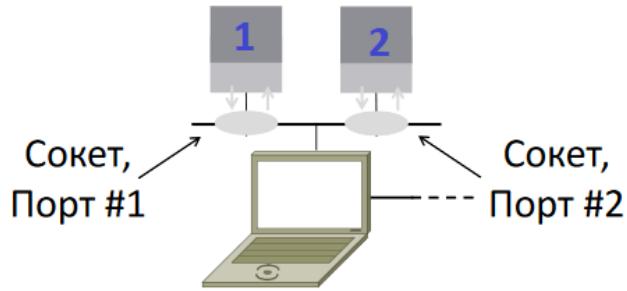
TCP je veoma razvijen mehanizam, dok UDP prakticno koristi datagram iz mreznog sloja.

<b>TCP (Strimovi)</b>	<b>UDP (Datagrami)</b>
Mora da se uspostavi veza	Salju se datagrami
Bajtovi se dostavljaju pouzданo, jednom i u istom redosledu u kojem su poslati	Poruke se mogu izgubiti, pomesati, duplirati
Moze da se posalje proizvoljan broj bajtova	Ogranicena duzina poruke
Kontrola toka se prilagodjava posiljaocu i primaocu	Salje se bez obzira na stanje primaoca
Kontrola zagusenja se prilagodjava stanju mreze	Salje se bez obzira na stanje mreze

### 38. Socket API, primer jednostavnog klijent-servera (pseudokod), portovi.

Socket API je interfejs izmedju transportnog I aplikativnog sloja. Predstavlja apstrakciju za upotrebu mreznih usluga. Cesto se kaze I "Mrejni API", ali je zapravo u pitanju upotreba transportnih servisa, a ne mreznih. Ovaj interfejs podrzava obe vrste servisa transportnog sloje, I strimove I datagrame.

Portovi predstavljaju klucni atribut za adresiranje. Pomocu portova aplikacija koristi transportni sloj. Razliciti portovi omogucavaju da vise aplikacija koristi isti transportni sloj. Ovo znaci da soketi omogucavaju aplikacijama da se povezuju na lokalnu mrezu putem razlicitih portova.



Isti API se koristi I za tokove I za datagrame.

Операција	Значење
SOCKET	Креира крајњу комуникациону тачку
BIND	Придружује сокет локалној адреси и порту
LISTEN	Најављује спремност за прихватање долазних захтева за везу
ACCEPT	Пасивна успостава везе са тачком која шаље захтев
CONNECT	Активно покушавање за успоставом везе
SEND(TO)	Слање података преко сокета
RECEIVE(FROM)	Примање података преко сокета
CLOSE	Гашење сокета

Користе се само код Токова  
Варијанте са „to/from“ само код датаграма

Kod datagrama moramo specifirati kome saljemo i od koga primamo podatke. Takodje mozemo slati podatke razlicitim primaocima I primati podatke od razlicitih posiljalaca pomocu istog soketa.

Pseudokod klijenta:

```
s = SOCKET() - pravi novi soket
connection = s.CONNECT(host) – blokira klijenta I aktivno zapocinje proces povezivanja
write(connection, buffer)
read(connection, buffer)
connection.CLOSE()
```

Pseudokod servera:

```
s = SOCKET() - pravi novi soket  
s.BIND(port) - dodeljuje soketu mreznu adresu  
s.LISTEN() - stavlja u red za cekanje u slucaju da vise klijenata zeli da se poveze, ne blokira server  
connection = s.ACCEPT() - blokira server dok ne stigne poziv  
read(connection, buffer)  
write(stdin)  
connection.CLOSE()
```

Aplikacioni procesi se identifikuju uredjenom trojkom - IP adresa, protokol (UDP, TCP), port. Portovi su su 16-bitni pozitivni celi brojevi koji predstavljaju lokalne "postanske sanducice" koje procesi uzimaju na koriscenje kako bi koristile servise transportnog sloja.

Serveri se obicno vezuju za "opste-poznate portove", a to su portovi sa vrednostima manjim od 1024.

Klijenti mogu da koriste bili koji port. Bira ih operativni sistem i oni se koriste privremeno. Oni ne moraju biti opste-poznati, jer ih niko ne "cilja".

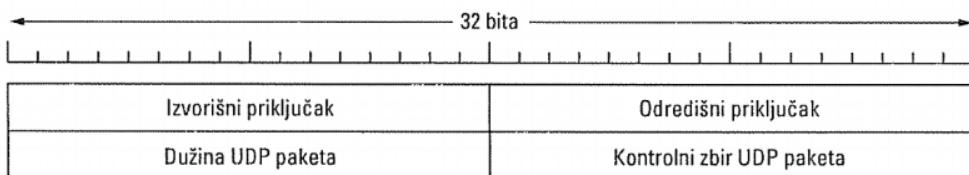
Neki opste-poznati portovi:

Порт	Протокол	Намена
20, 21	FTP	Пренос датотека
22	SSH	Удаљени приступ
25	SMTP	Слање и примање електронске поште
80	HTTP	Приступ WWW
110	POP-3	Примање електронске поште (клијенти)
143	IMAP	Примање електронске поште (клијенти)
443	HTTPS	Сигурни HTTP
543	RTSP	Пренос токова у реалном времену
631	IPP	Дељење штампача

### 39. UDP - User Datagram Protocol.

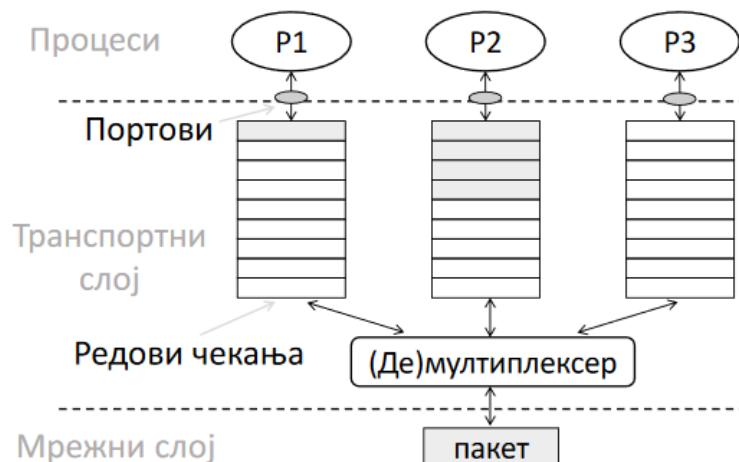
UDP je protokol za rad bez uspostavljanja direktne veze. Njega koriste programi kojima nije preterano bitna pouznadost i i koji ne zele strimove, tj. baziraju se na porukama (npr. voice-over-IP – internet telefonija, DNS, RPC, DHCP). Ovaj protokol omogucava aplikacijama da salju kapsilirane IP datagrame za koje ne moraju prethodno da uspostavljaju vezu. UDP prenosi segmente koji se sastoje od 8-bajtnog zaglavlja i korisnickih podataka. Velicina datagrama je do 64KB. Zaglavljje koristi port kako bi prepoznao proces. Glavna prednost protokola UDP nad osnovnim IP

protokolima lezi u tome sto on paketima dodaje izvorisni i odredisni prikljucak (port). Bez tih polja u zaglavlju, transportni sloj ne bi znao sta da radi sa paketima. Ovako ih on isporucuje na pravu adresu.



UDP ne upravlja tokom, ne kontrolise greske i ne salje ponovo pogresno primljene segmente. Sve to prepusta korisnickim procesima. On predstavlja interfejs ka IP protokolu i dodatno demultiplexira procese koji istovremeno koriste isti port. Za aplikacije koje zahtevaju preciznu kontrolu toka, gresaka ili vremenskog usklajivanja, protokol UDP radi tacno ono sto mu nalozi "visa instanca".

Kada segment pristigne od posiljaoca, cuva se u baferu u transportnom sloju sve dok aplikacija ne pozove `RECVFROM`. Takodje, ako aplikacija zeli da posalje segment pozivom funkcije `SENDTO`, segment se prebacuje u bafer.



Soketi u slucaju datagrama:



\* = блокирајући позив

#### 40. Uspostava I prekid veze na transportnom sloju (uopsteno).

Prije bili kakvog slanja ili primanja posiljalac i primalac moraju biti svesni uspostave veze. Da bi se ovo desilo moraju da se slože oko skupa parametara kao što je npr. maksimalna veličina segmenta.

Uspostava veze podrazumeva podešavanje stanja krajnjih čvorova. Strane takođe treba da usaglase pocetne brojeve segmenata kako više dalje mogao implementirati protokol kliznih prozora.

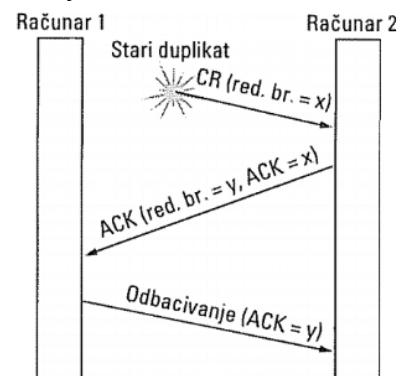
Trofazno rukovanje je način uspostave veze koji se koristi u TCP.

Posiljalac (klijent) biranje redni broj (SEQ)  $x$  i salje ga sa sinhronizacionim segmentom (SYN). Primalac (server) odgovara potvrdom kojom prihvata  $x$  i najavljuje svoj pocetni redni broj  $y$  (ACK), ovaj odgovor je u stvari sinhronizacioni segment koji sadrži vrednosti  $x+1$  i  $y$ . Na kraju, klijent potvrđuje pocetni redni broj  $y$  slanjem još jednog sinhronizacionog segmenta koji sadrži brojeve  $x+1$  i  $y+1$ . Sinhronizacioni segmenti se ponovo salju ako se izgube.



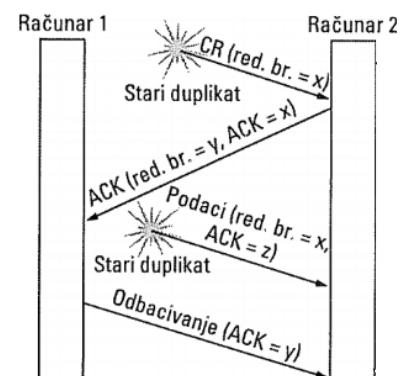
Slučaju kada postoji zakasneli duplikat (kasnjenje I retransmisija):

Server prima zakasneli duplikat bez znanja klijenta. Server reaguje tako što klijentu salje potvrdu kojom, u stvari, on traži potvrdu da klijent zaista želi da uspostavi novu vezu. Kada klijent odbije pokušaj servera da uspostavi vezu, server shvata da ga je prevario zakasneli duplikat i odustaje od povezivanja. Na taj način zakasneli duplikat ne pravi nikakvu stetu.

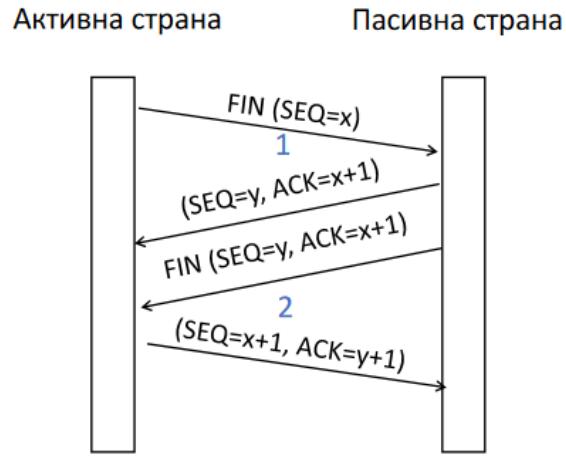


Slučaj kada postoje dva zakasnela duplikata (kasnjenje I retransmisija):

Kao u prethodnom primeru, server dobija zakasneli duplikat i odgovara na njega slanjem rednog broja  $y$ . Kada serveru stigne drugi zakasneli duplikat, on zna da je to stari duplikat iz činjenice da je za pocetni redni broj prihvatio  $z+1$  umesto  $y+1$ .



Lakše je raskinuti vezu nego je uspostaviti. U TCP vrza se smatra sistemom od dve paralelne jednosmerne veze od kojih se svaka mora zasebno raskinuti. Za to nam je potrebno dva koraka. Klijent salje serveru FIN (finish) segment sa sinhronizacionim brojem  $x$ . Server prima  $x$  i salje odgovor koji sadrži njegov redni broj  $y$  i  $x+1$ . Nakon toga server salje još jedan segment, FIN segment sa istim rednim brojevima. Na to klijent odgovara sa segmentom koji sadrži  $x+1$  i  $y+1$ . Naravno ne mora samo klijent zatruditi prekid konekcije, to može učiniti i server. Nakon slanja FIN signala i dobijanjem odgovora za isti, svaki računar gasi svoju stranu.



