

# [RM] Pitanja i odgovori

Momir Adžemović

11. maj 2020.

## Sažetak

Ova skripta predstavlja kombinovan materijal sa slajdova profesora Kartelja i knjige Tanenbauma tj. više je opširna od slajdova, a manje od knjige. Preporučio bih kolegama da svakako pročitaju knjigu ili odgledaju video prezentacije profesora Tanenbauma.

## Sadržaj

<b>1 Komponente mreže, tipovi mreža, primeri mreža, mreže prema dimenziji, međumreže</b>	<b>6</b>
1.1 Komponente mreže . . . . .	6
1.2 Tipovi mreža . . . . .	6
1.3 Primeri mreža . . . . .	6
1.4 Mreže prema dimenziji . . . . .	7
1.5 Međumreža . . . . .	7
<b>2 Protokoli i slojevi</b>	<b>7</b>
<b>3 Referentni modeli protokola i slojeva, jedinice podataka, organizacije za standarde.</b>	<b>9</b>
3.1 Referentni modeli protokola i slojeva . . . . .	9
3.1.1 OSI model . . . . .	10
3.1.2 TCP/IP model . . . . .	11
3.1.3 TCP/IP ili OSI model . . . . .	12
3.2 jedinice podataka . . . . .	12
3.3 organizacije za standarde . . . . .	13
<b>4 Fizički sloj: uloga, pojednostavljeni model, kašnjenja, BDP, primeri.</b>	<b>13</b>
4.1 uloga . . . . .	13
4.2 pojednostavljeni model . . . . .	13
4.3 Kašnjenje . . . . .	13
4.4 BDP . . . . .	14
4.5 Primeri (kašnjenje i BDP) . . . . .	14
<b>5 Žičani i optički komunikacioni medijumi</b>	<b>14</b>
5.1 UTP kablovi . . . . .	14
5.2 Koaksijalni kablovi . . . . .	14
5.3 Instalacija za prenos struje . . . . .	15
5.4 Optički kabal . . . . .	15

<b>6 Bežični komunikacioni medijumi</b>	<b>15</b>
6.1 Radio talasi . . . . .	15
6.2 Mikrotalasi . . . . .	15
6.3 Svetlost . . . . .	16
6.4 Žičani i bežični prenos . . . . .	16
<b>7 Komunikacioni sateliti</b>	<b>16</b>
7.1 Geostacionarni sateliti (GEO) . . . . .	17
7.2 Srednje visoko-orbitni sateliti (MEO) . . . . .	17
7.3 Nisko-orbitni sateliti (LEO) . . . . .	17
7.4 Sateliti i optika . . . . .	18
<b>8 Signali: prenos, frekvenciona reprezentacija, signal u žičanim, optičkim, bežičnim medijumima.</b>	<b>18</b>
8.1 Prenos - Furijeova analiza . . . . .	18
8.2 Prenos preko žice . . . . .	19
8.3 Prenos preko svetlosti . . . . .	20
8.4 Prenos preko bežičnih medijuma . . . . .	20
<b>9 Modulacija i multipleksiranje signala.</b>	<b>20</b>
9.1 Modulacija . . . . .	20
9.1.1 Direktna modulacija (baseband) . . . . .	20
9.1.2 Modulacija preko nosača (passband) . . . . .	21
9.2 Multipleksiranje signala . . . . .	23
<b>10 Prirodna ograničenja prenosa signala.</b>	<b>24</b>
<b>11 Pregled relevantnijih sistema komunikacije.</b>	<b>24</b>
11.1 Sistem fiksne telefonije . . . . .	24
11.2 Sistem mobilne telefonije . . . . .	25
11.3 Internet preko kablovske . . . . .	25
11.4 Kablovska ili (A)DSL . . . . .	25
<b>12 Sloj veze: uloga, komunikacija sa slojem ispod i iznad, kratko objašnjenje spiska aktivnosti na sloju veze.</b>	<b>26</b>
<b>13 Uokvirivanje u sloju veze</b>	<b>26</b>
<b>14 Kodiranje grešaka u sloju veze.</b>	<b>28</b>
14.1 Hamingovo rastojanje . . . . .	29
<b>15 Detekcija grešaka u sloju veze.</b>	<b>29</b>
15.1 Provera parnosti: . . . . .	29
15.2 Kontrolni zbir: . . . . .	29
15.3 Internet kontrolni zbir . . . . .	30
15.4 Ciklična provera redundanse (CRC) ili polinomijalni kodovi	31
15.5 Detekcija grešaka u praksi . . . . .	31
<b>16 Korekcija grešaka u sloju veze</b>	<b>32</b>
16.1 Hamingov kod za korekciju HD=3 . . . . .	32
16.2 Kodovi za korekciju u praksi . . . . .	33
16.3 Detekcija ili korekcija . . . . .	33

<b>17 Sloj veze: tipovi servisa, okruženje, utopijski jednosmerni protokol.</b>	<b>33</b>
17.1 Okruženje: Pregled struktura i funkcija . . . . .	34
17.2 Utopijski jednosmerni protokol . . . . .	35
<b>18 Kontrola toka, ARQ, pauze (tajmauti), duplikati, protokol „stani i čekaj“ za savršen i nesavršen kanal.</b>	<b>36</b>
18.1 Protokol „Stani i čekaj“ za savršen kanal . . . . .	36
18.2 Protokol „Stani i čekaj“ za nesavršen kanal . . . . .	36
<b>19 Protokol kliznih prozora u sloju veze, „1-bitni“, „vrati se N“, „selektivno ponavljanje“.</b>	<b>38</b>
19.1 Protokol kliznih prozora u sloju veze (opšta priča) . . . . .	39
19.2 Protokol „1-bitni“ . . . . .	40
19.3 Protokol „Vrati se N“ . . . . .	41
19.4 Protokol „Selektivno ponavljanje“ . . . . .	42
<b>20 MAC podsloj: uloga, alokacija kanala, ALOHA protokol.</b>	<b>42</b>
20.1 protokol ALOHA . . . . .	43
<b>21 CSMA, CSMA/CD, BEB.</b>	<b>43</b>
21.1 protokol CSMA . . . . .	44
21.2 protokol CSMA/CD . . . . .	44
21.3 Binarno eksponencijalno odlaganje (BEB) . . . . .	45
<b>22 MAC protokoli zasnovani na redosledu: „Token Ring“</b>	<b>45</b>
22.1 Protokol „Token Ring“ . . . . .	45
<b>23 MAC protokoli za bežične mreže</b>	<b>47</b>
23.1 Protokol MACA . . . . .	48
<b>24 Klasični Eternet - IEEE 802.3</b>	<b>50</b>
<b>25 Moderni (komutirani) Eternet</b>	<b>50</b>
25.1 Razvodnik (HUB) . . . . .	51
25.2 Skretnica (Switch) - opšte . . . . .	51
25.3 Skretnica (Switch) - učenje unazad . . . . .	52
25.4 Eliminacija petlji . . . . .	53
<b>26 Mrežni sloj: uloga, motivacija, rutiranje i prosleđivanje (ukratko), tipovi servisa na mrežnom sloju, objašnjenja i njihov uporedni odnos.</b>	<b>54</b>
26.1 Tipovi servisa na mrežnom sloju . . . . .	54
26.2 Internet protokol (IP) . . . . .	55
26.3 Datagrami i virtuelna kola . . . . .	56
26.4 Povezivanje različitih mreža (međumreža) . . . . .	57
<b>27 IP adrese i prefiksi</b>	<b>57</b>
27.1 Prefiksi . . . . .	57
27.2 Privatne i javne IP adrese . . . . .	58
<b>28 IP prosleđivanje</b>	<b>58</b>
<b>29 ARP i DHCP</b>	<b>60</b>
29.1 Dodeljivanje IP adrese (DHCP) . . . . .	60
29.2 Određivanje adrese u sloju veze za ciljnu IP adresu (ARP) .	61

<b>30 ICMP i NAT</b>	<b>62</b>
30.1 Protokol ICMP . . . . .	62
30.2 NAT tehnika . . . . .	63
<b>31 Rutiranje: mehanizmi alokacije protoka, modeli isporuke, ciljevi rutiranja, principi dizajna algoritama rutiranja, rutiranje sa najkraćim putevima (najmanjim troškom), Dijkstrin algoritam.</b>	<b>64</b>
31.1 Modeli isporuke . . . . .	64
31.2 Ciljevi rutiranja . . . . .	65
31.3 Principi dizajna algoritama rutiranja . . . . .	65
31.4 Rutiranje sa najkraćim putevima (najmanjim troškom) . . . . .	65
31.5 Dijkstrin algoritam . . . . .	66
<b>32 Rutiranje zasnovano na vektoru razdaljine. (Distance Vector Routing)</b>	<b>66</b>
32.1 Problem brojanja do beskonačnosti: . . . . .	67
<b>33 Plavljenje (Flooding)</b>	<b>68</b>
<b>34 Rutiranje zasnovano na stanju veza (Link state routing)</b>	<b>68</b>
<b>35 Višeciljno rutiranje sa najkraćim putevima (ECMP)</b>	<b>69</b>
<b>36 Hijearhijsko nasleđivanje</b>	<b>70</b>
36.1 Podela i sažimanje IP prefiksa . . . . .	71
<b>37 Transportni sloj: Uloga, tipovi servisa i njihovo poređenje</b>	<b>71</b>
37.1 Tipovi servisa . . . . .	72
37.2 Poređenje servisa (TCP i UDP) . . . . .	72
<b>38 Socket API, primer jednostavnog klijent-server (uopšteno), portovi</b>	<b>73</b>
38.1 Socket API . . . . .	73
38.2 Portovi . . . . .	74
<b>39 UDP</b>	<b>74</b>
<b>40 Uspostava i prekid mreže na transportnom sloju (uopšteno)</b>	<b>76</b>
40.1 Proces uspostave veze . . . . .	76
40.2 Proces prekidanja veze . . . . .	77
<b>41 Protokoli kliznih prozora</b>	<b>79</b>
41.1 Pošiljalac . . . . .	79
41.2 Primalac . . . . .	80
<b>42 Kontrola toka podataka na transportnom sloju</b>	<b>80</b>
<b>43 Retransmisija i prilagodljive pauze (tajmauti) na transportnom sloju</b>	<b>81</b>
43.1 Prilagodljive pauze . . . . .	82

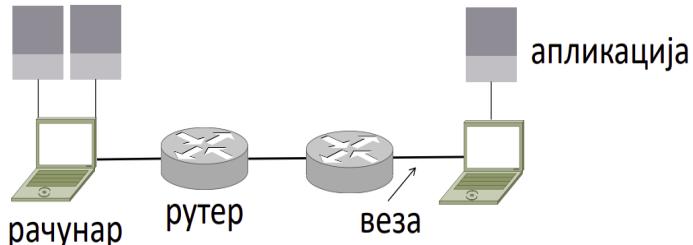
<b>44 TCP, svojstva, zaglavje, realizacija kliznih prozora, uspostava i prekid veze (specifično).</b>	<b>83</b>
44.1 TCP zaglavje . . . . .	84
44.2 Realizacija kliznih prozora . . . . .	85
<b>45 Zagušenje na transportnom sloju, opis problema i mehanizma za rešavanje (AIMD).</b>	<b>85</b>
45.1 Efekti zagušenja . . . . .	86
45.2 AIMD . . . . .	86
<b>46 Aplikativni sloj: uloga, interakcija sa slojem ispod, pregled Internet aplikacija.</b>	<b>87</b>
46.1 Interakcija sa slojem ispod . . . . .	87
46.2 pregled internet aplikacija . . . . .	88
<b>47 DNS: uloga, raniji pristupi, moderni pristup, TLD, slogovi</b>	<b>88</b>
47.1 Raniji pristup . . . . .	88
47.2 Moderni pristup . . . . .	88
47.3 TLD (Top-Level Domains) . . . . .	89
47.4 DNS slogovi (DNS records) . . . . .	89
<b>48 DNS: zone, opis mehanizma određivanja adresa</b>	<b>90</b>
48.1 Tipovi DNS servera . . . . .	91
48.2 Keširanje . . . . .	91
48.3 Lokalni i koreni DNS serveri . . . . .	92
48.4 DNS poruke . . . . .	92
<b>49 HTTP protokol: preuzimanje veb dokumenta</b>	<b>93</b>
49.1 Proces preuzimanje veb dokumenta (klijent) . . . . .	93
49.2 Proces preuzimanja veb dokumenta (server) . . . . .	94
49.3 Komande, kodovi i dodatne opcije HTTP protokola . . . . .	95
<b>50 HTTP performanse</b>	<b>96</b>
<b>51 HTTP keširanje i HTTP proksiji</b>	<b>97</b>
51.1 HTTP keširanje . . . . .	97
51.2 HTTP proksiji . . . . .	97
<b>52 CDN (Content Delivery Networks)</b>	<b>98</b>
<b>53 P2P</b>	<b>99</b>
53.1 BitTorrent protokol . . . . .	100

# 1 Komponente mreže, tipovi mreža, primeri mreža, mreže prema dimenziji, međumreže

## 1.1 Komponente mreže

Mrežu možemo da posmatramo kao neusmereni graf (teoretski mreža može da bude usmereni graf tj. veza slanja podataka između dva čvora može da bude jednosmerna). Komponente mreže su:

- Aplikacija, koristi mrežu i predstavlja komponentu lista grafa mreže.  
Primer: Skype, Itunes, Minecraft...
- Računar, podržava aplikaciju predstavlja završni čvor u grafu mreže.  
Primer: Dekstop računar, laptop, telefon...
- Ruter, prosledjuje poruke između čvorova i predstavlja unutrašnji čvor grafa mreže.  
Primer: kablovski/DSL modem, ...
- Веза или канал, spaja čvorove i predstavlja granu u čvoru.  
Tipovi: Žičani i bežični.



## 1.2 Tipovi mreža

- **Puni dupleks** - Mogu da se šalju podaci u oba smera istovremeno.  
Primer: Telefon.
- **Polu dupleks** - Mogu da se šalju podaci u oba smera, ali ne istovremeno.  
Primer: Voki-toki.
- **Simpleks** - Mogu samo da se šalju ili samo da se primaju podaci.  
Primer: Tastatura, monitor.

## 1.3 Primeri mreža

- WiFi (802.11)
- Poslovne / Ethernet
- ISP (Internet Service Provider)
- Kablovska / DSL

- Mobilna telefonija (2G, 3G, 4G, 5G)
- Bluetooth
- Telefon
- Sateliti
- ...

#### 1.4 Mreže prema dimenziji

- **PAN (Personal Area Network)** - Odnosi se na jednu osobu.  
Primer: bluetooth.
- **LAN (Local Area Network)** - Odnosi se na mrežu u okviru neke zgrade.  
Primer: WiFi, Ethernet.
- **MAN (Metropolitan Area Network)** - Odnosi se na mrežu na nivou nekog grada, univerziteta ili regiona.  
Primer: kablovksa, DSL .
- **WAN (Wide Area Network)** - Odnosi se na mrežu na nivou neke velike geografske površine.  
Primer: veliki ISP (Telekom, SBB).
- **Internet.**

#### 1.5 Međumreža

Međumreža (eng. internetwork) ili internet (malim početnim slovom) je mreža koja se dobija povezivanjem više različitih mreža. Internet (velikim slovom) je internet koji svi mi koristimo.

Intuitivno spajanjem LAN i WAN ili dva LAN-a dobijamo međumrežu. Postojeći dogovor oko terminologije međumreže:

- Ako dve različite institucije razvijaju dva različita dela mreže, onda to nazivamo međumreža.
- Ako se dva dela mreže razlikuju u tehnologiji izgradnje, onda to nazivamo međumreža.

## 2 Protokoli i slojevi

### Šta sve treba mreža da radi za korisnike?

- Pravi i prekida konekciju;
- Pronalazi putanju za transfer podataka;
- Pouzdano šalje podatke;
- Šalje podatke proizvolje veličine;
- Brzina slanja se prilagođava mogućnosti mreže;
- Omogućava siguran prenos tokom transakcije;
- Omogućava dodavanje čvorova
- ...

Neki od ovih zahteva su apstraktniji od drugih što je jedan od motiva za raslojavanje.

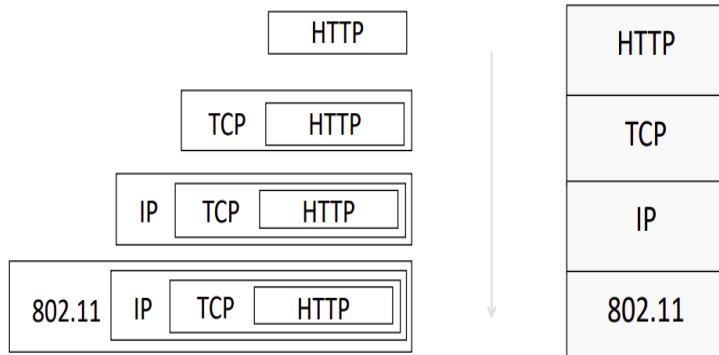
Prve verzije mreža su pravljenje tako da hardver ima prioritet u odnosu na softver. To više nije moguće, jer je mreža previše kompleksna. Kako bi se smanjila kompleksnost mreže, struktura mreže je organizovana kao stek slojeva (nivoa). Svrha svakog sloja (sem aplikativnog čija je svrha da omogući usluge korisniku) je da omogući određene servise višem sloju.

**Internet sloj** je grupa protokola i mehanizama koji čine nekakvu funkcionalnu celinu.

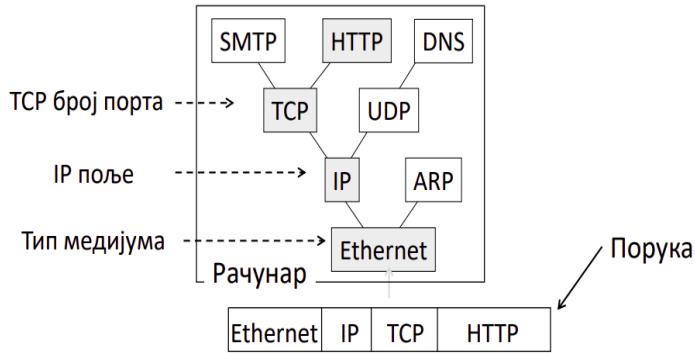
**Protokoli** predstavljaju skup pravila koji se poštuju u komunikaciji. Primer: http, ftp, smtp, tcp, udp, ip, 802.3, 802.11 ...

**Enkapsulacija** je mehanizam u kome svaki sloj prima poruku od višeg sloja i obmotava ga okvirom koji sadrži informacije vezane za taj sloj i prosleđuje nižem sloju.

Primer: Šalje se HTTP zahtev sa aplikativnog sloja koji koristi TCP protokol transportnog sloja. U transportnom sloju se podaci HTTP zahteva obmotavaju u segment koji sadrži dodatne informacije, pored HTTP zahteva, i prosleđuje taj segment mrežnom (nižem) sloju. Mrežni sloj obmotava segment u paket i prosleđuje paket sloju veze koristeći IP protokol. Sloj veze obmotava paket u okvir koristeći određeni protokol i prosleđuje fizičkom sloju. Neka je taj protokol 802.11 (bežična mreža). Fizički sloj prenosi podatke kao tok bitova. Niz ovih protokola čini (od najvišeg ka najnižem) jedan konkretan stek protokola.



Kako jedan niži sloj može da uslužuje više različitih slojeva na višem nivou, onda mora da postoji neki mehanizam demultiplesiranja kako bi primalac mogao da raspakuje poruku do aplikativnog sloja. Preko TCP porta možemo odrediti koji protokol se uslužuje na aplikativnom sloju (primer. http port je 80). Preko IP polja možemo odrediti da li se koristi TCP ili UDP (ili neki treći manje poznati). Preko tipa medijuma može se odrediti koji protokol se koristi na mrežnom nivou.



#### Prednosti raslojavanja:

- Prikrivanje informacija.
- Nezavisnost višeg sloja od nižeg sloja (HTTP radi uvek isto, nezavisno od toga da li koristimo wifi ili ethernet).
- Laka konverzija poruka (primer: Oba čvora u regionu povezani žičano i koriste ethernet, ali da bi se poslala poruka potrebno je u nekom trenutku prenosi podatke Preko brežične mreže).

#### Mane raslojavanje:

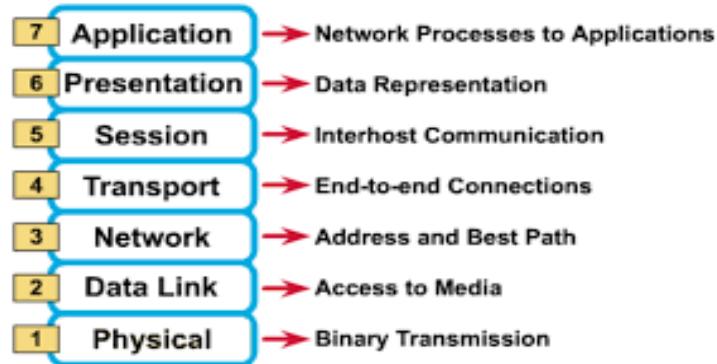
- Povećani troškovi memorije (manje bitno za duže poruke).
- Prikrivanje informacija (prednost i mana). Generalno korisno, ali možda neke aplikacije žele da znaju da li se podaci prenose preko kabla ili bežično.

## 3 Referentni modeli protokola i slojeva, jedince podataka, organizacije za standarde.

### 3.1 Referentni modeli protokola i slojeva

Potrebno je izvršiti podelu funkcionalnosti na slojeve. Koja funkcionalnost pripada kom sloju? Dva najpoznatija referentna modela su TCP/IP referentni model koji je dobio ime po popularnim protokolima TCP i IP, i OSI (Open System Interconnection) referentni model.

### 3.1.1 OSI model



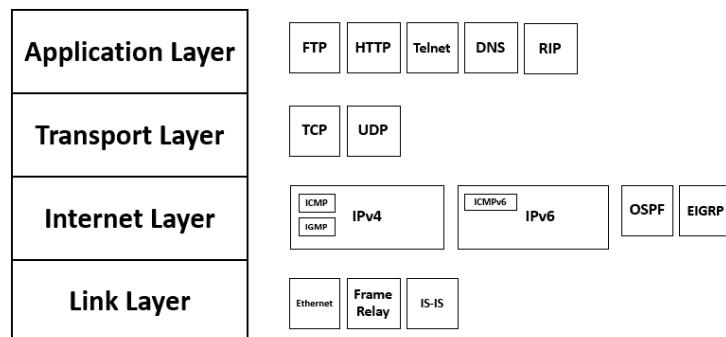
Referentni OSI model je zasnovan na sledećim principima:

1. Tamo gde je potrebna posebna apstrakcija treba da se napravi poseban sloj.
2. Svaki sloj treba da ima svoju dobru definisanu funkciju.
3. Svaki sloj treba da se bira na osnovu već poznatih standardizovanih protokola.
4. Granice slojeva treba da se izaberu tako da prenos podataka sa sloja na sloj minimalan.
5. Broj slojeva treba da bude dovoljno veliki da se različite funkcije nalaze na različitim slojevima, ali i dovoljno mali da sistem ne bude glomazan.

**Slojevi (od najnižeg):**

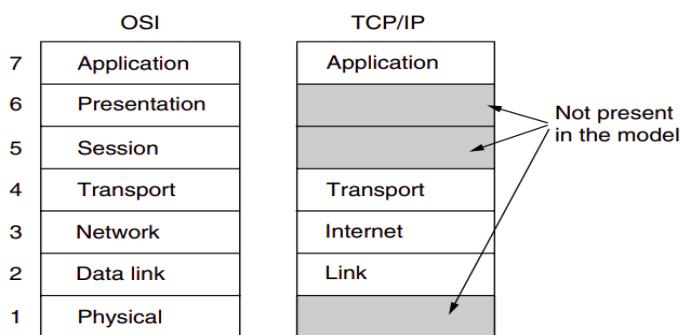
1. **Fizički sloj:** Prenosi bitove preko komunikacionog kanala.
2. **Sloj veze podataka:** Prenosi okvire na nivou kanala sa mogućom proverom greške i eventualnim ispravkama ili retransmisijom.
3. **Mrežni sloj:** Prenosi pakete na nivou mreže preko rutera. Na ovom sloju se vrši rutiranje.
4. **Transportni sloj:** Prenosi segmente podataka od posiljaoca do primaoca (eng. end-to-end layer).
5. **Sloj sesije:** Omogućava različitim mašinama da uspostavljaju sesije između njih.
6. **Sloj prezentacije:** Odgovoran za sintaksu i semantiku informacija koje se prenose.
7. **Aplikativni sloj:** Pruža usluge krajnjim korisnicima. Rad sa porukama.