#### 41. Protokoli kliznih prozora na transportnom sloju.

U skoku veze, ovi protokoli su se odnosili na prenos podataka kroz kabl između dva susedna cvora. U transportnom sloju, ovi protokoli omogućavaju prenos između krajnjih tacaka bilo gde na Internetu.

Povećana pouzdanost na nizim nivoima doprinosi efikasnost na visim nivoima, ali nije nuzno imati te mehanizme na nizim nivoima. U ekstremnom slučaju, moglo bi sve da se radi na transportnom nivou npr. kontrola toka, provjeru gresaka, ... ali ovo bi bilo manje efikasno.

Postoji dosta varijacija ovih protokola, u zavisnosti od baferisanja, potvrđivanja poruka i retransmisija. "Vrati se N" je jednostavna verzija koja može biti neefikasna. "Selektivno ponavljanje" je složenija verzija, ali i efikasnija. Zajednicko za sve verzije je da od aplikativnog sloja dobijaju segmente po redu i aplikativnom sloju salju segmente po redu.

Posiljalac:

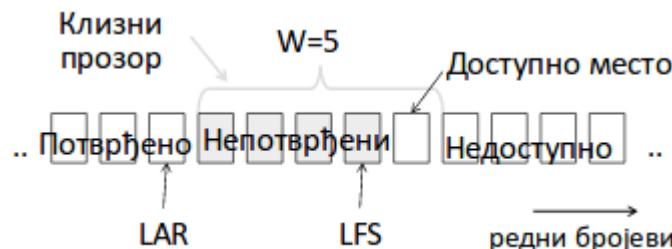
Posiljalac baferise najviše  $W$  segmenata dok ne stignu potvrde za njih.

LFS – poslednji poslat segment

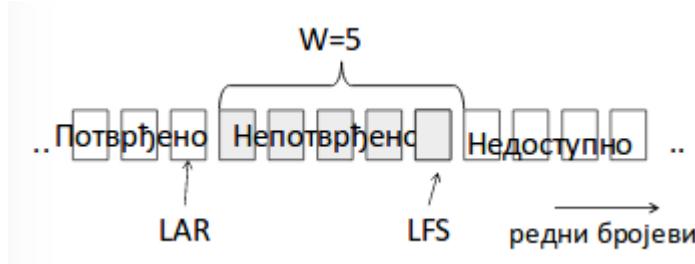
LAR – poslednji potvrđeni segment pre koga su svi potvrđeni

Salje se dok je  $LFS - LAR \leq W$

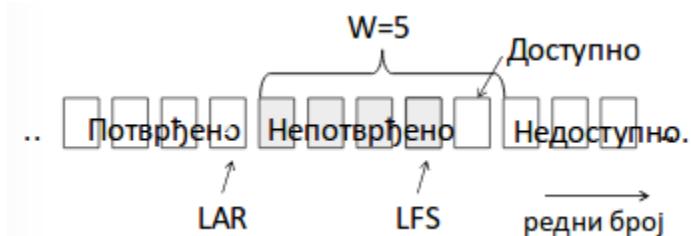
Primer:



Transportni sloj prihvata od aplikativnog još jedan segment, a transportni salje jer je LFS-LAR = 5.



U međuvremenu stize potrda za naredni segment. Prozor se pomera, I jedno mesto u baferu se oslobadja. Ponovo može da se salje jedan LFS-LAR = 4.



Primalac – varijanta “Vrati se N”:

Primalac ima bafer velicine 1 i cuva redni broj poslednjeg segmenta prosledjenog aplikativnom sloju – LAS. Nakon primanja segmenta, ako je redni broj LAS+1, onda ga prihvati, prosledi aplikativnom sloju, azuriraj LAS=LAS+1 i posalji potvrdu, a inace odbaci.

Primalac – varijanta “Selektivno ponavljanje”:

Primalac prosledjuje aplikativnom sloju po redu, a baferise segmente i ako nisu po redu (bafer velicine W). Putem ACK segmenta (za potvrdu) primalac potvrdjuje najvise uredjeni segment, a dodatno salje informaciju o segmentu koji nisu po redu. TCP koristi ovaj pristup. Primalac baferise W segmenata, odrzava stanje promenljive LAS, prihvata ako je iz opsega [LAS+1, LAS+W] i pritom baferise segmente iz opsega [LAS+1, LAS+W], prosledjuje aplikativnom sloju ako stigne segment sa brojem LAS+1, a pritom azurira LAS = LAS+1, i na kraju salje poruku.

“Vrati se N” posiljalac ima jedan timer, kada istekne, ponovo salje sve baferisane segmente pocev od LAR+1

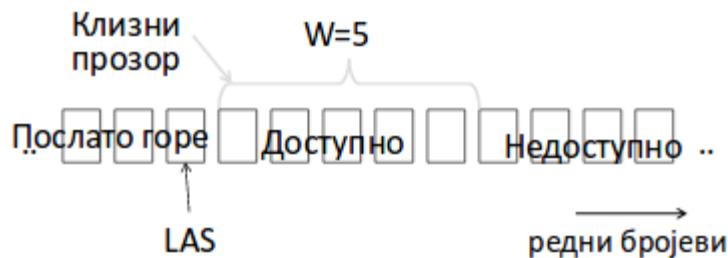
“Selektivno ponasanje” posiljalac ima tajmer za svaki nepotvrdjeni segment. Po isteku salje ponovo. U proseku radi manje retransmisija.

## 42. Kontrola toka podataka na transportnom sloju.

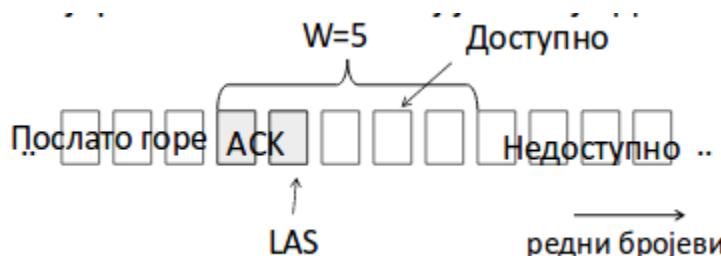
Kontrola toka je jedan od glavnih problema. Stvar je u tome da se na vezi moraju primeniti klizni prozori ili slicna sema da neumerno brz posiljalac ne bi zatrpaо sporijeg primaoca poruka.

Primer:

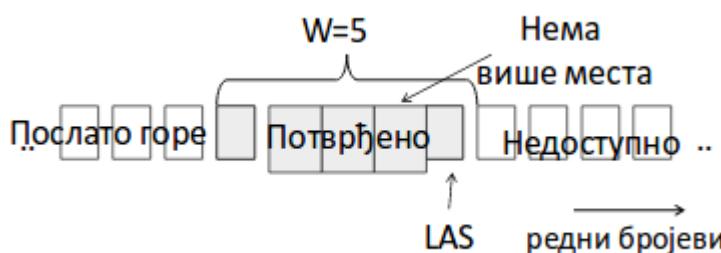
Neka primalac ima bafer velicine  $W$ . Inicijalno bafer je prazan, a deo pre toga je poslat aplikativnom sloju. Aplikacija skida sa bafera redom podatke pozivom funkcije `recv()` koja je blokirajuća.



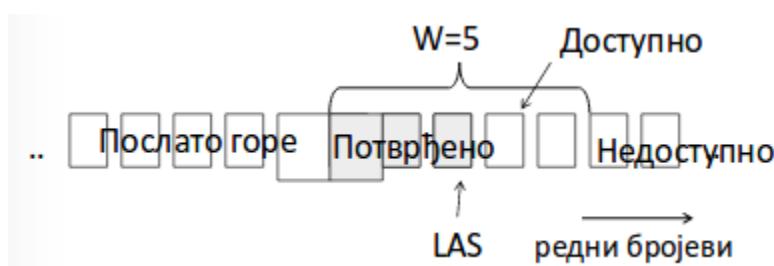
Neka nakon toga pristignu dva segmenta, ali program I dalje ne poziva funkciju `recv()` (spor je, možda zato što je zauzet). Tada LAS raste.



Kada stignu naredni segmenti, popunjava se bafer i nakon toga nije više moguce primati, sve dok aplikativni sloj ne prihvati segmente. Azurira se LAS.

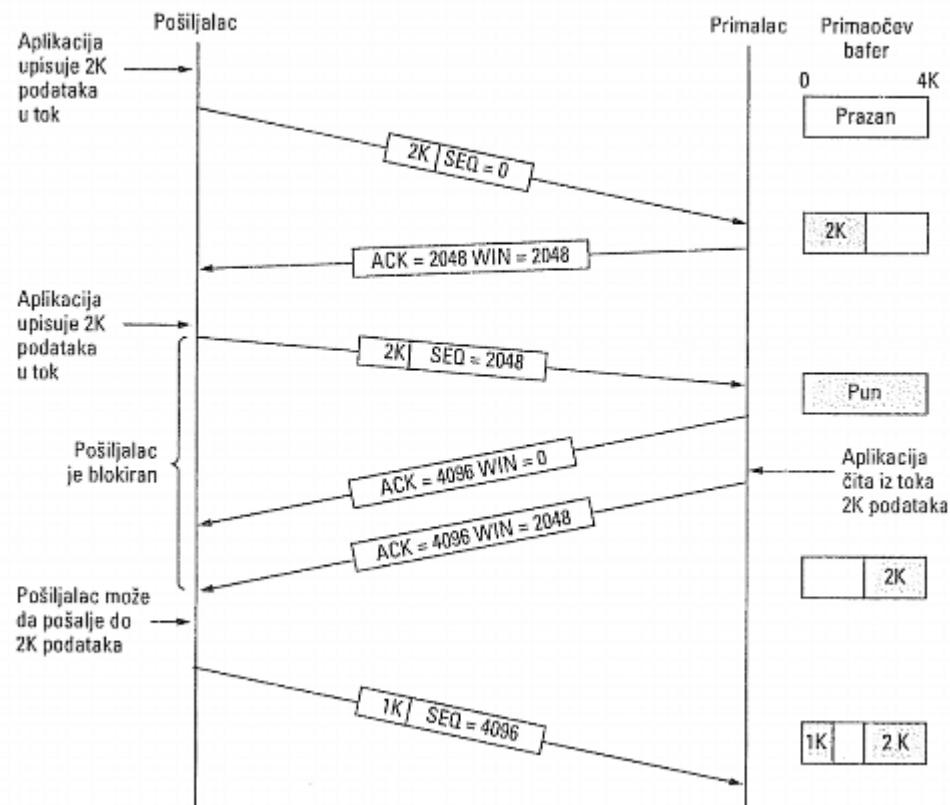


Nakon toga aplikativni sloj konacno prihvata dva segmenta i prozor se pomera.



U ovom primeru srecem nakon popunjavanja bafera nije poslato vise segmenata i nije doslo do gubitka. Ovaj problem se resava tako sto se posiljaocu jednostavno kaze koliko je mesta u baferu slobodno. Primalac salje broj WIN koji je jednak broju slobodnih mesta u baferu. Posiljalac koristi WIN kao efektivnu informaciju o velicini prozora.

Primer resenja:



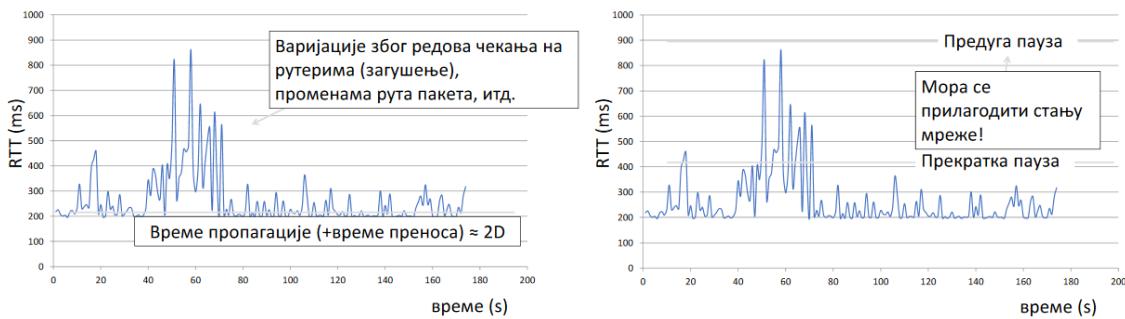
TCP funkcione ovako. SEQ predstavlja poziciju u baferu od koje posiljalac trazi da se upise segment. ACK predstavlja poziciju od koje pocinje prazan prostor u baferu. WIN je velicina praznog prostora. Bafer se popunjava ciklicno. Mora da vazi  $SEQ + \text{duzina segmenta} \leq ACK + WIN$ .