### 3.1.2 TCP/IP model

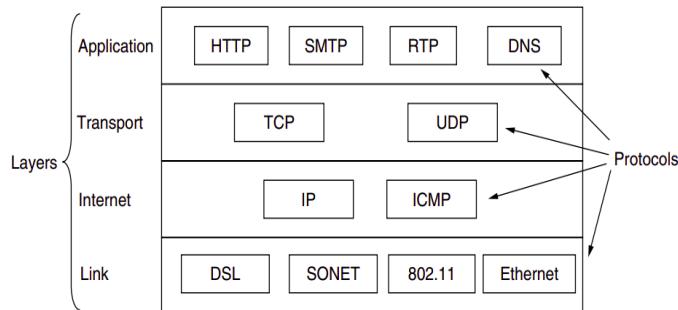


**Slojevi (od najnižeg):**

1. **Sloj veze:** Ne predstavlja konkretno sloj, već interfejs. Definiše kako koja veza radi.
2. **Internet sloj:** Odgovara sloju veza za OSI model.
3. **Transportni sloj:** Odgovara transportnom sloju za OSI model.
4. **Aplikativni sloj:** Najviši sloj koji predstavlja interfejs ka korisnicima.



### Protokoli:



#### 3.1.3 TCP/IP ili OSI model

##### Zašto se OSI model ne koristi?

- Loš tajming za investicije.
- Loša tehnologija (sloj sesije i prezentacije su skoro prazni).
- Loša implementacija.
- Loša politika.

Model Osi je svakako doprineo u razvoju mreža.

##### Mane TSP/IP sloja:

- Nije dobar za izradu novih tehnologija, jer ne razlikuje koncepte servisa, interfejsa i protokola.
- Loš za definisanje stek protokola (primer. nemoguće je definisati stek protokola za bluetooth)
- Ne postoji razlika između fizičkog sloja i sloja veze podataka.

Model TCP/IP se koristi u praksi.

## 3.2 jedinice podataka

- Aplikativni - poruka
- Transportni - segment
- Mrežni - paket
- Sloj veze - okvir
- Fizički - bit

### 3.3 организације за стандарде

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

## 4 Физички слој: улога, pojednostavljeni model, каšњења, BDP, примери.

### 4.1 улога

Подаци који се преносе су дигитални (битови). Џице преносе аналогни (физички) сигнал. Ово је управо улога физичког слоја тј. физички слој шалje поруке путем комуникационог канала.

### 4.2 pojednostavljeni model

**Кarakteristike:**

- Проток: брзина преноса података (б/с, битова по секунди).
- Каšњење: време потребно да порука стigne на циљну адресу (секунди).
- Да ли канал еmituje сигнал или не.
- Раподела вероватноћа грешака.
- ...



### 4.3 Каšњење

**Каšњење** представља време потребно да порука стigne на циљну адресу. Мerna единица за каšњење је секунда (нajćešće se koristi milisekunda). Састоји се из два дела:

- **Kašnjenje prenosa (Transmission Delay):** Vreme potrebno da se M-bitovna poruka postavi na komunikacioni kanal:  
 $T\text{-delay} = M \text{ (b)} / B \text{ (b/s)} = M/B \text{ (s)}$ , gde je B brzina postavljanja podataka na komunikacioni kanal.
- **Kašnjenje propagacije (Propagation Delay):** Vreme potrebno da bitovi prođu kroz komunikacioni kanal:  
 $P\text{-delay} = \text{dužina kanala} / \text{brzina signala} (\frac{2}{3}c) = X \text{ (s)}$
- **Ukupno kašnjenje:**  
 $L = T + P = M/B + P$

#### 4.4 BDP

Količina podataka prisutnih na kanalu u nekom momentu je **BDP** (Bandwidth-Delay Product) tj. **proizvod protoka i kašnjenja**. Ako se podatak posmatra kao materija, onda bi ovo bila zapremina (količina) materije:

$$BDP = B \cdot D$$

Vrednost BDP-a je mala za lokalne mreže, a velika za velike „debele mreže“.

#### 4.5 Primeri (kašnjenje i BDP)

Primer kašnjenja:  $P = 5 \text{ ms}$ ,  $B = 56 \text{ Kb/s}$ ,  $M = 1250 \text{ B}$   
 $\Rightarrow P = 5/1000 \text{ s}$ ,  $B = 56*1024 \text{ b/s}$ ,  $M = 1250*8 \text{ b}$   
 $\Rightarrow L = 0.005 \text{ s} + 1250*8 \text{ b} / 56*1024 \text{ b/s}$   
 $\Rightarrow L = 0.005 \text{ s} + 0.174 \text{ s}$   
 $\Rightarrow L = 0.179 \text{ s}$   
 $\Rightarrow L = 179 \text{ ms}$

Primer BDP-a:  $B = 40 \text{ Mb/s}$ ,  $D = 50 \text{ ms}$   
 $\Rightarrow B = 1024*1024*40 \text{ b/s}$ ,  $D = 0.05 \text{ s}$   
 $\Rightarrow BDP = 1024*1024*40 \text{ b/s} * 0.05 \text{ s}$   
 $\Rightarrow BDP = 2097152 \text{ b}$   
 $\Rightarrow BDP = 262144 \text{ B}$   
 $\Rightarrow BDP = 256 \text{ KB}$   
 $\Rightarrow BDP = 0.25 \text{ MB}$   
Ovo se smatra visokim BDP-om.

### 5 Žičani i optički komunikacioni medijumi

#### 5.1 UTP kablovi

**Upredene parice (UTP - Unshielded Twisted Pair)** su veoma čest tip kanala. Sastoje se iz četiri upredene parice napravljene od bakra. Uvrtanjem bakarnih kablova se smanjuje smetnja. Koristi se u telekomunikacija i za ADSL internet pristup. Danas se koristi peta kategorija. Brzina prenosa je do 1 Gbps (full duplex). Parice su osjetljive na elektromagnetne smetnje.

#### 5.2 Koaksijalni kablovi

**Koaksijalni kablovi (Coaxial Cable)** su bolji komunikacioni medijum od upredenih parica po boljim performansama i manjim smetnjama,

ali teži za održavanje. Sastoje se od bakra u sredini, preko kojeg ide izolator, pa preko kojeg opet ide bakar ili aluminijum i spolja je obavljen dodatnom izolacijom. Uglavnom idu uz kablovske.

### 5.3 Instalacija za prenos struje

Već postoje u okviru naših kuća, ali imaju loše performanse.

### 5.4 Optički kabal

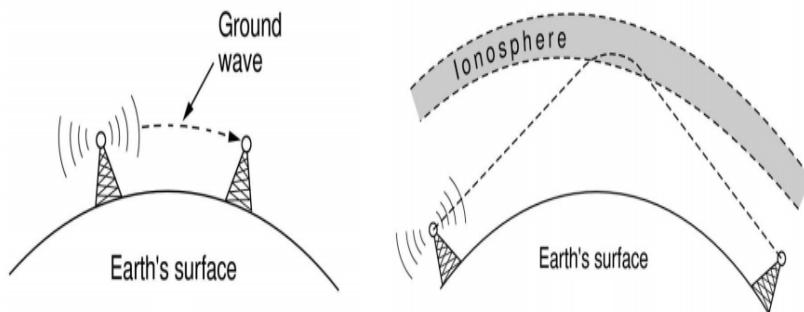
**Optički kablovi (Fiber-Optic Cable)** se prave od dugačkih, tankih i čistih vlakni stakla. Imaju ogroman protok zbog opsega frekvencije i predstavljaju odličan komunikacioni medijum za velike udaljenosti zbog malog slabljenja signala. Postoje dve vrste: višemodalno (za manje udaljenosti) i unimodalno (za veće udaljenosti). Prednosti staklenih vlakana u odnosu na bakar su: bolje performanse, manja osetljivost na elektromagnete smetnje i bolji signal na većim udaljenostima.

## 6 Bežični komunikacioni medijumi

Prednost i manja bežičnog prenosa je što se signal šalje u svim pravcima. To znači da može da ima više primaoca, ali signali sličnih frekvencija mogu da se mešaju i samim tim oštećuju.

### 6.1 Radio talasi

**Radio talasi (Radio Waves)** se lako generišu, mogu da putuju daleko i prolaze kroz zidove. Na malim frekvencijama (VLF - Very Low Frequency, LF - Low Frequency, MF - Medium Frequency) talasi prate zakrivljenost Zemlje, dok na velikim frekvencijama se odbijaju od jonsfere.



### 6.2 Mikrotalasi

**Mikrotalasi (Microwaves)** imaju veliki frekventni opseg i koriste se često za zatvorene namene poput *wifi*, kao i za otvorene poput 3G i sateliti. Signal mikrotalasa slabi i reflektuje se od objekata okruženja. Jačina varira od udaljenosti, sabiranja signala, itd...

## 6.3 Svetlost

Svetlost kao talas se može koristiti kao komunikacioni medijum, ali to je više za hrabre koji vole avanture.

## 6.4 Žičani i bežični prenos

Prednost bežičnih u odnosu na žičane kanale:

- Jednostavne i jeftine;
- Prirodno podržavaju mobilnost;
- Prirodno podržavaju emitovanje.

Mane bežičnih u odnosu na žičane kanale:

- Mešanje signala se mora razrešiti;
- Jačina signala (samim tim i protok) jako variraju.

Prednost žičanih u odnosu na bežične kanale:

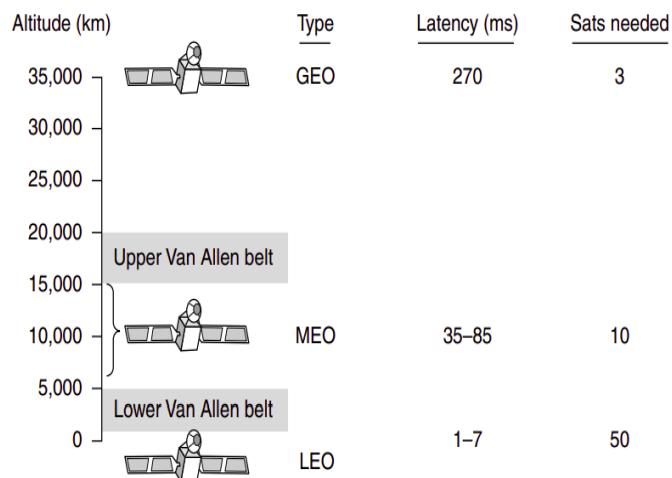
- Lako se projektuje fiksni protok duž odabranih ruta.

Mane žičanih u odnosu na bežične kanale:

- Skupo za postavljanje, posebno na većim udaljenostima;
- Nisu projektovani za mobilnost i emitovanje.

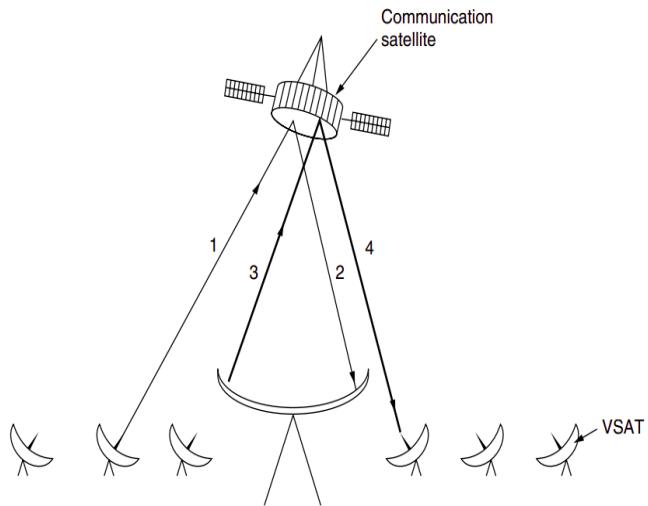
## 7 Komunikacioni sateliti

Sateliti su efikasni za emitovanje i komunikaciju „bilo kad, bilo gde“. Osnovna podela satelita je po udaljenosti od zemlje gde razlikujemo **LEO** (Low Earth Orbit), **MEO** (Medium Earth Orbit) i **GEO** (Geostationary orbit) satelite. Postoje pojasevi Zemljine atmosfere gde ne mogu biti postavljeni sateliti. Ti pojasevi se nazivaju *Van Allen pojasevi* i svaki satelit koji se tu postavi biva brzo uništen od strane čestica. Postoje dva takva pojasa i nalaze se između LEO i MEO, MEO i GEO što je jedan od razloga za ovakvu podelu.



## 7.1 Geostacionarni sateliti (GEO)

Ovi sateliti orbitiraju 35km iznad fiksne lokacije. Mikrostanice **VSAT** (Very Small Aperture Terminals) dobijaju i šalju signal ka centralnoj stanicu HUB koja služi za komunikaciju između VSAT mikrostanica. Signali se šalju uz pomoć GEO satelita. Primer: emitovanje televizijskog programa.



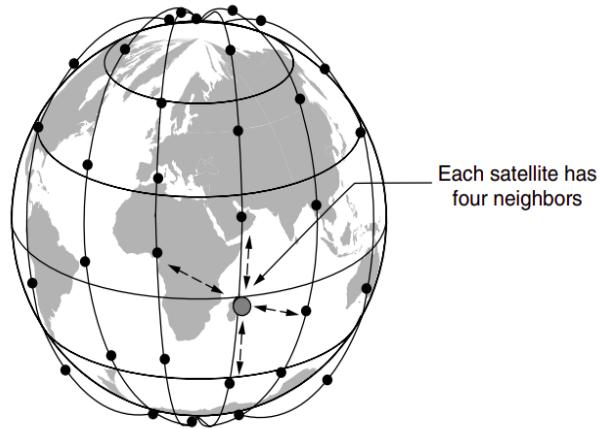
## 7.2 Srednje visoko-orbitni sateliti (MEO)

Postavljeni su između dva Van Allen pojasa i koriste se uglavnom za GPS

## 7.3 Nisko-orbitni sateliti (LEO)

Zbog velike brzine potreban je veliki broj ovakvih satelita za kompletan sistem. Ovi sateliti imaju manje kašnjenje zbog male udaljenosti od zemlje. Projekat koji je pokrenuo masovno korišćenje ovih satelita je *Iridium* (77 satelita i 77 elemenata u tablici, a zapravo je na kraju poslat 66 satelita).

Postavljeni su kao 6 lanaca oko planete. Imaju brži odziv od GEO satelita.



## 7.4 Sateliti i optika

**Prednosti satelita u odnosu na optiku:**

- Nakon uspostavljenog kompletнnog sistema se može komunicirati bilo gde;
- Emitovanje na velika područja;
- Instalacija nije skupa i komplikovana kao kod optike.

**Prednosti optike u odnosu na satelite:**

- Sateliti imaju ograničen protok i mešaju se signali;
- Ogroman protok duž velikih udaljenosti.

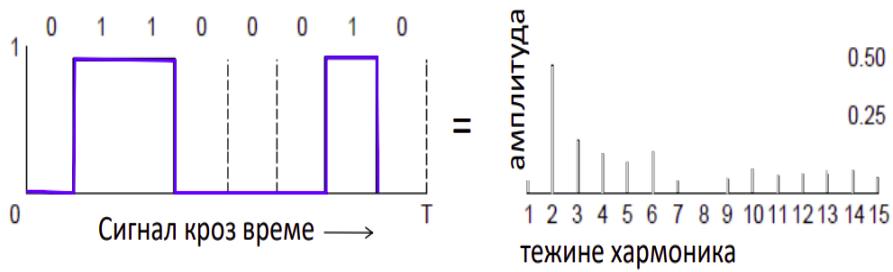
## 8 Signalni prenos, frekvencijska reprezentacija, signal u žičanim, optičkim, bežičnim medijumima.

### 8.1 Prenos - Furijeova analiza

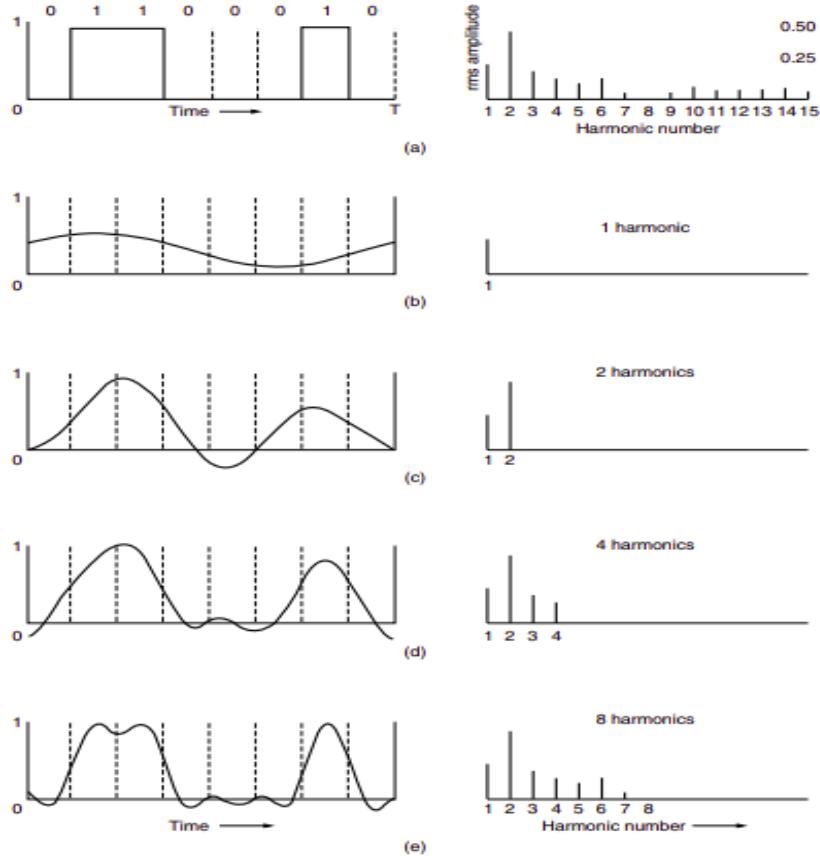
Svaka periodična funkcija  $g(t)$  može da se predstavi kao beskonačna suma sinusa i kosinusa:

$$g(t) = \frac{1}{2}c + \sum_{n=1}^{\infty} a_n \sin(2\pi n f t) + \sum_{n=1}^{\infty} b_n \cos(2\pi n f t)$$

Vrednosti  $a_n$  i  $b_n$  su sinusna i kosinusna amplituda  $n$ -te harmonijske težine, i mogu se lako izračunati (kao i konstanta  $c$ ). Takođe iz Furijeovog reda je moguće rekonstruisati nazad funkciju. Pošto su svi podaci konačni, možemo ih posmatrati kao da su periodični što je dovoljan uslov za Furijeovu transformaciju. Sa desne strane su prikazane vrednosti  $\sqrt{a_n^2 + b_n^2}$ :



U praksi ne možemo uzeti beskonačno frekvencija. Dovoljno je uzeti broj frekvencija tako da može da se rekonstruiše funkcija. Sa slike(dole) se vidi da pod (e) je dovoljno precizno i nije potreban veći broj frekvencija (manji skup frekvencija = manji protok):



## 8.2 Prenos preko žice

Šta se dešava sve sa signalom dok prolazi kroz žicu?

- Signal kasni (brzina je  $\frac{2}{3}c$ , nije beskonačna);

- Signal slabi (srazmerno udaljenosti)
- Frekvencije iznad neke granice brže slabe;
- Dešava se šum (zbog spoljašnjih efekata).

### 8.3 Prenos preko svetlosti

Svetlost se prenosi sa veoma malim gubitkom u tri široka frekvetna opsega.

### 8.4 Prenos preko bežičnih medijuma

Zbog velikih frekvencija bežičnih prenosa, nije moguće digitalni signal direktno kodirati u analogni, već se koristi koncept **signala nosača** (signal putuje jako brzo, ali slabi sa kvadratom rastojanja). Takođe, problem kod bežičnih komunikacija je što se signali mešaju (ako su dovoljno blizu). Dodatni otežavajući efekti su to što je propagacije bežičnog signala složena i zavisi od okruženja, i karakteristike zavise od frekvencije (primer: problem sa sabiranjem odbijenih talasa). Signali mogu da se odbijaju od objekata i putuju kroz više nezavisnih putanja. Kada signali stignu do primaoca oni mogu da budu pogrešno sabrani.

## 9 Modulacija i multipleksiranje signala.

### 9.1 Modulacija

Modulacija je način predstavljanja digitalnih informacija u okviru fizičkog medijuma.

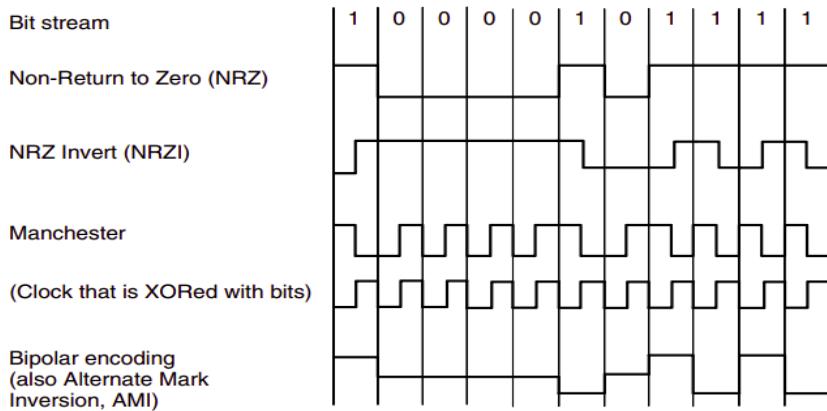
#### 9.1.1 Direktna modulacija (baseband)

Primer jednostavne šeme je **NRZ (Non-Return to Zero)** šema, gde visoki napon (+V) predstavlja 1, a niski napon (-V) predstavlja 0. Može se koristiti više od dva nivoa (primer: prenosi se po 2 bita). To zavisi od tehnoloških mogućnosti medijuma.

**Problem kod dekodiranja:** Prilikom dekodiranja časovnici nisu toliko precizni u slučaju dugih jedinica i nula. Niz 15 nula dosta liči na niz 16 nula osim ako časovnik nije poprilično precizan.

- **Odvjeni kanal za časovnik:** Koriste se dupli kanali, gde se na drugom kanalu šalje signal časovnika.
- **Mančester kodiranje:** Problem prethodnog rešenja je u tome što se bezveze koristi još jedan kanal, gde on može da se koristi za prenos podataka. Bolje rešenje je da se pomeša signal časovnika sa signalom podataka koristeći XOR operater. Ovo kodiranje se naziva Mančester kodiranje. ([Manchester Encoding](#))
- **NRZI (Non-Return to Zero Inverted):** Kodiramo 1 kao prelaz ( $1 \rightarrow 0, 0 \rightarrow 1$ ), a 0 kao neprelaz ( $0 \rightarrow 0, 1 \rightarrow 1$ ). Standard za USB koristi ovakav način kodiranja. Ovom metodom se samo rešava problem uzastopnih jedinica, ali ne i uzastopnih nula.

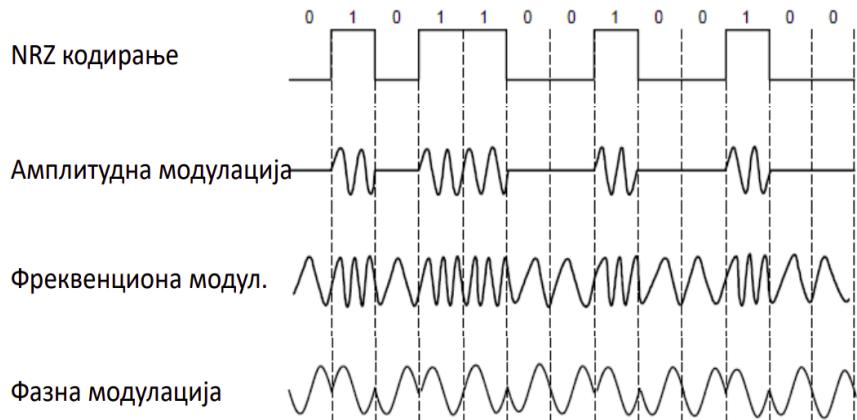
- **4B/5B:** Svaki niz četiri bita se preslikava u niz 5 bitova. Preslikavanje se bira tako da nikad nema više od tri uzastopne nule. Ovim se koristi 25% više bitova da se prenese poruka (i dalje bolje od Mančestera koji tehnički koristi 100% više).
- **Scrambling:** Generiše se pseudo nasumični broj i XOR-uje se sa podacima u nadi da neće da se desi niz dugačkih nula.



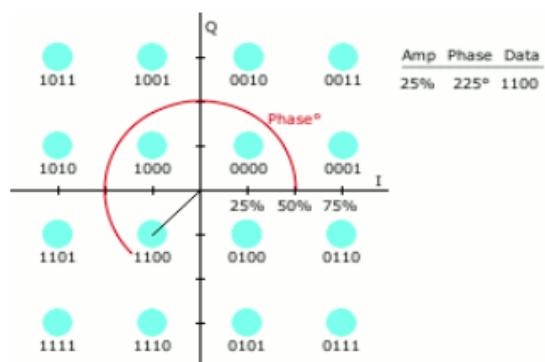
### 9.1.2 Modulacija preko nosača (passband)

Direktna modulacija je moguća samo kod žičanih komunikacionih kanala. Kod bežičnih signala se koristi modulacija preko nosača, gde se signal kodira indirektno. Signal nosač oscilira na na željenoj frekvenciji, a potom se modulira promenom amplitude, frekvencije ili faze.