### 43. Retransmisijske i prilagodljive pauze (tajmauti) na transportnom sloju

Strategija za detekciju gubitaka kod klizajucih prozora je tajmaut. Postavlja se tajmer kada je segment poslat. On se deaktivira kada se dobije potvrda. Ali ako tajmer istekne, segment se verovatno izgubio i vrsi se retransmisijska pauza.

Glavni problem je adaptirati tajmer. On mora biti dobro ocenjen kako ne bi dolazio do nezeljenih situacija. Prevelike pauze mogu uzrokovati da se ne koristi pun potencijal mreze, gubi se vreme. Premale pauze izazivaju retransmisijsku pauzu i ako to nije potrebno.

Ali kako podesiti tajmer? To je lako učiniti za LAN zato što je kabl kratak i rute su poznate, te se može predvideti RTT (vreme propagacije + vreme prenosa). Veoma je tesko za Internet zato što ima sirok opseg i dosta orimenljiv RTT.

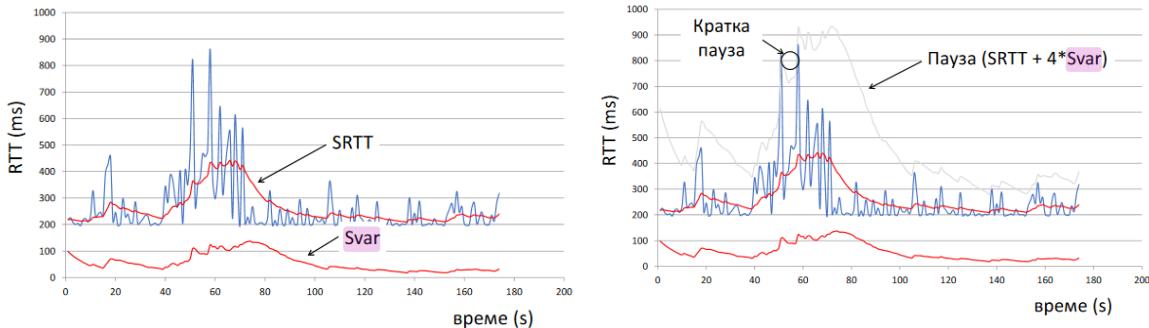


Ovaj problem se resava pomocu prilagodljive pauze. Ideja je da se ocenti kratkorocni RTT i varijacije u RTT i onda to iskoristiti za postavljanje trajanja pauze. Formula zasnovana na pomerajucim procesima:

$$SRTT(n+1) = 0.9 * SRTT(n) + 0.1 * RTT(n+1)$$

$$Svar(n+1) = 0.9 * Svar(n) + 0.1 * |RTT(n+1) - SRTT(n+1)|$$

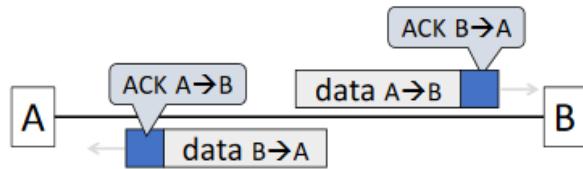
Pauza treba uvek da bude iznad ocene RTT, pa se postavlja na  $SRTT(n) + 4*Svar(n)$ . Sto je veća varijansa, manje smo sigurni, pa je i gornja granica više udaljena.



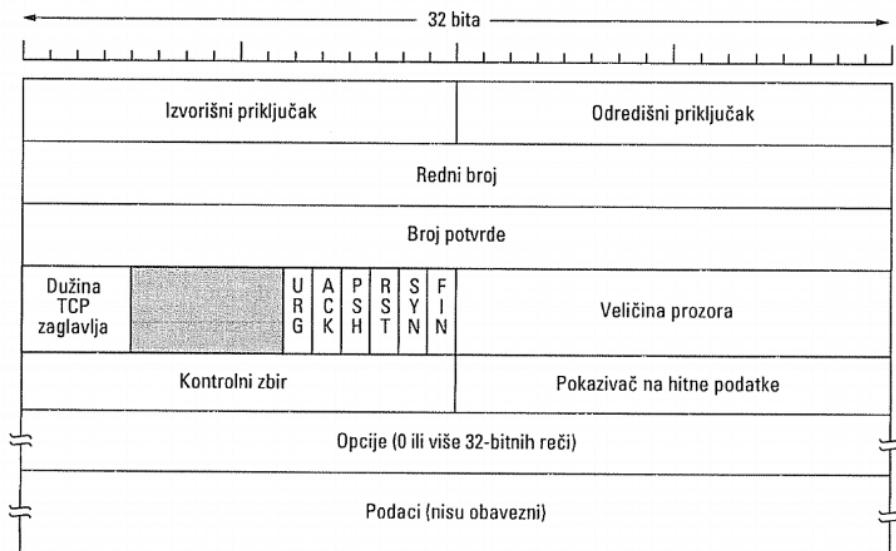
#### 44. TCP, svojstva, zaglavlje, realizacija kliznih prozora, uspostava I prekid veze (specifino)

TCP je protokol koji se koristi za prenos velike većine podataka. On implementira pouzdan strim servis koji koristi sokete. Strim servis je implementiran koriscenjem veza. Klizni prozori se koriste zarad pouzdanosti i to sa prilagodljivim pauzama. Takođe TCP takođe sadrži kontrolu toka kako bi pomogao sporim primaocima.

Šta znači pouzdan strim (tok) bajtova? Segmenti se mogu kretati neuredjeno i nepouzdano kroz mrežu i ostale nize slojeve, međutim, transportni sloj ih uređuje, proverava, i aplikativnom sloju salje pouzданo i po redu. Slanje i primanje podataka se odvija u oba smera. Kontrolne informacije (npr. ACK) se često salju kao delovi dolaznih segmenata za podatke iz drugog smera.

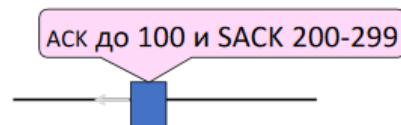


Na sledecoj slici je prikazana organizacija TCP segmenta.

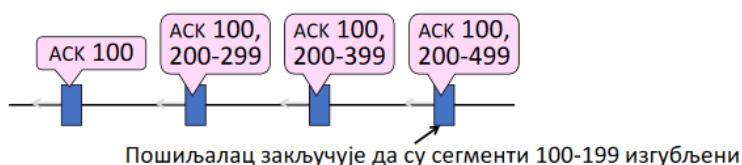


Izvorisni i odredisni port (prikljucak) identificuju krajnje tacke veze, tj. programe. U zaglavljima svaki od portova zauzima 16 bita. Sledeca dva elementa u zaglavljima sluze za implementaciju klizajuceg prozora, a to su SEQ I ACK brojevi.

Primalac: Kumulativni ACK govori koji je sledeci ocekivani bajt (LAS+1). Selektivni ACK-ovi (SACK) koriste se zarad optimizacije i daju nagovestaj o stanju bafera (listanje do tri opsega primljenih bajtova).



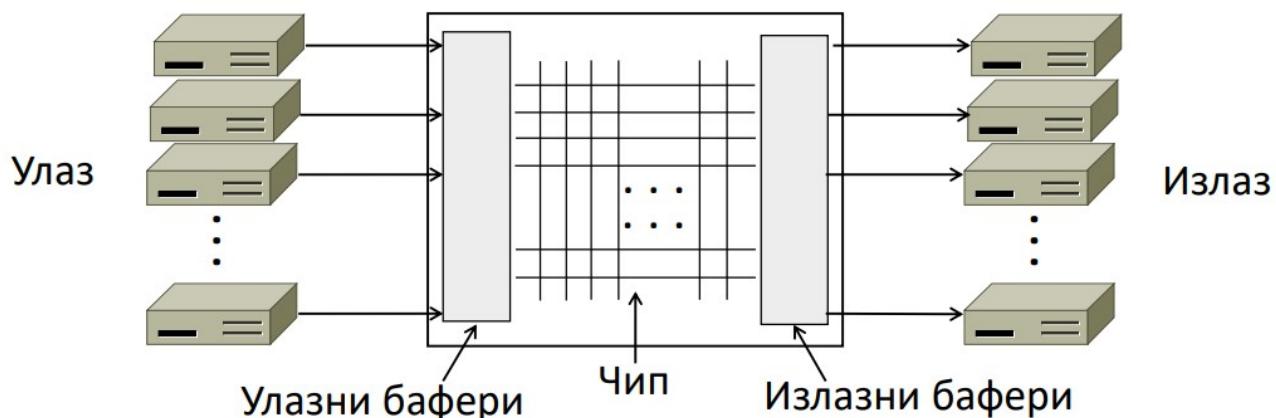
Posiljalac: Koristi prilagodljivu pauzu za retransmisiju segmenata koji pocinju od LAS+1. Koristi heuristiku kako bi brze zaključio koji segmenti su izgubljeni i time izbegao istek pauze.



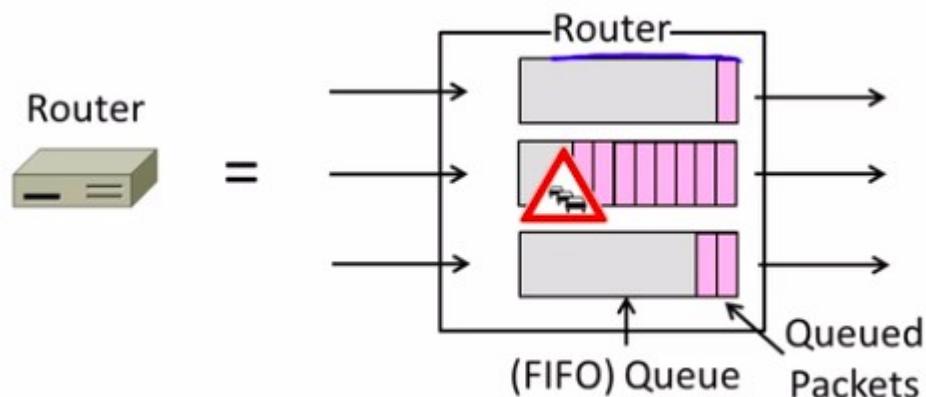
Heuristika – tri duplikata ACK-a implicira gubitak, obzirom da tri duplikata imaju iste ACK vrednosti, a razlicite SACK vrednosti, posiljalac zaključuje da su neki segmenti izgubljeni.

#### **45. Zagušenje na transportnom sloju, opis problema I mehanizam za rešavanje AIMD**

Zagušenje se može posmatrati kao gužva na mreži. Pošto ruteri i svičevi imaju internu baferisanje kada nekoliko ulaza hoće da ide u jedan izlaz i ona je obično organizovana kao FIFO.



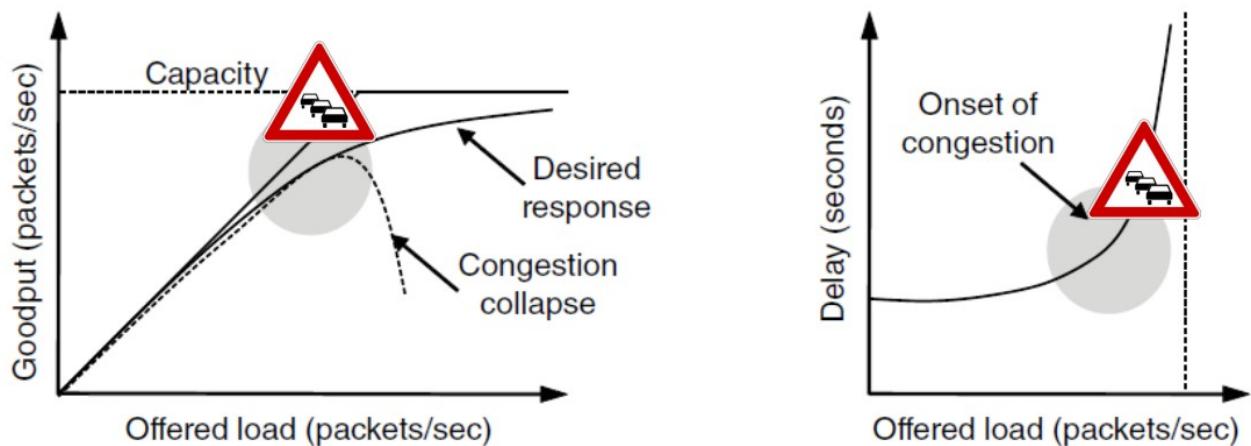
Na slici iznad možemo videti kako se saobraćaj odvija preko rutera ili sviča. Posmatraćemo situaciju kada bi na primer tri ulaza htela da idu u jedan izlaz, tu bi se desilo zagušenje.



Svaki izlaz ima red u koji se stavljuju ulazi koji žele određeni izlaz. Vidi se na srednjem izlazu da mnogo ulaza hoće da ga koristi i da u redu postoji mnogo ulaza gde se stvara zagušenje.