- **ASK (Amplitude Shift Keying)** - dve različite amplitude se koriste za reprezentaciju nule i jedinice.
- **FSK (Frequency Shift Keying)** - slično, koriste se različite frekvencije.
- **PSK (Phase Shift Keying)** - slično, koriste se različite faze.
- Takođe se može se koristiti kombinacije ASK, FSK i PSK.

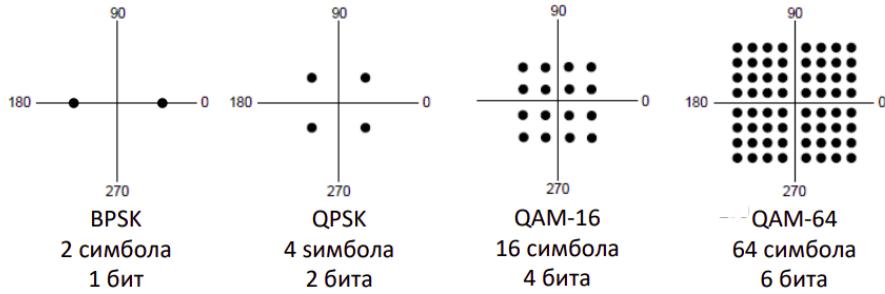


**Kombinovana amplitudno-fazna modulacija(QAM-16):**



#### Ostale kombinacije:

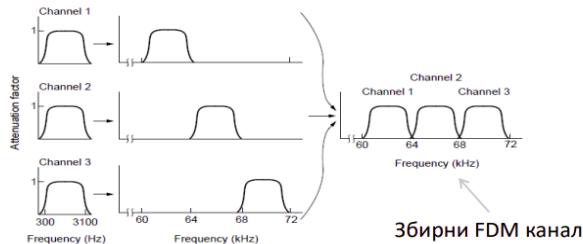
- BPSK: variranje faze, dve faze (jedan bit).
- QPSK: varivanje faze, četiri faze (dva bita)
- QAM-64: slično kao QAM-16, ali kodira 6 bitova.



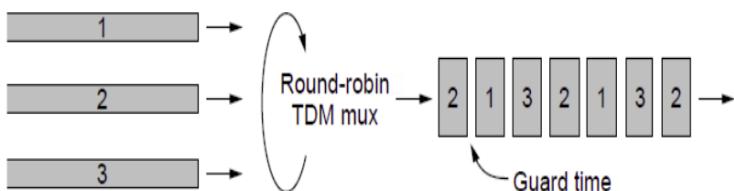
## 9.2 Multipleksiranje signala

Multipleksiranje se bavi deljenjem kanala između više korisnika. Analogan iz života problem: U sobi ima puno ljudi koji trebaju međusobno da komuniciraju:

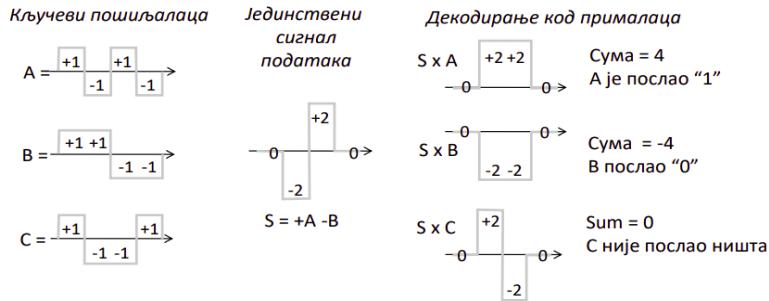
- **FDM (Frequency Division Multiplexing):** Korisnici kanala su postavljeni na različite frekvencije. U sobi punoj ljudi ovo bi bilo analogno da se ljudi fokusiraju na ljude koji pričaju glasno, srednje ili slabo.



- **TDM (Time Division Multiplexing):** Kanal se deli vremenski, gde se korisnici drže fiksнog rasporeda (upotrebljava se u sistemima fiksne telefonije). U sobi punoj ljudi ovo bi značilo da svi čute dok drugi priča. Primer prioriteta: Round robin.



- **CDMA (Code Division Multiple Access):** Korisnicima se dodjeljuju ključevi. Ključevi su međusobno ortogonalni. Na ortogonalni signal se primenjuje ključ (skalarни proizvod). U sobi punoj ljudi ovo bi značilo da ljudi pričaju različitim jezicima.



## 10 Prirodna ograničenja prenosa signala.

Kako odrediti da li je prenos signala dobar?

Ako brzina prenosa signala blizu prirodnog ograničenja, onda je sistem dobro realizovan. Pojmovi:

- **B (Bandwidth, protok)** - ograničava brzinu promena (karakteristika kanala);
- **S (Signal power, jačina signala)** i **N (Noise power, jačina šuma)**. Jačina šuma ograničava broj razlučivih nivoa signala (karakteristika primaoca). **Napomena:** Prilikom računanja limita i kapaciteta koriste prosečna jačina signala i prosečna jačina šuma.

**Najkvistov limit:** Maksimalan broj promena je  $2B$ . Ako postoji  $V$  nivoa signala, onda je maksimalan protok u bitovima:  $R = 2B \cdot \log_2 V$ . Za  $V=2$  je to  $R = 2B$ . **Napomena:** Ovo važi za kanale bez šuma.

**Šenonov kapacitet:** Broj razlučivih nivoa signala zavisi od odnosa jačine signala i jačine šuma  $SNR = \frac{S}{N}$  (**SNR - Signal Noise Ratio, odnos šum-signal**). Vrednost SNR-a može samo da varira, zbog čega se on računa u decibilima tj.  $SNR_{dB} = 10 \log_{10} \frac{S}{N}$ . Šenonov kapacitet (maksimalan broj bitova po sekundi) je:  $C = B \cdot \log_2 (1 + \frac{S}{N})$  b/s. Broj razlučivih signala se dobija iz odnosa:  $\frac{S+N}{N} = 1 + \frac{S}{N}$ . **Napomena:** Da bismo duplirali Šenonov kapacitet nije dovoljno duplirati protok, jer je šum zavisan od protoka.

Kod žičanih komunikacija se može projektovati ciljni SNR i samim tim i ciljni prenos. Kod bežičnih SNR može drastično da varira za dato B (do 60db). Kod bežičnih komunikacija se mora „živeti“ na visokim varijacima.

## 11 Pregled relevantnijih sistema komunikacije.

### 11.1 Sistem fiksne telefonije

Sistem fiksne telefonije (landline, fixed-line) se sastoji iz tri glavne komponente:

- Lokalne konekcije (upredene parice koje idu do zgrada)
- Međumesne konekcije (optički kablovi)
- Centrale (gde se pozivi preusmeravaju)

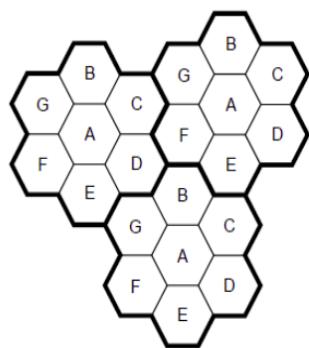
Preko ovog sistema se realizuje DSL (ADSL).

## 11.2 Sistem mobilne telefonije

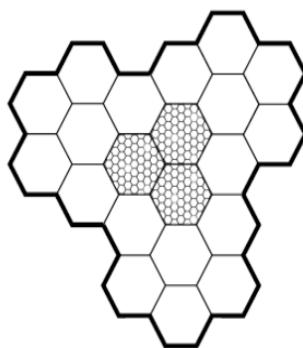
Generacije:

- **1G: analogni glas**, FM modulacija (kao kod radija), odvojena frekvencija za slanje i primanje
- **2G: digitalni glas**, GSM (Global System for Mobile communication), QPSK modulacija
- **3G: digitalni glas i podaci**, UMTS (Universal Mobile Telecommunications System), CDMA
- **4G: digitalni glas i podaci**, LTE (Long Term Evolution), OFDM (napredni FDM)

**Organizacija baznih stanica:** Svaki mobilni korisnik koristi ćelijsku frekvenciju. Pri napuštanju jedne ćelije ulazi u drugu ćeliju (handoff). Iste frekvencije se mogu pronaći u nesusednim ćelijama. Za podržavanje većeg broja korisnika, obično se ograničava geografski prostor ćelije.



Расподела фреквенција



Мање ћелије за гушће насељене области

## 11.3 Internet preko kablovske

Internet kabl može da koristi već postojeću infrastrukturu za kablovsku televiziju. Drugačija je topologija u odnosu na telefonski sistem. Ovde organizacija podseća na topologiju magistrale. Preuzimanje i postavljanje podataka koriste opsege koji se ne koriste za gledanje TV programa.

## 11.4 Kablovska ili (A)DSL

Prednosti kablovske:

1. Koristi Koaksijalne kablove ka korisnicima koji imaju bolji protok od upredenih parica koje koristi ADSL.

Prednosti (A)DSL-a:

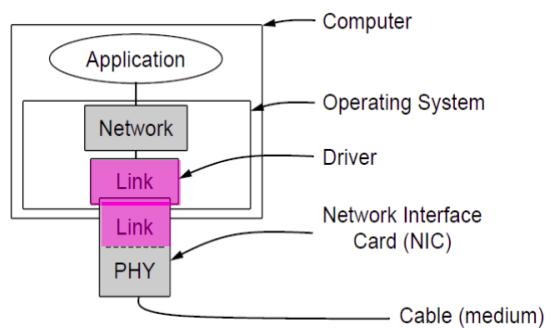
1. Podaci se ne šalju svima kao kod kablovske, već posebnom koristniku.
2. Protok nije deljen i ne varira toliko.

## 12 Sloj veze: uloga, komunikacija sa slojem ispod i iznad, kratko objašnjenje spiska aktivnosti na sloju veze.

Od fizičkog sloja imamo tok podataka (bitova). Želimo da šaljemo podatke kao celine tj. okvire fiksirane veličine koristeći mogućnosti fizičkog sloja. Poslovi sloja veze:

- Pruža dobro definisan interfejs mrežnog sloju;
- Proverava da li je došlo do greške tokom transmisije;
- Vodi računa o brzini protoka (ako je primalac sporiji od pošiljaoca).

Sloj veze je obično realizovan delom na mrežnoj kartici, a drugim delom na nivou operativnog sistema.

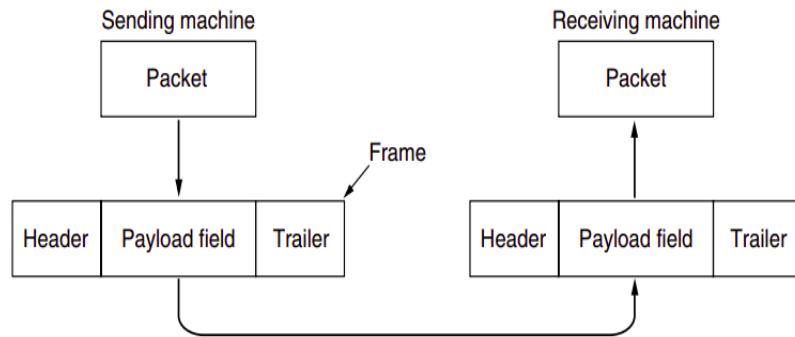


### Servisi koje sloj veze nudi mrežnom sloju:

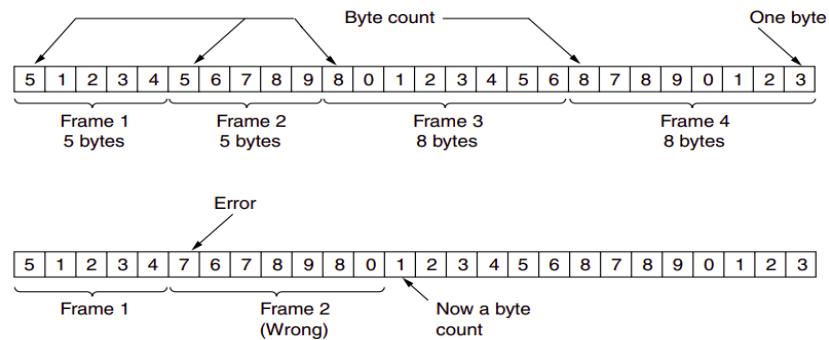
- **Unacknowledged connectionless service:** Slanje nezavisnih okvira bez provere da li je stigao ispravan okvir do primaoca. Primer: Ethernets.
- **Acknowledged connectionless service:** Slanje nezavisnih okvira sa proverom da li je stigao ispravan okvir do primaoca. Primer: WiFi.
- **Acknowledged connection-oriented service:** Uspostavlja se veza pre slanja niza okvira, gde se vodi računa za svaki okvir da li je stigao do primaoca. Retko se koristi.

## 13 Uokvirivanje u sloju veze

U teoriji uokvirivanje je potpuno nevidljivo fizičkom sloju. Ipak u praktici fizički sloj pomaže u identifikaciji okvira.

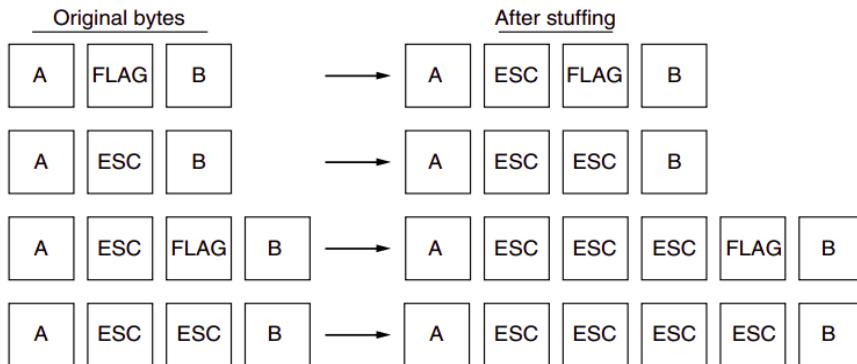


**Brojanje bajtova:** Okvir započinjemo sa poljem o njegovoj dužini. U slučaju greške na polju o njegovoj dužini dolazi do neusaglašavanja i ceo tok se poremeti.



**Umetanje bajtova:** Koristi se indikatorska oznaka, zastavica „FLAG“ u vidu bajta koja označava početak i kraj okvira. Potencijalni problem:

- **Šta ako se indikator nalazi u podacima?** Koristi se „ESC“ (escape) bajt sa leve strane „FLAG“ zastavice.
- **Šta ako se u podacima nalazi i „ESC“?** Opet, sa leve strane se doda „ESC“. Prijemnik uvek usklanja prvi „ESC“ bajt, ako nađe na „ESC“ bajt, i čita sledeći u nizu bajt. U slučaju da imaju dva „ESC“ za redom, onda bi pročitao tačno jedan. Ako nađe na četiri uzaspona „ESC“ bajta, onda prijemnik čita tačno dva „ESC“ bajta.



**HDLC (High Level Link Control) protokol (umetanje bitova):**  
Početak i kraj okvira se označavaju sa nizom bitova „01111110“ (zastavica). Ako se u podacima nalazi pet uzastopnih jedinica, onda se nakon petog bita automatski ubaci jedna nula. U tom slučaju je nemoguće da se zastavica nađe u podacima. Primalac, analogno, nakon svakih pet uzastopnih jedinica obriše jednu nulu.

Poređenje metoda na podacima dužine 100 bajtova:

- Bez zastavica se šalje 100 bajtova.
- Sa umetanjem bajtova u najgorem slučaju su sve specijalni bajtovi i šalje se 200 bajtova umesto 100 bajtova (do 100% povećanja).
- Sa umetanjem bitova u najgorem slučaju je povećanje oko 12.5%, jer se uz svaki bajt doda po jedan bit.

## 14 Kodiranje grešaka u sloju veze.

Zbog sporednih efekata dolazi do šuma u signalu zbog čega nastaju greške u podacim. Postoje više načina za rešavanje ovakvog problema u zavisnosti od tipa šuma i njegove frekvencije. Postoje dva glavna pristupa: korekcija greške i detekcija greške zajedno sa retransmisijom. Kod oba ova pristupa se dodaju redundantni podaci u okviru osnovnih podataka. Na osnovu redundantnih podataka se zaključuje da li je došlo do greške. Cilj je rešiti što više grešaka sa što manje redundantnosti.

**Naivni pristup:** Poslati dve kopije za svaku poruku. Ako poruke nisu iste, onda je došlo do greške. Ovaj pristup može da detektuje grešku, međutim ako je došlo do greške ne možemo znati koja je kopija ispravna.

U opštem slučaju kodna reč se sastoji iz D bitova podataka i R kontrolnih bitova. Proces:

- Pošiljalac izračuna R na osnovu D koristeći neku funkciju  $f$  tj.  $R=f(D)$ .
- Pošiljalac šalje D+R bitova.
- Primalac prihvata D+R bitova sa potencijalnim greškama.
- Primalac računa R' na osnovu dobijenog D na isti način.
- Ako se R i R' ne poklapaju, onda je došlo do greške.

**Intuicija:** Skup validnih kodnih reči je dosta manji od skupa mogućih reči. Slučajno odabrana reč ima malu šansu da bude ispravna. To znači da ako se desi greška, onda je mala šansa da će novodobijena reč biti validna.

### 14.1 Hamingovo rastojanje

Hamingovo rastojanje je minimalan broj inverzija koji je potreban da se od jedne validne reči dobije druga validna reč. Hamingovo rastojanje koda je minimalno Hamingovo rastojanje između parova validnih kodnih reči.

- **Detekcija grešaka:** Da bi se pouzdano otkrilo  $d$  grešaka ( $d$  izmenjenih bitova), Hamingovo rastojanje koda mora biti najmanje  $d+1$ . Tada je nemoguće da  $d$  jednobitnih grešaka promene validnu kodnu reč u neku drugu validnu reč.
- **Korekcija grešaka:** Za kod sa Hamingovim rastojanjem  $2d + 1$ , najviše  $d$  grešaka se uvek može ispraviti do najbliže ispravne validne reči. Primer:
  - Validne reči: 0000000000, 0000011111, 1111100000, 1111111111.
  - Hamingovo rastojanje: 5,  $(2 \cdot 2 + 1) \Rightarrow$  može se napraviti bilo koja dvobitna greška.
  - Primer: 0000000111 se ispravlja u 0000011111 ako je dvobitna greška, ali ako je trobitna greška onda je možda 0000000000 poslata reč.

## 15 Detekcija grešaka u sloju veze.

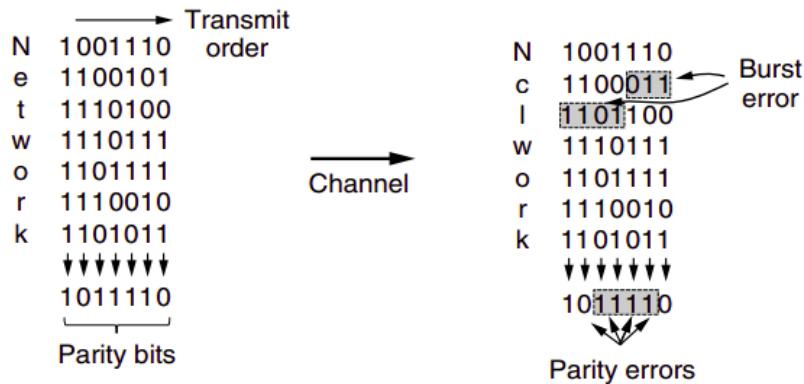
Na mrežama gde je šum manji kao što su optički kablovi ili bakar visokog kvaliteta, umesto korekcije kodova se više isplati detekcija uz retransmisiju.

### 15.1 Provera parnosti:

Koristi se jedan dodatni bit za proveru parnosti. Ovaj bit se računa kao zbir podataka po modulu 2, odnosno XOR svih bitova u podacima. Ako se desi jednobitna greška onda se greška može detektovani. Ako se desi greška dva puta onda se greška ne može detektovati.

### 15.2 Kontrolni zbir:

Vrši se sumiranje podatak po kolonama (umesto po redovima). Ovaj metod se naziva „**interleaving**“ (računanje sume drugačijim redom od slanja podataka). Suma se dodaje kao poslednji red. Ova metoda ima bolju detekciju od provere parnosti. Kontrolna suma na „**burst error**“ (grupne greške, eksplozija grešaka):



### 15.3 Internet kontrolni zbir

Bolja verzija kontrolnog zbirja svodi se na korišćenje nepotpunog komplementa. Kontrolni zbir ima 16 bitova i predstavlja nepotpuni komplement sume reči po kolonama. Algoritam:

- Slanje:
  1. Složiti podatke kao 16 bitne reči jednu ispod druge;
  2. Sabrati 16-bitne reči u nepotpunom komplementu;
  3. Negirati dobijenu sumu;
  4. Rezultat predstavlja kontrolnu sumu.
- Prihvatanje:
  1. Složiti podatke kao 16 bitne reči jednu ispod druge;
  2. Sabrati 16-bitne reči u nepotpunom komplementu;
  3. Negirati dobijenu sumu;
  4. Nema greške ako je dobijeni rezltat 0, u suprotnom ima greške.

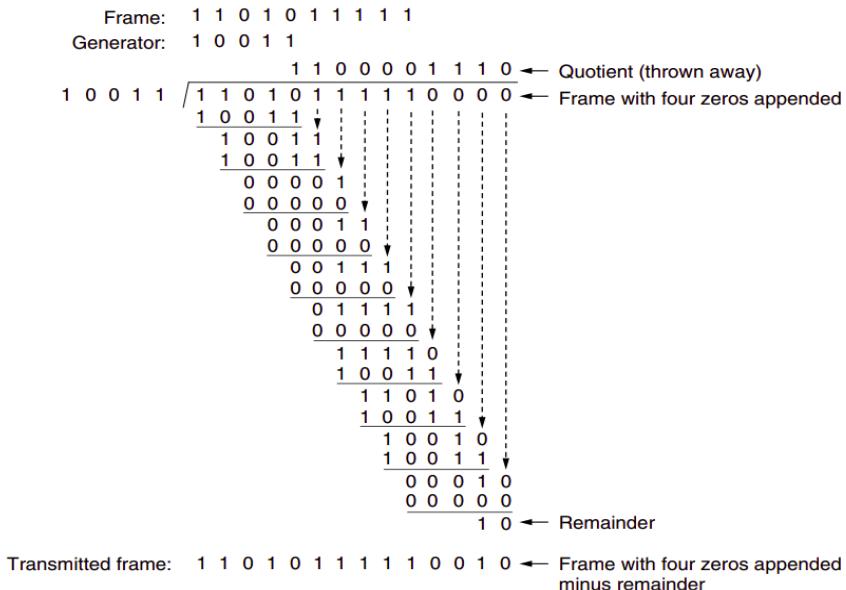
<b>slanje:</b> podaci $  \begin{array}{r}  0001 \\  \textbf{f}203 \\  \textbf{f}4\textbf{f}5 \\  \textbf{f}6\textbf{f}7 \\  + (0000) \\  \hline  2\text{dd}\text{f}0  \end{array}  $ $  \begin{array}{r}  \text{d}\text{d}\text{f}0 \\  + 2 \\  \hline  \text{d}\text{d}\text{f}2  \end{array}  $ kontrolna suma <span style="border: 1px solid black; padding: 2px;">220d</span>	<b>prihvatanje:</b> primljeni podaci $  \begin{array}{r}  0001 \\  \textbf{f}203 \\  \textbf{f}4\textbf{f}5 \\  \textbf{f}6\textbf{f}7 \\  + 220d \\  \hline  2\text{ff}\text{fd}  \end{array}  $ $  \begin{array}{r}  \text{f}\text{ff}\text{d} \\  + 2 \\  \hline  \text{f}\text{ff}\text{f}  \end{array}  $ <span style="border: 1px solid black; padding: 2px;">0000</span> nema greške
--	--

## 15.4 Ciklična provera redundanse (CRC) ili polinomijalni kodovi

Ideja je da se niz bitova tretira kao polinom sa koeficijentima 0 ili 1. Bitovski kod dužine  $k$  se posmatra kao polinom stepena  $k-1$  tj.  $b_0 + b_1x + b_2x^2 + \dots + b_{k-1}x^{k-1}$ , gde je prvi bit (nulti bit) skroz desno u binarnom zapisu. Primer: 110001 odgovara polinomu  $1x^5 + 1x^4 + 0x^3 + 0x^2 + 0x^1 + 1$ . Sve aritmetičke operacije su po modulu 2 (sabiranje, oduzimanje, deljenje, množenje).