Redovi pomažu da se reši situacija gde izlazi ne mogu da se sprovedu brzinom kojom stižu ulazi.

Međutim ovi baferi nisu neograničena memorija, ako se određeni bafer u dužem vremenskom periodu puni, može doći do overflowa i da počnemo da gubimo pakete.



Na slici iznad, na prvom grafiku možemo posmatrati odnos povećanja broja paketa po sekundi koji prolaze mrežom (x osa) i `goodput`a (to je poželjan/željeni izlaz paketa po sekundi). Prva funkcija sa leve strane koja je dijagonalna pa vodoravna kad se postigne kapacitet mreže je idealna funkcija koju bismo želeli da ostvarimo, tj da povećanjem broja paketa koji mogu da se pošalju mrežom da se povača i broj željenih izlaza. Druga kriva na slici bi bila neka kriva koja bi bila realističnija da se desi jer ne možemo računati na optimalan ishod. Međutim, približavajući se granici kapaciteta propusnosti mreže, dešava se da povećanjem opterećenja na mreži drastično opada goodput – na primer šaljemo 7 puta paket koji se redovno gubi i samo jednom se pošalje na destinaciju. Ovakve situacije se nazivaju *kolaps* kada mreža prestane da radi kako treba usled velikog opterećenja.

Osenčena zona na grafiku predstavlja zonu u kojoj bi mogle da se dese ovakve situacije, tj zonu zagušenja.

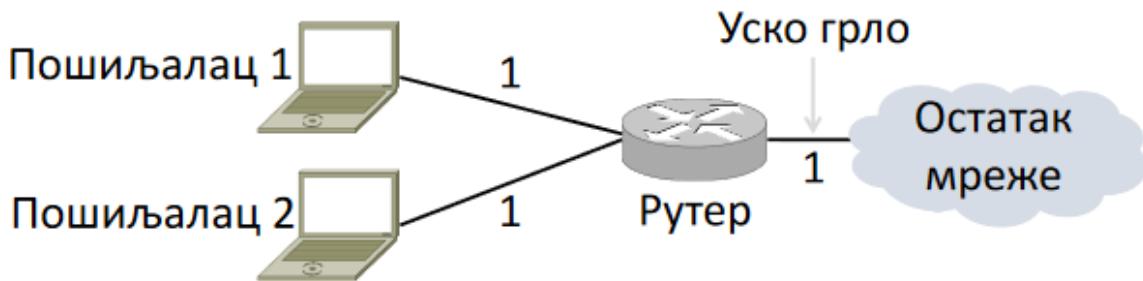
Na drugom grafiku možemo posmatrati odnos kašnjenja i broja paketa po sekundi koji se šalju mrežom. Isprekidana linija predstavlja maksimum broja paketa po sekundi koji mreža može da propagira. Idealno kašnjenje bi bilo konstantno, međutim približavanjem broja paketa po sekundi maksimumu, dešava se da se mnogo paketa šalje u FIFO redove i samim tim se povećava kašnjenje. Siva zona takođe predstavlja zonu zagušenja.

Dakle zagušenje se dešava kada počnu da se prepunjavaju baferi u svičevima i ruterima. Gubljenje podataka i kašnjenje u ovakvim situacijama drastično raste.

Da bismo pokušali da spričimo zagušenja, dodelićemo kapaciteta protoka pošiljaocima tako da skoro ceo bude upotrebljen bez zagušenja i da svi pošiljaoci dobiju razuman deo protoka.

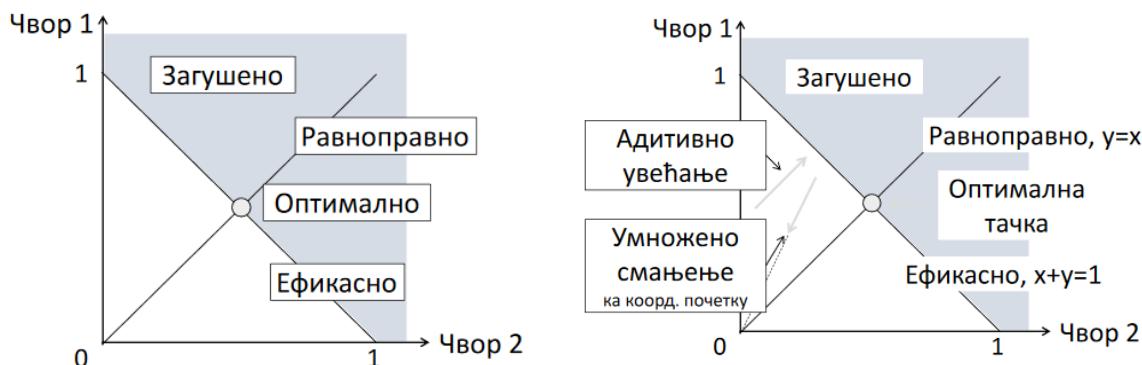
Na rešavanju zagušenja, mrežni i transportni sloj moraju da rade zajedno gde će mrežni sloj detektovati zagušenje i transportni redukovati opterećenje mreže. Znači pošiljaoci će prilagođavati odlazni saobraćaj u zavisnosti od toga šta detektuju na mreži.

AIMD (Additive Increase Multiplicative Decrease) je kontrolni mehanizam koji omogućava dostizanje dobre alokacije. Posiljaoci aditivno povećavaju brzinu slanja podataka dok mreža ne postane zagusena. Nakon toga je umnozeno smanjuju kada uoce zagusenje. TCP koristi ovo u nekoj formi.



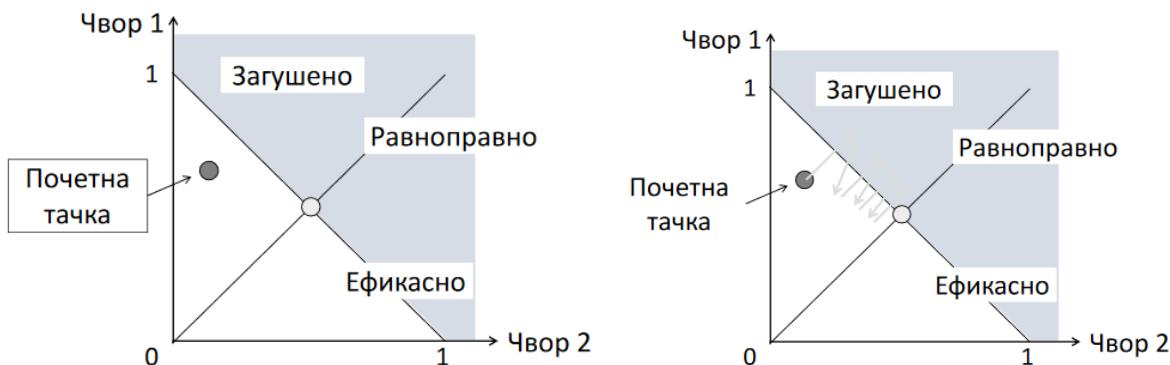
Na slici vidimo da podaci posiljaoca 1 i 2 prolaze kroz istu tacku zagusenja "usko grlo". Medjutim, posiljaoci ne mogu da komunicijaru. Ruter je taj koji moze da signalizira. On salje binarno 0/1 ako (ne)postoji zagusenje.

Svaka alokacija je dopustiva, ali nisu sve dobre. AI i MD pomeraju tacku alokacija na sledeći nacin.



$x$  i  $y$  osa označavaju protok (odlazni saobracaj) posiljaoca 1 i posiljaoca 2 u procentima, redom. Ravnopravno je da oba posiljaoca imaju isti odlazni saobracaj, a efikasno je kada je zbir jednak 1, tj. kada se koristi pun kapacitet mreže. Siva zona označava da se salje više podataka nego sto mreža može da podnese.

Aditivno uvecanje pomera pod uglom od 45 strpeni. Umnozeno smanjenje vraca relativno prema 0.



AIMD mehanizmom se konvergira ka optimalnoj tacki alokacije (preseku pravih efikasnosti i ravnopravnosti). On radi i u visedimenzionom scenariju. Ostali pristupi uvecanja i smanjenja kao sto su MIAD, MIMD, AIAD itd. ne rade posao.

Kako bi algoritam radio, zahteva samo binarni odgovor/signal od mreze. Nekoliko mogucih tipova signala je u upotrebi.

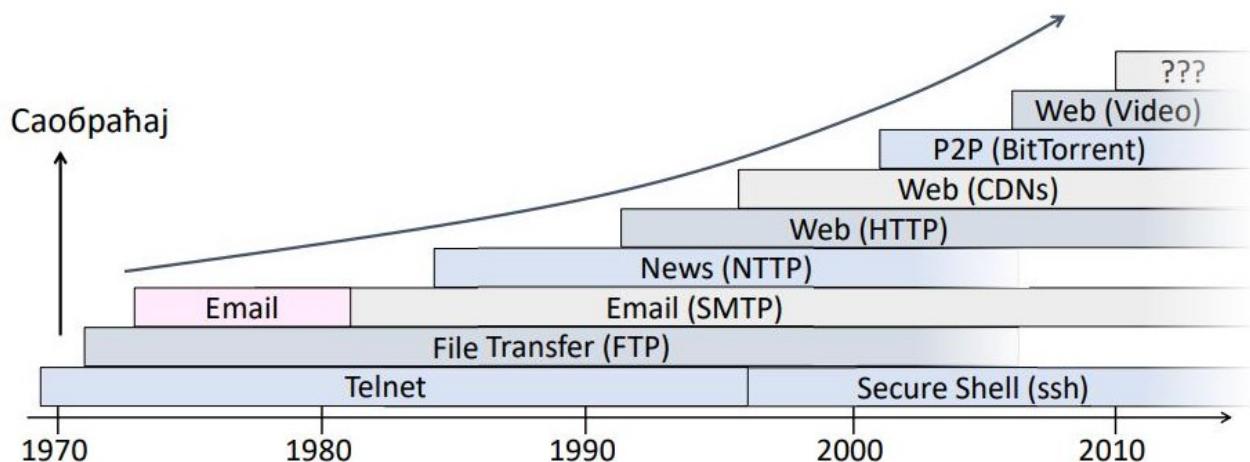
Сигнал	Пример протокола	+/-
Губитак пакета	TCP NewReno Cubic TCP (Linux)	Поуздано детектује Касно чује
Кашњење пакета	Compound TCP (Windows)	Рано чује Прави претпоставку
Сигнал рутера	TCP са експлицитним сигналом загушења	Рано чује Захтева подршку рутера

#### 46. Aplikativni sloj, uloga, interakcija sa slojem ispod, pregled Internet aplikacija.

Aplikativni sloj je najviši nivo i OSI I TCP/IP referentnih modela. U njemu je opisan rad aplikacija i njihovih interakcija sa servisima i protokolima nižih slojeva. Ovaj sloj je sloj najbliži korisniku, tj. sloj koji dostavlja mrežne usluge aplikacijama krajnjeg korisnika. Iako se može činiti da su servisi na aplikativnom sloju ono što korisnik vidi, to ne mora da znači – postoje servisi na aplikativnom sloju koji nemaju GUI i korisnik ih ne vidi – na primer DNS (Domain name system).

Poruke na aplikativnom sloju su uglavnom prevelike, npr HTTP poruka dovlači celu internet stranicu i samim tim neće stati u jedan paket – to povlači da se poruke na aplikativnom sloju dele na segmente koji će pripadati različitim paketima.

## Еволуција Интернет апликација



6

#### 47. DNS, uloga, raniji pristup, moderni pristup, TLD, sloganovi.

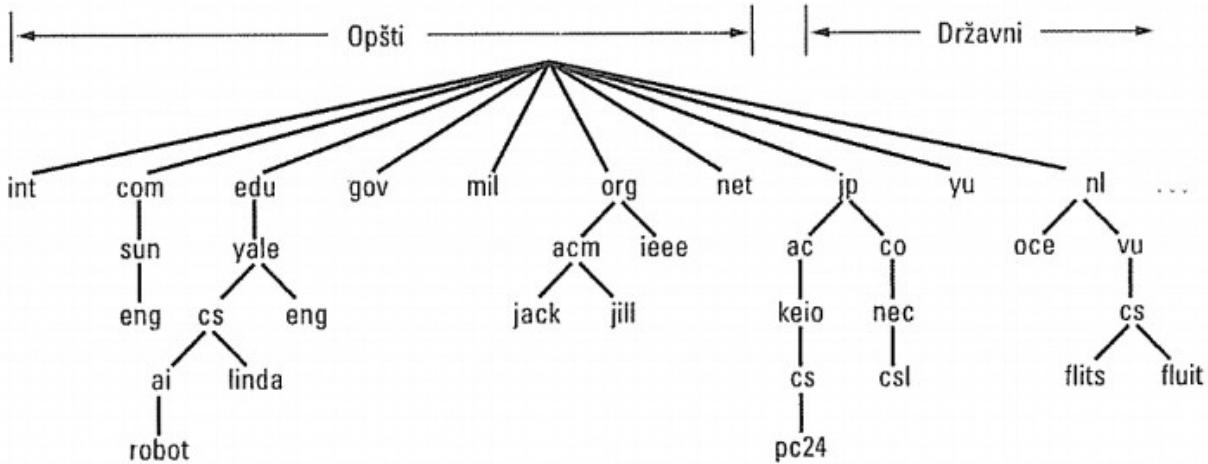
DNS (Domain name system) – Sistem imenovanja domena

Iako se programi teoretski mogu obraćati drugim računarima preko IP adresa, te IP adrese ljudi teško pamte. Ako bismo slali elektronsku poruku na [jana@128.111.24.41](mailto:jana@128.111.24.41), pa Janin internet provajder ili radna organizacija promene server na drugu lokaciju značilo bi da bismo sledeći put Jani slali poruku na drugu adresu. Zbog ovoga uvedena su imena kojima se ime računara razdvaja od njegove adrese. Ovako dobijamo da bi Janina adresa mogla da izgleda na primer [jana@alas.matf.bg.ac.rs](mailto:jana@alas.matf.bg.ac.rs). Naravno mreža radi samo sa brojevima i da bi se ovakav problem rešio, potrebno je naći rešenje za prevođenje tekstualnih imena u mrežne adrese.