Potrebno je da se pošiljalac i primalac dogovore oko generatora polinoma  $G(x)$ . Ideja je da se doda  $k$  (dužina  $G(x)$ ) bitova (CRC) na kraj tako da polinom bude deljiv sa  $G(x)$  i da se pošalje novodobijeni kod. Primalac vrši deljenje dobijenog koda sa  $G(x)$ . Ako se dobije ostatak različit od nule, onda je došlo do greške. Algoritam:

1. Neka je  $k$  dužina polinoma  $G(x)$ . Dodaje se  $k$  nula na kraj (desno) podatka  $D$  dužine  $n$  (dodavanje  $k$  nula sa desne strane na bitovski zapis odgovara množenju polinoma sa  $x^k$ ), kojem odgovara polinom  $M(x)$ . Novodobijeni kod je dužine  $n+k$  i odgovara polinomu  $x^k M(x)$ .
2. Izvrši se deljenje  $x^k M(x)$  sa  $G(x)$ . Dobije se ostatak  $R(x)$  dužine (najviše)  $k$ .
3. Šalje se podatak koji odgovara polinomu  $x^k M(x) + R(x)$  (doda se polinom  $R(x)$  sa desne strane  $M(x)$ ) koji je deljiv sa  $G(x)$ .



## 15.5 Detekcija grešaka u praksi

Od generatora zavisi kvalitet CRC detekcije. Postoje standardni CRC-32 brojevi koji se koristi u praksi. Ciklična provera se koristi za Ethernet, 802.11, ADSL, kablovsku, ... Kontrolni zbir (iako je lošiji) se koristi za protokole na višim slojevima kao što su IP, TCP, UDP, itd. Kontrola parnosti se slabo koristi.

## 16 Korekcija grešaka u sloju veze

Korekcija greške se više primenjuje za bežičnu komunikaciju, gde je retransmisija skuplja i više se isplati pokušati sa izrvšavanjem korekcije.

### 16.1 Hamingov kod za korekciju HD=3

Koristi n bitova podataka i k kontrolnih bitova, gde važi veza  $n = 2^k - k - 1$  (primer.  $n = 4$ ,  $k = 3$ ). Ideja je da se stavi kontrolni bit na pozicije koje su stepen dvojke (1, 2, 3,...). Kontrolni bitovi se i dalje uključuju kao bitovi za kontrolu parnosti uz bitove koji imaju vrednost (po indeksu) 1 na (i)-tom mestu. Primer ( $n = 4$ ,  $k = 3$ ):

1: 001, 2: 010, 3: 011, 4: 100, 5: 101, 6: 110, 7: 111.

*Grupisanje:*

- Po bitu najmanje težine (desnom) tj. prvom bitu: 1, 3, 5 i 7;
- Po drugom bitu: 2, 3, 6, 7;
- Po trećem bitu se grupišu 4, 5, 6, 7.



Primer: Neka su podaci  $D = 0101 \Rightarrow$  tri kontrolna bita se koriste! Podaci za slanje su oblika: ? ? 0 ? 1 0 1, tj.  $p_1p_20p_3101$ , gde su  $p_1, p_2, p_3$  kontrolni bitovi parnosti. U opštem slučaju su podaci oblika:  $p_1p_2d_1p_3d_2d_3d_4$ , gde su  $d_1, d_2, d_3, d_4$  bitovi podataka.

- $p_1 = d_1 \oplus d_2 \oplus d_4 = 0 \oplus 1 \oplus 1 = 0$
- $p_2 = d_1 \oplus d_3 \oplus d_4 = 0 \oplus 0 \oplus 1 = 1$
- $p_3 = d_2 \oplus d_3 \oplus d_4 = 1 \oplus 0 \oplus 1 = 0$

*Dodatao objašnjenje:*  $d_1$  ima poziciju 3,  $d_2$  poziciju 5,  $d_4$  poziciju 7, a iz prethodnog znamo da se bitovi na pozicijama 1, 3, 5, 7 (grupisanje po prvom bitu) koriste za kontrolu parnosti. Ovakvim biranjem vrednost  $p_1, p_2, p_3$  odmah znamo da li su podaci koju su pristigli ispravni ili ne ako njihova suma po modulu 2 nije 0 (primer: za dobijeni podatak  $b_1b_2b_3b_4b_5b_6b_7$  proveravamo da li je  $b_1 \oplus b_3 \oplus b_5 \oplus b_7 = 0$ ). Tačnije,  $p_1, p_2$  i  $p_3$  biramo tako da XOR grupa (iznad navedeno grupisanje) bude 0.

*Dekodiranje:*

- *Prvi slučaj:* Pristigao kod je 0100101 ( $b_1b_2b_3b_4b_5b_6b_7$ ). Proverava se parnost bitova p1, p2 i p3 (XOR):
  - $p_1 = b_1 \oplus b_3 \oplus b_5 \oplus b_7 = 0 \oplus 0 \oplus 1 \oplus 1 = 0$
  - $p_2 = b_2 \oplus b_3 \oplus b_6 \oplus b_7 = 1 \oplus 0 \oplus 0 \oplus 1 = 0$
  - $p_3 = b_4 \oplus b_5 \oplus b_6 \oplus b_7 = 0 \oplus 1 \oplus 0 \oplus 1 = 0$

Rezultat je 000  $\Rightarrow$  Nema greške! Podatak je 0101 ( $b_3b_5b_6b_7$ , čitaju se svi bitove čiji indeks nije stepen dvojke).

- *Drugi slučaj:* Pristigao kod je 0100111 (greška na pretposlednjem bitu tj. šestom bitu). Proverava se parnost bitova p1, p2 i p3:

$$\begin{aligned} - p_1 &= b_1 \oplus b_3 \oplus b_5 \oplus b_7 = 0 \oplus 0 \oplus 1 \oplus 1 = 0 \\ - p_2 &= b_2 \oplus b_3 \oplus b_6 \oplus b_7 = 1 \oplus 0 \oplus 1 \oplus 1 = 1 \end{aligned}$$

$$- p_3 = b_4 \oplus b_5 \oplus b_6 \oplus b_7 = 0 \oplus 1 \oplus 1 \oplus 1 = 1$$

Rezultat (sindrom) je 110 ( $p_3p_2p_1$ , unazad)  $\Rightarrow$  Greška, bit na poziciji 110 (6) je komplementiran. Ispravlja se greška i konačan podatak se čita iz 0100101. Konačan rezultat je 0101. **Napomena:** Da su dva bita bila komplementirana ne bi se mogla ispraviti greška, ali bi se mogla uočiti (detektovati).

## 16.2 Kodovi za korekciju u praksi

U praksi se Hamingovi kodovi retko koriste. U upotrebi su najčešće:

- Konvolucioni algoritmi
- Metoda parnosti za malu gustinu
- Rid-Solomonovi kodovi

## 16.3 Detekcija ili korekcija

Korekcija grešaka se koristi kada su greške očekivane ili kada nema vremena za retransmisiju. Najčešće se koristi na fizičkom nivou. Detekcija grešaka je efikasna kada greške nisu očekivane tj. kada su retke. Koriste se na sloju veze u kombinaciji sa retransmisijom okvira.

Primer: Neka je verovatnoća greške 0.001 (jednobitna greška na svakih 1000 bitova ili 100-bitna greška na svakih 100000 bitova). Neka je veličina okvira 100. Da li je bolja detekcija sa retransmisijom ili korekcija? Ako koristimo Hamingov kod sa  $n = 4$ ,  $k = 3$  (neki drugi je verovatno značajno efikasniji) onda na svakih 4 bita koristimo 7 bitova tj. na 100 bitova koristimo 175 bitova (75% dodatnih bitova). Za jednobitne greške je ovo solidno rešenje (u slučaju dvobitne greške izvršiti retransmisiju). Ako koristimo kontrolne sume sa dodatnih 16 bitova onda u proseku se svaki deseti okvir šalje duplo (na 10 potrebnih okvira se šalje 11 okvira). Deset okvira po 100 bitova (obični podaci) čini 1000 bitova. Jedanaest okvira po 116 bitova čini 1276 bitova (to je 27.6% dodatnih bitova). Dodatno, ovom tehnikom se dedukuju i rafalne grešle. U ovom slučaju se verovatno isplati vršiti detekciju sa retransmisijom.

# 17 Sloj veze: tipovi servisa, okruženje, utočijski jednosmerni protokol.

Servisi koje sloj veze nudi mrežnom sloju:

- Servis bez uspostavljenje veze i bez potvrde prijema (Unacknowledged connectionless service): Slanje nezavisnih okvira bez provere da li je stigao ispravan okvir do primaoca. Primer: Ethernet.
- Servis bez uspostavljenje veze sa potvrdom prijema (Acknowledged connectionless service): Slanje nezavisnih okvira sa proverom da li je stigao ispravan okvir do primaoca. Primer: Wifi.
- Servis sa uspostavljenom vezom i potvrdom prijema (Acknowledged connection-oriented service): Uspostavlja se veza pre slanja niza okvira, gde se vodi računa za svaki okvir da li je stigao do primaoca. Retko se koristi.

## 17.1 Okruženje: Pregled struktura i funkcija

Sloj veze je obično realizovan delom na mrežnoj kartici, a drugim delom na nivou operativnog sistema.

Pregled nekih osnovnih konstanti i struktura koji su potrebni za pseudokodove:

```
#define MAX_PKT 1024           // determines packet size in
                                bytes
2 typedef enum {false, true} boolean; // boolean type
3 typedef unsigned int seq_nr;      // sequence or ack numbers
4 typedef struct {
5     unsigned
6     char data[MAX_PKT];
7 } packet;                      // packet definition
8 typedef enum {
9     data,
10    ack,
11    nak
12 } frame_kind;                // frame kind definition
13 typedef struct {
14     this layer
15     frame kind kind;          // what kind of frame is it?
16     seq_nr seq;               // sequence number
17     seq_nr ack;               // acknowledgement number
18     packet info;              // the network layer packet
} frame;
```

Pregled nekih funkcija za interakciju sloja veze sa slojem ispod i iznad:

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)	Прихвата 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

## 17.2 Utopijski jednosmerni protokol

Utopijski jednosmerni protokol radi na sistemu u kojem se smatra da je vreme procesiranja nula, pošiljalac i primalac su uvek spremni da šalju i primaju podatke, okviri su otporni na greške, i okviri se ne gube tokom komunikacije. Zbog svoje savršenosti i nerealnosti se ovaj sistem naziva „Utopia“. Protokol se sastoји из две procedure. Jedna procedura je za slanje, a druga za primanje podataka. Pseudokod za jednosmerni kanal:

```
void sender()
{
    frame s;
    packet buffer;

    while(true)
    {
        from_network_layer(&buffer); // mrežni sloj prosledjuje paket
        s.info = buffer;           // paket se stavlja u okvir
        to_physical_layer(&s)     // okvir se prosledjuje za slanje
    }
}

void receiver()
{
    frame s;
    event_type event; // postoji samo jedan tip dogadjaja u ovom slučaju

    while(true)
    {
        wait_for_(&event);      // ceka se dogadjaj
        from_physical_layer(&s); // uzima se okvir sa fizickog sloja
        to_network_layer(&s.info); // prosledjuje se paket na mrežni sloj
    }
}
```

## 18 Kontrola toka, ARQ, pauze (tajmauti), duplikati, protokol „stani i čekaj“ za savršen i nesavršen kanal.

### 18.1 Protokol „Stani i čekaj“ za savršen kanal

Sada se prepostavlja da pošiljalac i primalac ne moraju da imaju iste brzine slanja i primanja. U ovom slučaju je potrebna nekakva kontrola toka. Greške i dalje ne postoje. Jedno rešenje je da se napravi da primalac bude dovoljno brz da prihvati sve poruke bez problema. Ova metoda ne rešava problem, već ga samo odlaže.

**Protokol „stani i čekaj“** predstavlja jedno rešenje ovog problema. Pošiljalac šalje jedan okvir i čeka da stigne ACK (acknowledgement) okvir od primaoca koji potvrđuje da je taj okvir stigao. Okvir ACK može da bude prazan. Kada stigne ACK okvir, pošiljalac može da pošalje sledeći okvir. **Napomena:** Ovde je dvosmerna komunikacija na kanalu, ali na se podaci šalju u jednom smeru. Pseudokod:

```
void sender()
{
    frame s;
    packet buffer;
    event_type event;

    while(true)
    {
        from_network_layer(&buffer);
        s.info = buffer;
        to_physical_layer(&s);
        wait_for_event(&event); // jedina razlika je sto se ovde
        ceka ACK
    }
}

void receiver()
{
    frame r, s; // r - receive, s - send
    event_type event;

    while(true)
    {
        wait_for_event(&event);
        from_physical_layer(&r);
        to_network_layer(&r.info);
        to_physical_layer(&s); // jedina razlika sto se salje ACK
    }
}
```

### 18.2 Protokol „Stani i čekaj“ za nesavršen kanal

Sada se dodatno prepostavlja da se mogu desiti greške i okviri mogu biti izgubljeni. Problemi:

- Okvir može da se zagubi. Kako pošiljalac da zna da treba opet da ga pošalje?
- Ako primalac nekako javi pošiljaocu da se okvir zagubio, pošiljalac mu pošalje nov okvir, ali posle nekog vremena stigne i taj koji se zagubio. Kako primalac da zna da treba da odbaci ovaj duplikat?
- Šta ako se ACK okvir izgubi?