Kada je postojao ARPANET, lista imena i svih IP adresa se čuvana u datoteci hosts.txt. Svi računari su morali preko noći da preuzimaju datoteku. Naravno ovakvo rešenje je dobro za mrežu od par stotina računara, međutim za današnji Internet je jako loše usled velikog broja Internet korisnika. DNS je stvoren da reši ovaj problem i njegova primarna uloga je preslikavanje imena računara u IP adrese. Ukratko DNS radi na sledeći način: aplikacija poziva razrešivač(resolver) i prosleđuje joj

domen(alas.matf.bg.ac.rs na primer), razrešivač šalje UDP paket lokalnom DNS serveru koji traži dato ime i vraca IP adresu razrešivaču koji tu IP adresu šalje aplikaciji.

Internet je organizaciono podeljen na preko 200 osnovnih domena (top-level-domain TLD). Svaki domen je izdeljen u poddomene, poddomeni u poddomene itd. Može se predstaviti preko stabla gde listovi nemaju poddomene i mogu predstavljati jedan računar ili firmu sa nekoliko hiljada računara.



Ako bismo želeli da se uključimo na neki deo stabla potrebno je zatražiti od administratora čvora iznad. Na primer ako bismo hteli da nam domen bude student.alas.matf.bg.ac.rs morali bismo tražiti dozvolu od administratora servera alas.matf.bg.ac.rs.

Kada razrešivač preda ime DNS serveru, od njega ne dobija samo IP adresu nego još neke dodatne podatke. Ti podaci koje vrti DNS server se nazivaju slogovi resursa. Znači DNS je preslikavanje imena u slogove resursa.

Ime domena je domen za koji važi slog i ovo polje je primarni ključ za pretraživanje slogova.

Životni vek govori o stabilnosti zapisa – vrlo stabilnim zapisima se dodeljuje velika vrednost, a informacijama privremenog karaktera mala vrednost.

Klasa je treće polje I za podatke na internetu ona ima vrednost IN.

Tip označava vrsta zapisa I to može biti npr IP adresa(tip A), razmena pošte(tip MX), alias IP adresе.

Vrednost je vrednost u koju se mapira prethodna četvorka. Na primer

alas.matf.bg.ac.rs | 86400 | IN | A| --- → 147.91.64.2

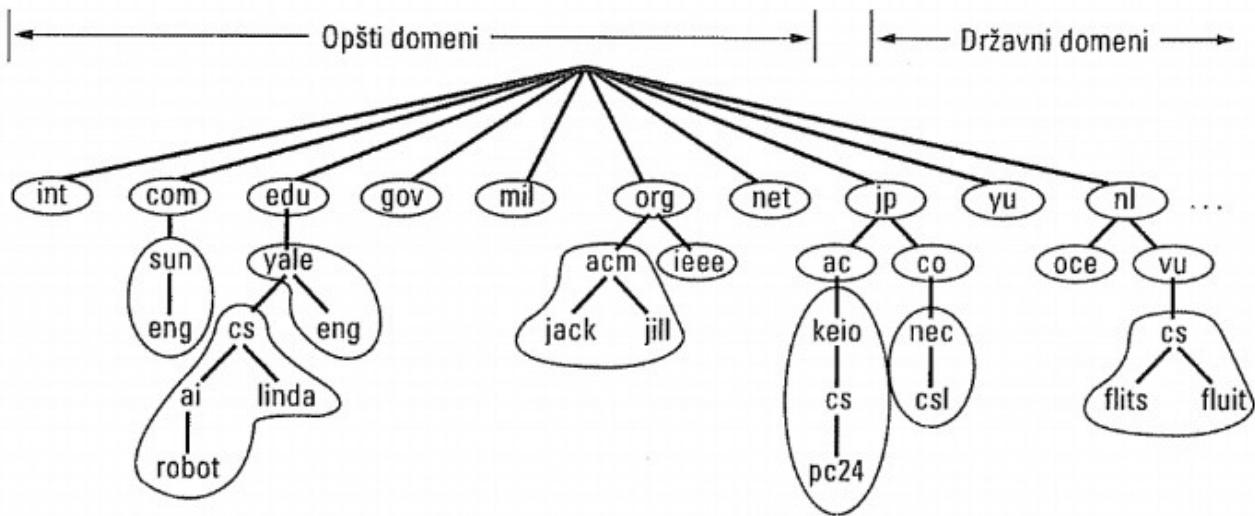
alas.matf.bg.ac.rs | 86400 | IN | MX -- → server alasa koji prima poštu(npr google ima brdo servera, jedan prima poštu)

alas.matf.bg.ac.rs | 86400 | IN | CNAME | alas.bg.ac.rs --- → (npr ako neko kuca alas.bg.ac.rs preusmeriće se na alas.matf.bg.ac.rs)

Znači DNS ne preslikava ime samo u IP adresu, nego I u još mnogo informacija koje DNS server vraća razrešivaču.

## 48. DNS, zone, opis mehanizama određivanja adresa

Teorijski bi jedan jedini server mogao da ima čitavu DNS bazu podataka, međutim on bi bio previše opterećen i u slučaju pada servera pao bi čitav Internet. Da bi se rešio ovaj problem DNS prostor je podeljen u više zona koje se međusobno ne preklapaju. Jedan način podele DNS na zone je na slici ispod.



Obično svaka zona ima primarni server koji izvlači informacije iz datoteke na svom disku I više sekundarnih servera koji informacije dobijaju od primarnog servera. Administratori određuju gde će se nalaziti granica između zona.

Kada razrešivač dobije upit za informacije nekog domena, on prosleđuje upit nekom lokalnom serveru imena.

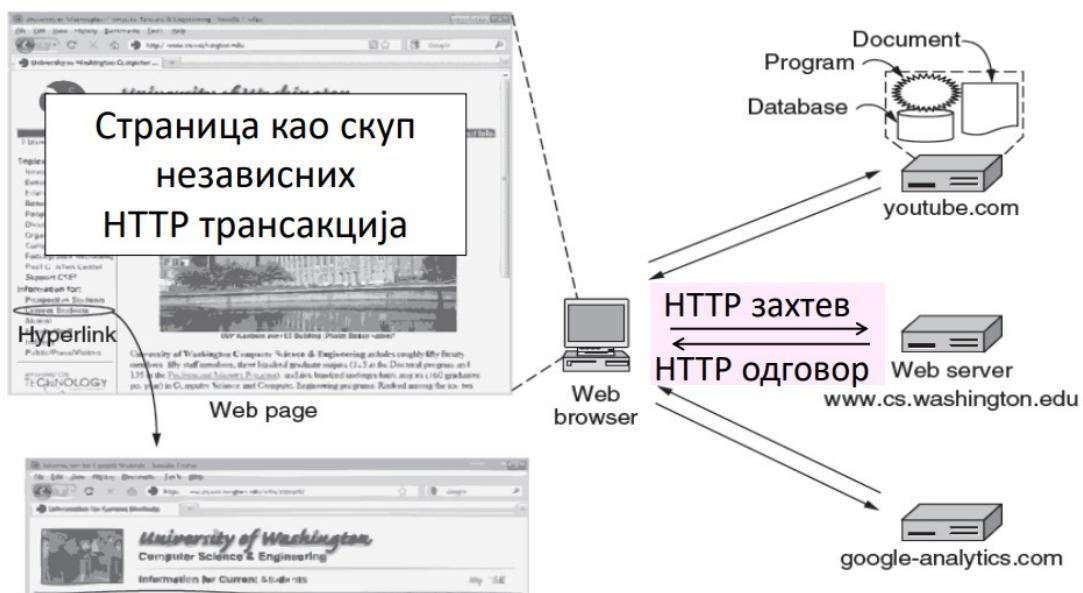
Ako je domen udaljen lokalni server šalje upit serveru iznad njega, ako on ne zna šalje iznad I tako se vrši pretraga po stablu. Ovakav postupak se naziva rekurzivno ispitivanje zato što će se informacija vratiti istim putem kojim je I upit otišao. Ovo je dobro rešenje jer je omogućeno keširanje gde server vrati put ako je već tamo išao, međutim zahtevno je sa memoriske strane.

Takođe postoji I iterativni DNS gde ako lokalni DNS server ne nađe odgovarajući domen, klijentu vraća referencu na drugi DNS server koji možda zna adresu, a ovaj može biti ili rekurzivni ili iterativni. Ovakva rešenja se na primer koriste kod korenog servera jer svi zahtevi idu preko njega pa bi keširanje bilo nemoguće zbog memoriskih ograničenja.

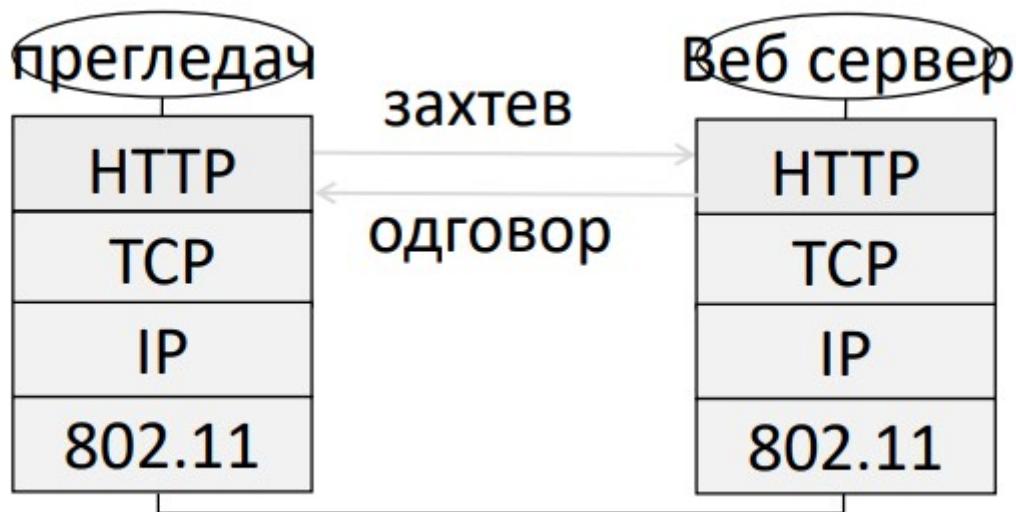
Postoje tajmeri kod DNS računara koji ako nakon određenog vremena nisu dobili odgovor zahtev šalju drugom serveru pod pretpostavkom da odgovor nije stigao jer je server isključen.

#### 49. HTTP Protokol, preuzimanje Veb dokumenata

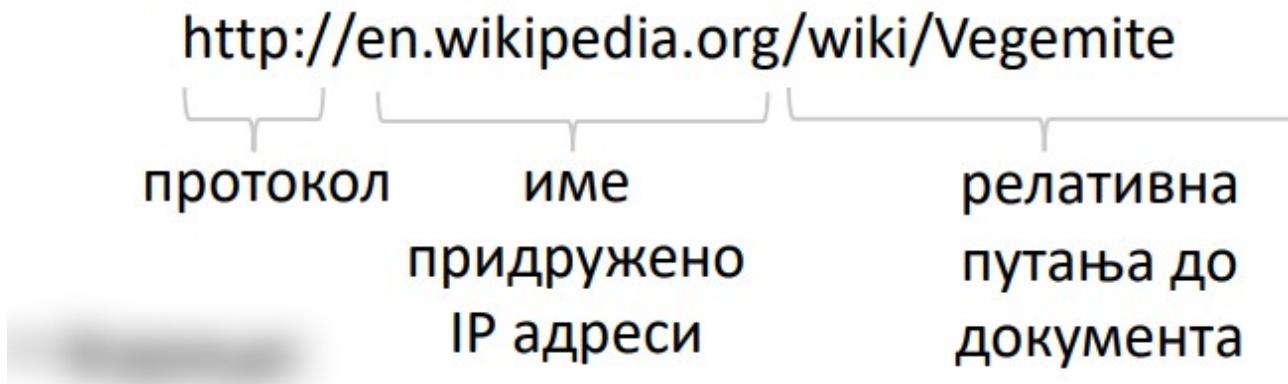
HTTP(Hypertext transfer protocol) je osnova današnjeg Interneta, to je protokol koji služi za transfer stranica na internetu između servera I klijenta.



Posao HTTP-a je da nađe način da prenese sve elemente(tekst, slika, javascript, ....) jedne stranice do korisnika, dok je posao Browzera da ih objedini I prikaže korisniku.



Na slici se može primetiti da je HTTP zahtev/odgovor(request/response) protokol koji služi za dohvatanje resursa koji se nalaze na Internetu. HTTP protokol se nalazi na TCP protokolu, obično na portu 80. Na slici iznad pregledač je odvojen od HTTP-a, ali se uglavnom HTTP nalazi unutar aplikacije, uglavnom sproveden putem biblioteke.



Na slici iznad možemo pogledati delove URL-a. Prvi deo je protokol I za URL uvek se koristi HTTP. Drugi deo je ime servera na kome se nalazi neki resurs, dok je poslednji putanja do određenog dokumenta na serveru.

Kako izgleda dohvatanje URL-a korišćenjem HTTP protokola?

Prvi korak je određivanje IP adrese korišćenjem imena servera I DNS-a. IP adresa nam je potrebna kako bismo znali lokaciju servera na internetu. Drugi korak bi bio uspostavljanje TCP konekcije sa serverom, pošto HTTP radi na TCP-u, ne možemo mnogo raditi ako prethodno ne uspostavimo TCP konekciju sa nekim serverom. Nakon uspostavljanja veze, sledi slanje HTTP zahteva serveru.

Nakon slanja zahteva, browser čeka da dobije HTTP odgovor od servera. Sledeći korak je opcioni I to je dohvatanje I izvršavanje ugrađenih dokumenata (slike, video klipovi, javascript, ...). Poslednji korak je gašenje veza I čišćenje.

Ukratko koraci dohvatanja URL-a korišćenjem HTTP protokola su:

- Nalaženje IP adrese servera
- Uspostavljanje TCP konekcije
- Slanje HTTP zahteva serveru

- Čekanje odgovora
- Izvršavanje dinamičkog koda ili dohvatanje ugrađenih resursa (ako ih ima)
- Gašenje TCP veza

Na sledećoj slici biće prikazan zahtev(request) deo HTTP protokola.

Метода	Опис
GET	Чита и враћа садржај Веб документа
HEAD	Чита заглавље Веб док.
POST	Додаје податаке Веб док.
PUT	Складишти Веб док.
DELETE	Уклања Веб док.
TRACE	Приказује долазни захтев
CONNECT	Веза кроз прокси
OPTIONS	Параметри захтева

Ako hoćemo da dohvatimo neku internet stranicu, koristili bismo GET komandu.

Na primer, GET [www.matf.bg.ac.rs](http://www.matf.bg.ac.rs) HTTP/1.1 šalje zahtev za dohvatanje stranice serveru Matematičkog fakulteta koristeći HTTP/1.1 protokol.

Post metod bismo koristili ako hoćemo da naš browser upiše neke podatke na internet stranici. Na primer ako bismo popunjavali neku anketu I upisivali naše odgovore.

Na primer,

POST [www.matf.bg.ac.rs/anketa.html](http://www.matf.bg.ac.rs/anketa.html) HTTP/1.1

...

student=da&godina+diplomiranja=2019

šalje podatke anketi na Matematičkom fakultetu korišćenjem HTTP/1.1 protokola.

Server kada pošalje odgovor, on uz ono što je klijent zahtevao pošanje I određeni kod. Lista kodova može se videti na sledećoj slici.

Код	Значење	Примери
1xx	Информација	100 = сервер приhvата захтев
2xx	Успех	200 = захтев успео
3xx	Преусмерење	301 = документ померен
4xx	Грешка клијента	403 =забрањен приступ; 404 = нема документа
5xx	Грешка сервера	500 = интерна логичка грешка

Kod koji mi kao klijenti želimo da dobijemo je uvek oblika 200 I nešto, jer tako počinju kodovi koji znače da se uspešno izvršila komunikacija.

404 kod je kod koji se jako često pojavljuje I on se vraća uz poruku ‘page not found’.

Primer za dobijanje koda 404:

<http://poincare.matf.bg.ac.rs/~ivan/index.php?content=ods>

403 kod je kod koji se dobija kada nemamo dozvolu za pristup nekom resursu.

Primer za dobijanje koda 403:

<http://poincare.matf.bg.ac.rs/~smalkov/files/>

Svi ovi kodovi predstavljaju sastavni deo HTTP protokola.

HTTP zahtevi pored samog contenta mogu pridružiti I neka zaglavlja da dodatno opišu sta je u pitanju. Na primer content-type: text/html nam govori da je u pitanju html stranica.

Функција	Примери заглавља
Инфо о прегледачу (клијент→ сервер)	User-Agent, Accept, Accept-Charset, Accept-Encoding, Accept-Language
Инфо о кеширању (оба смера)	If-Modified-Since, If-None-Match, Date, Last-Modified, Expires, Cache-Control, ETag
Стање прегледача (клијент→ сервер)	Cookie, Referer, Authorization, Host
Тип садржаја (сервер→ клијент)	Content-Encoding, Content-Length, Content-Type, Content-Language, Content-Range, Set-Cookie

38

## 50. HTTP performanse

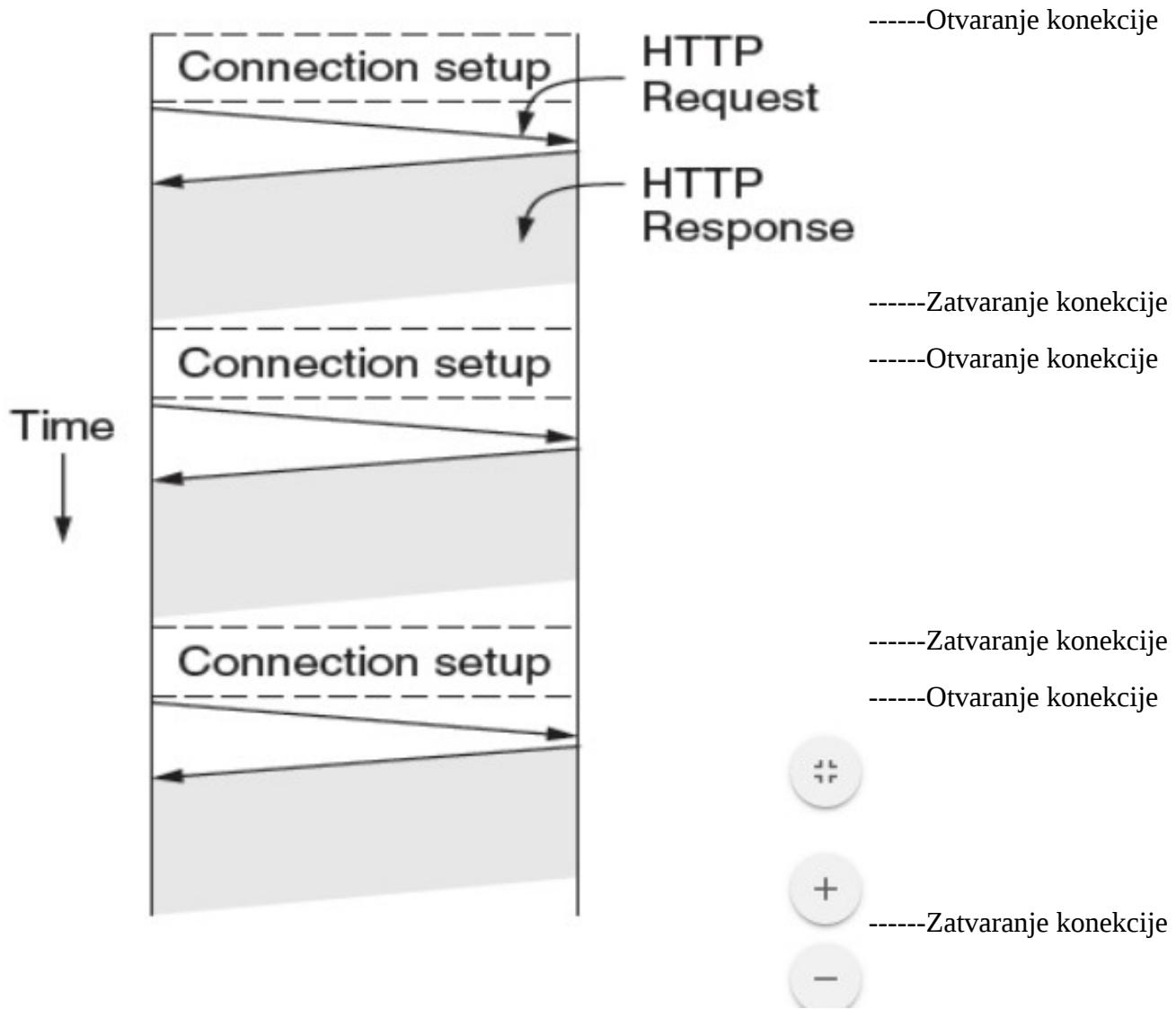
Prilikom korišćenja HTTP-a najdirektnija stvar koju želimo da popravimo je PLT(page load time) tj vreme proteklo od klika kojim šaljemo zahtev sa stranicom do pojave prvog razumnog prikaza u browseru. Niko ne voli stranice koje se dugo učitavaju.

PLT zavisi od nekoliko faktora. Jedan može biti struktura dokumenta gde ako je dokument neoptimizovan, pun slika, video klipova, javascript koda, učitavanje stranice se može znatno usporiti. Takođe zavisi I od same implementacije HTTP-a I veze između TCP-a I HTTP-a, a takođe PLT zavisi I od same brzine mreže.

U najranijim verzijama HTTP-a postojala je jedna TCP konekcija za preuzimanje resursa.

Na primer pokušavamo da dovučemo html stranicu koja ima nekoliko slika. Prvo bismo postavili konekciju sa stranicom, nakon toga poslali zahtev pa potom dobili odgovor tj sliku I zatvorili konekciju. Nakon toga ako bismo hteli sledeću sliku opet bismo morali da otvorimo konekciju,

pošaljemo zahtev, dobijemo odgovor I potom zatvorimo konekciju. Ovakav način bi na primer za galeriju slika bio katastrofalno neefikasan. Međutim u vreme stvaranja HTTPa ovo je bio jako jednostavan način za implementaciju.



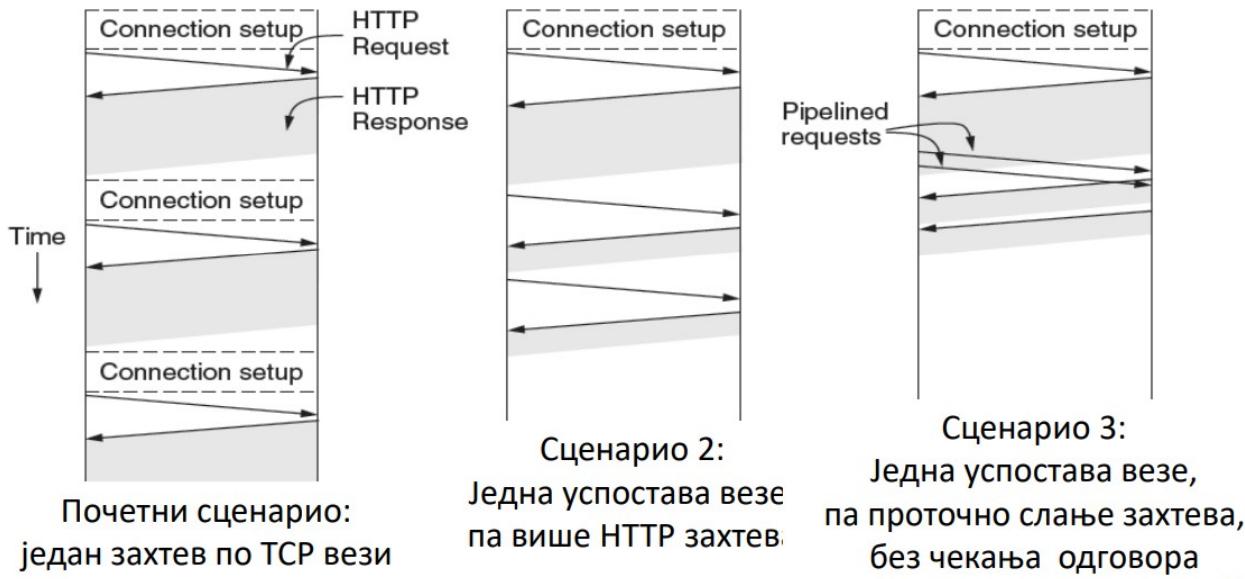
Na slici iznad osenčeni delovi predstavljaju HTTP odgovor, a neosenčeni vreme koje je protraćeno. Zamislite situaciju da su osenčeni delovi znatno manji. Za sekvencu malih fajlova imali bismo mnogo više potrošenog vremena na uspostavljanje konekcije nego na samo slanje odgovora.

Ovakav način povećava PLT iz razloga što se sve dešava sekvencijalno, jedno za drugim, a na Internetu ako su slike na primer na različitim serverima nema razloga da se slike sekvencijalno učitavaju. Otvaranje I zatvaranje nekoliko uzastopnih TCP konekcija na istom serveru je bespotrebno I to doprinosi uvećanju PLTa.

Neki od načina za smanjenje PLTa mogu biti kompresovanje samog sadržaja prilikom slanja(ako npr koristimo telefon, nema potrebe vratiti sliku u punoj rezoluciji, možemo programirati browser da prepoznače koji uređaj je u pitanju), prilagođavanje HTTPa da bolje koristi protok I o tome će u nastavku pitanja biti reči (da ne uspostavlja tcp konekcije kao iznad), možemo izbegavati slanje HTTP zahteva koji se često ponavljaju(keširamo ih ili koristimo veb proksije o kojima će biti priče u narednom pitanju), možemo takođe fizički približiti sadržaj korisniku(Content delivery network CDN, biće priče za dva pitanja).

Jedno od rešenja za smanjivanje PLTa je da imamo pametniji način korišćenja TCP veza. Browser će možemo programirati da drži 8 TCP konekcija *paralelno* otvorenih pošto serveri svakako mogu da rade sa parelelnim TCP konekcijama usled toga što je moguće da više klijenata komunicira sa serverom u isto vreme. Na ovaj način uspostavljanja veza će se dešavati u istom trenutku tako da neće biti gomilanja kašnjenja I korisniku će se činiti da se konekcija uspostavila jednom a da su se svi podaci zajedno poslali. Praktično mi na ovaj način predstavljamo jedan zahtev odnosno jednog klijenta kao 8 zahteva odnosno 8 klijenata. Naravno ni ovaj način nije najbolji baš zbog ove situacije, 1000 klijenata bi u ovom slučaju serveru izgledalo kao 8000.

Sledeći način bi bio da uspostavimo *samo jednu* TCP konekciju za više HTTP zahteva.

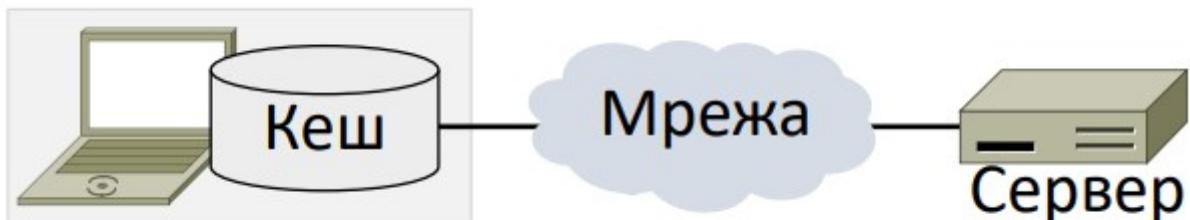


Obratite pažnju na osećene delove jer oni predstavljaju odgovore servera, ti odgovori su ono što nama treba, dok beli delovi predstavljaju razne obrade I čekanja. Mi želimo da smanjimo beli deo odnosno čekanje.

Ovakvo rešenje je ono koje se trenutno koristi u HTTP/1.1 protokolu ali mora se razumeti da ni ono nije savršeno - postavlja se pitanje koliko dugo držati TCP konekciju.

## 51. HTTP keširanje I HTTP proxy

Jedan od jednostavnih načina poboljšanja performansi je privremeno skladištenje već isporučenih stranica za slučaj da ih klijent zatraži ponovo I to se zove keširanje(caching). Na primer ima smisla keširati google.com, facebook.com, youtube.com jer ih korisnik često koristi.

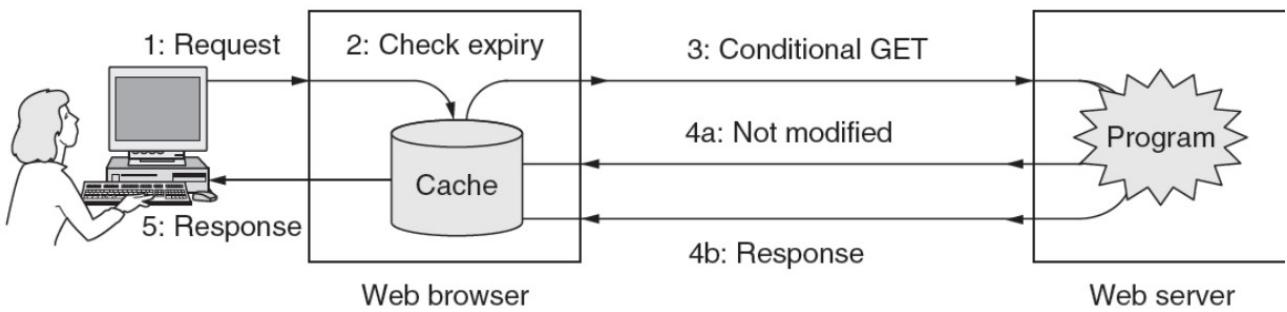


To skladištenje (cache) održava poseban zastupnički web proces (proxy) na Internetu dok svaki PC ima lokalni keš u browseru. Ako hoćemo određenu stranu na internetu, prvo pitamo proxy, ako je on ima zapamćenu-on je vrati, ako nema on je pronađe na internetu, zapamti u memoriji I vrati je nama. Ako internet stranica nije keširana na našem računaru, proksi će pitati proksi lokalne mreže da li ima keširanu stranicu(npr da li je neko u firmi tražio tu stranicu pre nas), ako ni on ne nadje taj proksi će pitati proksi internet provajdera. Ovakav sistem keša se zove hijerarhijsko web keširanje.

Pitanje je koliko dugo čuvati stranicu u memoriji. Keš koristi zaglavljje if-modified-since koje se može poslati serveru da se pita da li se stranica menjala nakon određenog datuma. Server pročita polje Last-modified I odgovara kešu(bio lokalni ili proksi) koji na osnovu ovoga zna da li da preuzme opet stranicu ili ne, na nekim sistemima postoji polje `expires` koje govori kada keš ističe. Drugi način može biti pravljenje sistema koji će prepostavljati koliko dugo određene stranice treba čuvati u sistemu, to su razne heuristike. (Npr sajtove berze ne treba uopšte keširati, dok na primer stranicu <http://poincare.matf.bg.ac.rs/nastavno/zrakic.html> možete keširati na 10 godina).

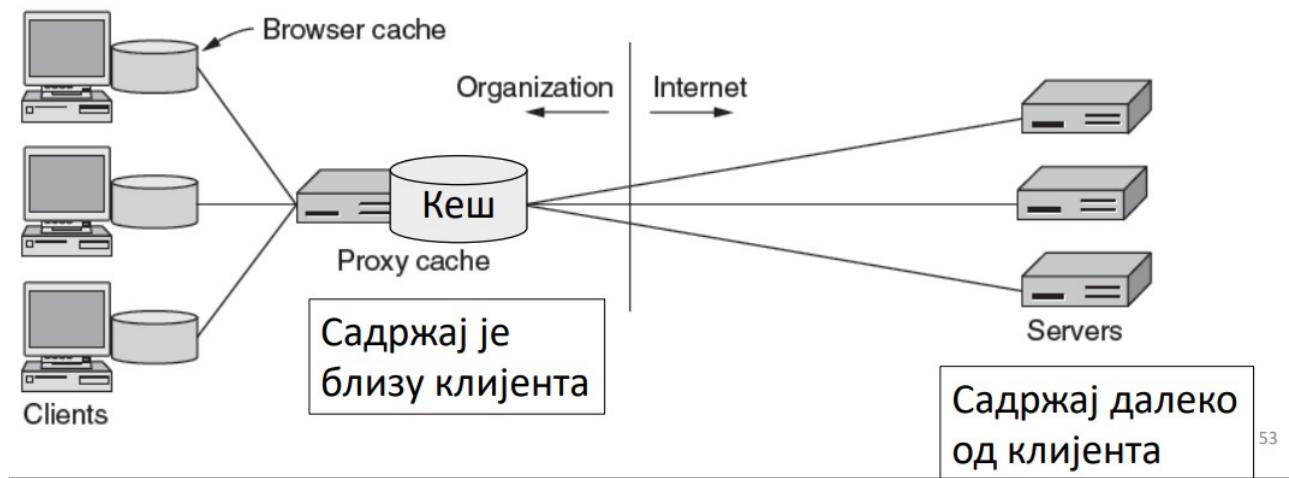
Dinamične veb stranice nikad ne treba keširati (npr generisane PHPom) jer se one menjaju u zavisnosti od zahteva do zahteva.

Još jedan način poboljšanja performansi je keširanje unapred gde će keš(lokralni ili proksi) nakon vraćanja stranice koje je zahtevao korisnik sve linkove sa te stranice smestiti u svoju memoriju u slučaju da mu zatrebaju u budućnosti. Ovaj način poboljšanja može dovesti do nepotrebognog skladištenja stranica koje korisnik nikad nije koristio.



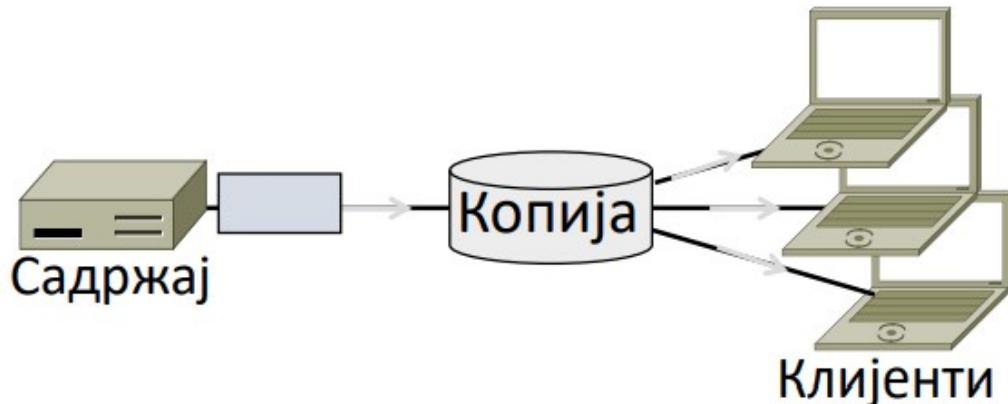
Na ovoj slici prikazan je ceo ciklus korišćenja metode keširanja. Naime klijent nakon što pošalje zahtev za određenom stranicom, prvo se proverava da li je ona keširana I ako jeste da li je I dalje ispravna. Prvo se pokušava lokalno utvrđivanje da li je ispravan sadržaj I ako jeste vraćamo ga korisniku, ako nije može se poslati uslovni GET (zove se GET jer želimo da dobijemo sadržaj od servera, a uslovni zato što želimo da dobijemo to samo ako nije promenjeno I ako nemamo keširano) zahtev serveru kojim se proverava da li je stranica ispravna I ako jeste (server vratí not modified) vratiti je korisniku, ako nije server vraca sadržaj koji se prosleđuje korisniku.

Proksiji su posrednici između grupe klijenata I veb servera koji uglavnom rade kao jedan veliki keš, da na primer u kompaniji delimo sadržaj među radnicima. Takođe u proksi se mogu ugraditi I sigurnosni sistemi, zabrana sadržaja itd (zabrane na fakultetskom internetu da odete na određene stranice na internetu) I ovaj deo je u principu bitniji od usluge pamćenja sadržaja.

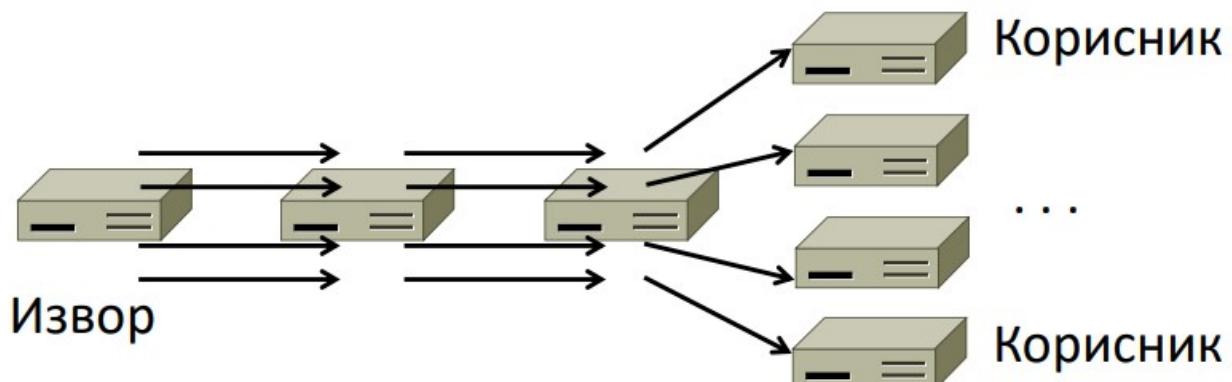


## 52. CDN – Content delivery network

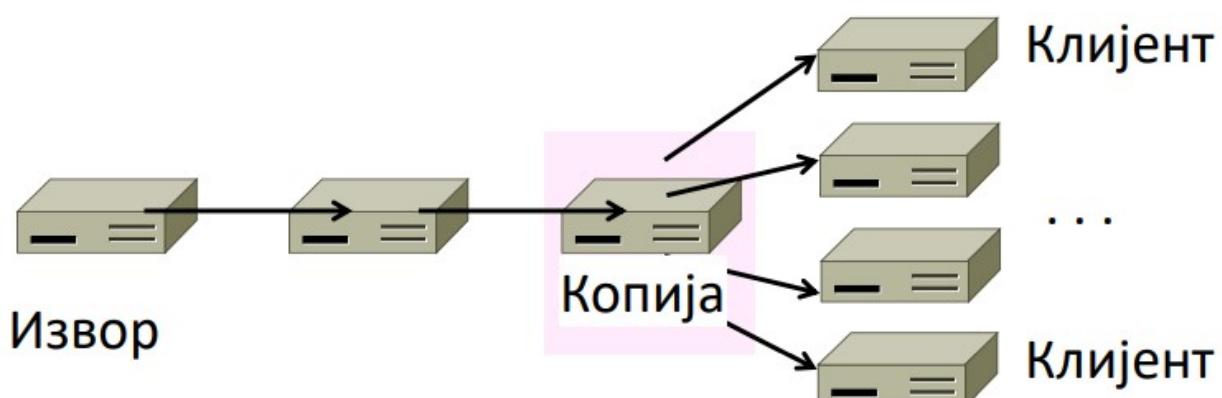
CDN predstavlja efikasan način približavanja sadržaja korisnicima, postavljajući kopije često korišćenih sadržaja na servere koji su fizički blizu korisnicima. Razlog pravljenja CDNa je što su popularni serveri postali zatrpani saobraćajem i samim tim su ugrožavali korisnički doživljaj.



Na sledećoj slici može se prikazati kako se jedan isti sadržaj šalje više puta preko servera preko kojih u principu ne bi bilo potrebe slati. Naime bilo bi lakše samo kopirati sadržaj na server najbliži grupi korisnika.

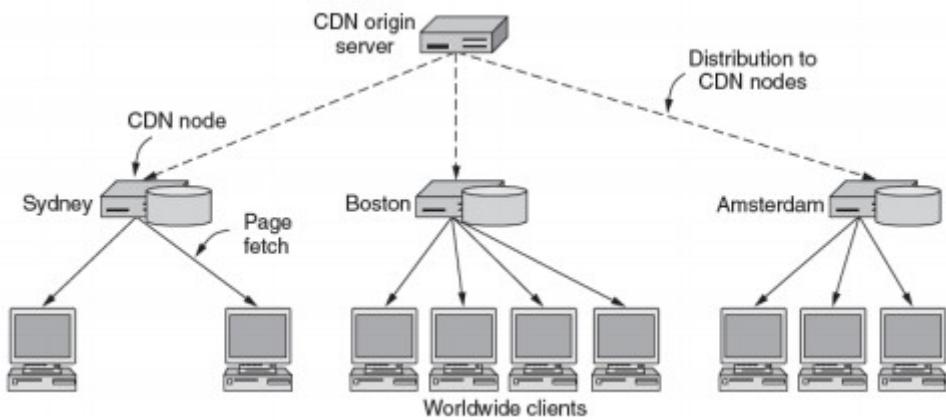


Umesto 12 hopova ovo bi moglo biti predstavljeno sa duplo manje odnosno 6.

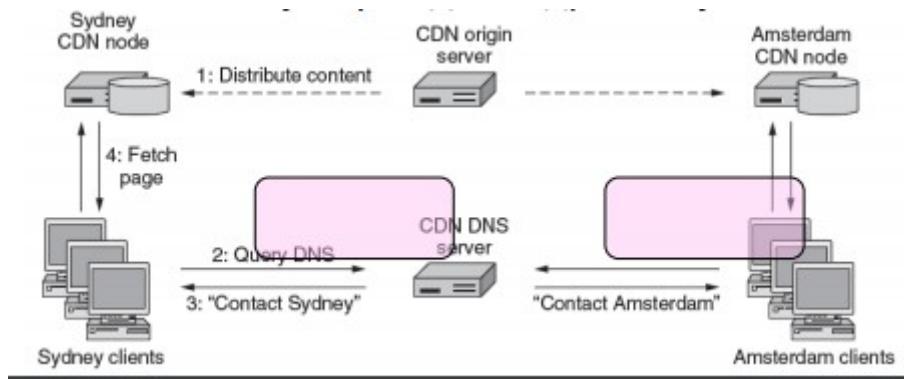


Na ovaj način ne samo što rasterećujemo server, nego rasterećujemo I servere koji se nalaze na putu do tog servera.

Želimo da približimo sadržaj geografski ciljano na primer, I ovakve kopije sadržaja se mogu pratiti putem DNSa. DNS će voditi računa gde se nalaze obližnje kopije tražene adrese.



Na slici iznad možemo primetiti kako se nekoliko CDN čvorova raspoređuju na internetu I na taj način pokrivaju određene geografske zone, I ovakvih čvorova ima mnogo. Svaki korisnik sadržaj traži od najbližeg CDNa. Potrebno je samo sada implementirati način pomoću koga se preko DNSa može mapirati određena internet adresa na određeni CDN čvor.



Na slici iznad imamo apstraktno objašnjeno kako ovo funkcioniše. Prvo Sydney clients pošalju zahtev DNSu, DNS im odgovara da se obrate Sydney CDN jer tu mogu naći kopiju sadržaja, nakon čega klijent iz Sidneja pita CDN u Sidneju I dobija odgovor.

Korisnici iz Amsterdama pitaju DNS za određeni sadržaj I on im odgovara da pitaju CDN u Amsterdamu nakon čega klijenti dobijaju sadržaj. Poenta grafika je da glavni serveri distribuiraju svoj sadržaj po CDN serverima, a da DNS serveri preusmeravaju klijente na najbliže CDN čvorove kako bi im Internet doživljaj bio što ugodniji.

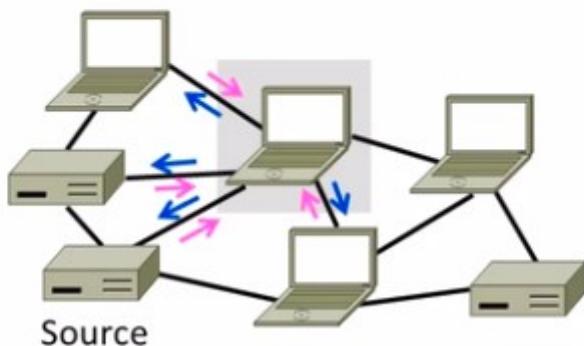
### 53. P2P (Peer-to-peer) i BitTorrent

P2P je model za efikasnu isporuku sadržaja koji je predstavljen kao alternativa CDN modelu. Ideja je da ne postoji jedan centralni server koji će usluživati klijente, već da klijenti međusobno budu I klijenti I serveri.

Naime CDN model je efikasan I jako dobro može da se skalira, takođe je jako pouzdan I održavan od strane stručnjaka. Međutim, CDN model nije savršen. Potrebno je napraviti celu ovu infrastrukturu, naći ljudе koji će ovo održavati I isposobiti, a to sve treba platiti novcem. Takođe, onaj ko postavlja tu infrastrukturu na CDNu je onaj koji ima kontrolu nad čvorovima ispod, on može kontrolisati sadržaj i dostavljati klijentima šta on hoće.

U P2P modelu cilj je dostavljanje sadržaja bez centralizovanog servera. Ideja je da imamo učesnike (peers) koji će se međusobno pomagati. Prvi sistem je bio Napster (ilegalna distribucija muzike), nakon toga BitTorrent. Čvorovi će se međusobno pomagati tako što će I preuzimati I slati(princip ja će poslati tebi ako ti pošalješ meni).

- Peer play two roles:
  - Download (→) to help themselves, and upload (←) to help others



Ostaje još pitanje kako korisnici da znaju gde da nabave sadržaj, ko od korisnika može da podeli sa njima. Rešenje su distribuirane heš tabele (DHT - Distributed hash tables). To su heš tabele koje su raspoređene po klijentima koje predstavljaju mapiranje izmedju imena fajla(ili nekog drugog primarnog ključa) I računara koji ga poseduje.

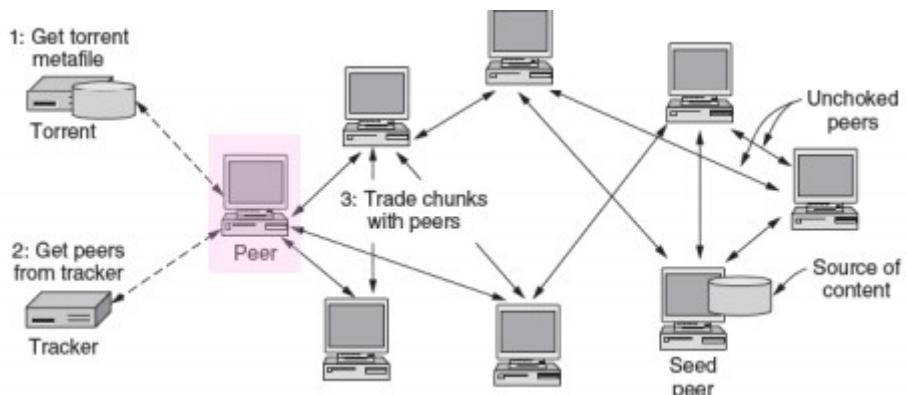
BitTorrent je glavni predstavnik P2P sistema trenutno. Jako brzo je porastao i koristi se za prenos velikih fajlova(na primer filmova, igrica, ...). Podaci se dostavljaju preko .torrent datoteka koje sadrže razne dodatne informacije poput veličine fajla npr. Sadržaj se preko bittorrenta šalje u delovima da bi smo mogli da koristimo paralelizam I samim tim imamo bolje performanse(na primer ako bismo preuzimali ceo fajl od nekoga ko uploaduje brzinom od 1mb/s a mi imamo download od 10mb/s, mi bismo protračili potencijalnih 9mb/s. Slanjem u delovima možemo

preuzimati od 10 ljudi koji pojedinačno imaju upload od 1mb/s I time iskoristiti naš pun download potencijal). Za pretragu su se ranije koristili serveri koji su govorili ko ima koji fajl, dok se danas sve više uvodi DHT.

Koraci u skidanju torrenta su:

1. Kontaktiranje tracker servera ili DHTa kako bismo saznali koje korisnike da kontaktiramo vezano za sadržaj. Informacije vezane za lokaciju tracker servera ili informacije vezane za DHT se nalaze u .torrent datoteci koju prvo preuzimamo.
2. Nakon kontaktiranja korisnika, međusobno se menjamo delovima fajla koji svi želimo da preuzmemos I tako svako dobija deo koji mu treba.
3. Ubrzavamo upload ka korisnicima koji nama daju dobar upload, dok usporavamo upload korisnicima koji nama daju spor upload. Znači pomagaćemo više onima koji nama više pomažu, a one usporavati one koji nas usporavaju.

Na slici ispod brojevima je označen redosled rada BitTorrenta.



Delovi fajla koji se međusobno šalju su nasumično izabrani, jer ako bismo uzimali sekvencijalno sama mogućnost trgovine u smislu kome šta treba bi se smanjila, svima bi trebao neki krajnji deo ili tako nešto, ovako se svaki deo nalazi kod nekoga.

Svi će stopirati kontakt sa onim klijentom koji ne daje dobru uslugu.