Potreban je napredniji protokol. Rešenje ovog problema nudi **ARQ** (**A**utomatic **R**epeat **re****Q**uest) protokol. Drugi naziv je **PAR** (**P**ositive **A**cknowledgement with **R**etransmission) protokol. Nakon što se pošalje okvir uključuje se tajmer. Ako tajmer istekne, onda se okvir opet šalje. Potrebno je izabrati dobro vreme za brojac da ne bude dovoljno kratko ili dovoljno dugačko kako bi se povećale performanse sistema. Ako tajmer istekne, to znači da se zagubio okvir koji je poslat ili se zagubio ACK okvir. Takođe se koristi serijski brojevi za okvire kako bi se oni razlikovali (kako bi znali da li stigao duplikat). Pošiljalac, dok traje tajmer, može da očekuje tri slučaja:

- Stigao je ispravan ACK okvir;
- Stigao je neispravan ACK okvir ili ACK duplikat;
- tajmer je istekao.

Ako je stigao ispravan ACK okvir, onda se uzima sledeći paket sa mrežnog sloja, pomera se serijski broj za sledeći ACK okvir koji se očekuje i šalje se sledeći okvir (paket). U suprotnom se šalje opet isti okvir. Slična stvar je i kod primaoca:

- Stigao je ispravan okvir;
- Stigao je neispravan okvir ili duplikat;

Ako je ispravan okvir, on se prosleđuje mrežnom sloju, šalje se ACK okvir primaocu i pomera se serijski broj za sledeći okvir koji se čeka. U suprotnom (neispravan okvir ili duplikat) se šalje opet ACK za poslednji ispravan okvir. Pseudokod:

```

2 void sender()
3 {
4     seq_nr next_frame_to_send;
5     frame s;
6     packet buffer;
7     event_type event;
8
9     next_frame_to_send = 0;
10    from_network_layer(&buffer);
11    while(true)
12    {
13        s.info = buffer;
14        s.seq = next_frame_to_send;
15        to_physical_layer(&s); // salje se okvir
16        start_timer(s.seq); // i aktivira se brojac za odg.
17        ser.broj
18        wait_for_event(&event);
19        if(event == frame_arrival) // stigao je novi okvir
20        {
21            from_physical_layer(&s);
22            if(s.ack == next_frame_to_send) // da li je ispravan
23            broj okvira?
24            {
25                stop(s.ack); // prekida se brojac za odg. ser.
26                broj
27                    from_network_layer(&buffer); // uzima se sledeci
28                paket za slanje
29                    inc(next_frame_to_send); // pomera se ser.
30                broj
31            }
32        }
33    }
34
35 void receiver()
36 {
37     seq_nr expected_frame;
38     frame r, s;
39     event_type event;
40
41     frame expected = 0;
42     while(true)
43     {
44         wait_for_event(&event);
45         if(event == frame_arrival) // stigao je okvir
46         {
47             from_physical_layer(&r);
48             if(r.seq == expected_frame) // da li je ispravan broj
49             okvira?
50             {
51                 to_network_layer(&r.info);
52                 inc(frame_expected);
53             }
54             s.ack = 1 - expected_frame; // priprema se potvrda
55             to_physical_layer(&s); // i salje se primaocu
56         }
57     }
58 }
```

## 19 Protokol kliznih prozora u sloju veze, „1-bitni“, „vrati se N“, „selektivno ponavljanje“.

Prethodni protokoli su prenosili okvire samo u jednom smeru, gde pošiljalac šalje okvir primaocu i onda primalac šalje prazan ACK okvir nazad.

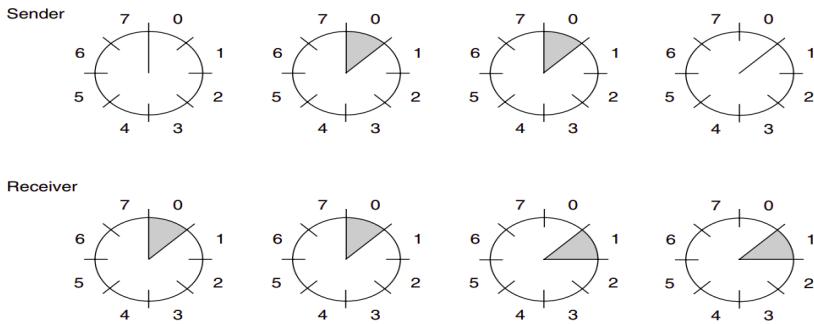
U slučaju dvosmerne komunikacije bi bio potreban dupli simpleks kanal za dvosmernu komunikaciju. Bolja ideja je da se koristi metoda „šlepanje“ („piggybacking“). Ova metoda podrazumeva da umesto da se odmah šalje prazan ACK okvir za potvrdu, sačeka (neko vreme) da se spremi okvir koji bi svakako trebalo da bude poslat i zakači ACK broj na njega. Koristeći ovu metodu ostvaruje se mogućnost dvosmerne komunikacije koristeći jedan kanal. Međutim, ovo stvara nove probleme.

Neka postoje dva čvora u komunikaciji na jednom kanalu tj. prvi čvor i drugi čvor. Oba čvora mogu da šalju i primaju okvire. Primjenjuje se „piggybacking“. Situacija je sledeća: Prvi čvor je poslao okvir drugom čvoru, i drugom čvoru je stigao ispravan okvir. Pošto se koristi „piggybacking“, drugi čvor treba da zakači ACK na okvir koji treba da se svakako pošalje prвом čvoru. Ako drugi čvor u nekom vremenskom intervalu ne treba da šalje okvire nazad (zajedno sa ACK), onda prвom čvoru može da istekne brojač (tajmer), pri čemu će duplikat biti poslat. U tom slučaju drugi čvor mora poslati prazan ACK okvir. Pitanje je koliko dugo treba da čeka drugi čvor na novi okvir (treba da stigne paket od mrežnog sloja za novi okvir) koji treba da pošalje pre nego što odluči da ipak pošalje prazan ACK okvir.

### 19.1 Protokol kliznih prozora u sloju veze (opšta priča)

Pošiljalac ima na raspolaganju nekoliko ( $W$ ) uzastopnih okvira koje može da pošalje. Koristi bafer veličine  $W$  da čuva okvire koje treba da pošalje. Slična priča je i za primaoca. Primalac ima bafer veličine  $W$  za okvire koje može da prihvati (baferi ne moraju nužno biti iste dimenzije).

**Zašto primalac koristi bafer?** I dalje se poštuje pravilo da paketi na mrežnom sloju stižu u istom redosledu kako su poslati, nezavisno od toga što okviri možda ne stižu „pravim“ redosledom. Zbog toga se u bafetu drže određeni okviri (i u njima paketi) dok ih ne prihvati mrežni sloj. Primer kliznih prozora sa  $W = 1$  (dvosmerni „Stani i čekaj“):



Obašnjenje slike:

- Prva kolona: Inicijalizacija;
- Druga kolona: Šalje se prvi okvir;
- Treća kolona: Stigao je ispravan prvi okvir;
- Četvrta kolona: Stigao je ispravan ACK okvir.

Veći prozori omogućavaju protočnu obradu za efikasniju potrebu kanala. Što je veći broj prozora to je efikasnija upotreba. Najneefikasniji slučaj je za  $W = 1$  (posebno na dužim kanalima).

Optimalno  $W$  zavisi od BDP-a. Cilj je da važi  $W \geq 2BD + 1$  radi što bolje iskorišćenosti, gde je  $BD = \frac{BDP}{FS}$ , gde je **FS** - Frame Size tj. dimenzija jednog okvira u bitovima. Vrednost BD predstavlja maksimalni broj okvira na kanalu u jednom trenutku, a „+1“ deo je tu zato što se ACK neće poslati dok ne stigne ceo okvir. Možemo definisati **efikasnost veze (Link Utilization)** kao:

$$\text{link utilization} \leq \frac{w}{1+2BD}$$

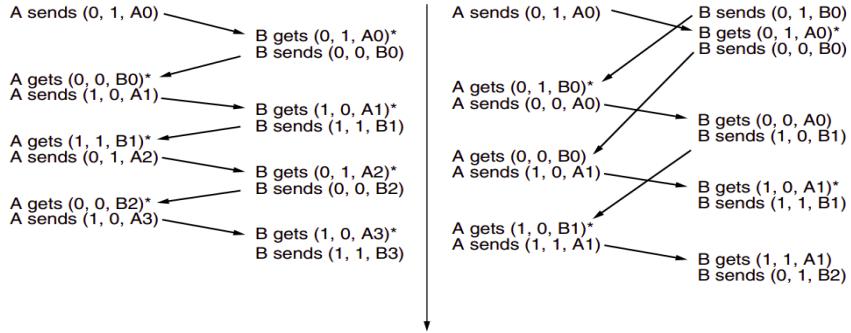
## 19.2 Protokol „1-bitni“

Nema odvojenih algoritama pošiljaoca i primoaca, jer je sad dvosmerni kanal i čvorovi se ponašaju simetrično. Svodi se na dvosmerni „Stani i čekaj“.

```

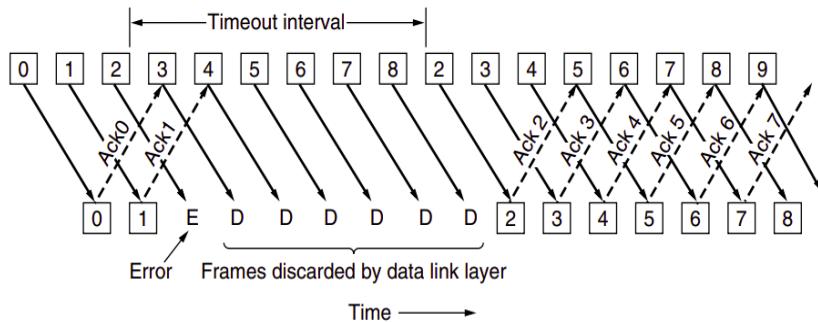
2   void protocol()
3   {
4       seq_nr next_frame_to_send = 0; // dovoljno je da se uzimaju
5       seq_nr expected_frame    = 0; // vrednosti 0 ili 1
6       frame r, s;
7       packet buffer;
8       event_type event;
9
10      from_network_layer(&buffer); // uzima se paket sa mreznog
11          sloja
12      s.info = buffer;           // okviruje se paket
13      s.seq = next_frame_to_send; // postavlja se serijski broj
14      s.ack = 1-frame_expected; // zakaci se ACK ser. broj.
15      to_physical_layer(&s)      // salje se okvir
16      start_timer(s.seq)        // i pokreće se brojac
17      while(true)
18      {
19          wait_for_event(&event);
20          if(event == frame_arrival)
21          {
22              from_physical_layer(&r);
23              if(r.seq == expected_frame) // stigao je validan okvir
24              {
25                  to_network_layer(&r); // prosledjuje se mrežnom
26                  nivou
27                  inc(expected_frame); // „,pomera se prozor“
28              }
29              if(r.ack == next_frame_to_send) // poslat okvir je
30              uspesno stigao
31              {
32                  stop_timer(s.seq);      // prekida se brojac
33                  from_network_layer(&buffer); // uzima se sledeći
34                  paket
35                  inc(next_frame_to_send); // azurira se ser. br
36                  . za slanje
37              }
38              s.info = buffer;           // uokviruje se sledeći
39              paket koji je
40              s.seq = next_frame_to_send; // novi ili stari ako nije
41              stigao ACK
42              s.ack = 1-expected_frame; // zakaci se odg. ACK
43              to_physical_layer(&s);     // salje se okvir
44              start_timer(&s.seq);       // i pokreće se brojac
45          }
46      }
47  }
```

Ovaj protokol radi solidno u opštem slučaju. U specijalnom slučaju oba čvora simultano započinju komunikaciju. Posledice su duplikati i okinuti brojači (tajmeri). Primer (dobar i loš slučaj):



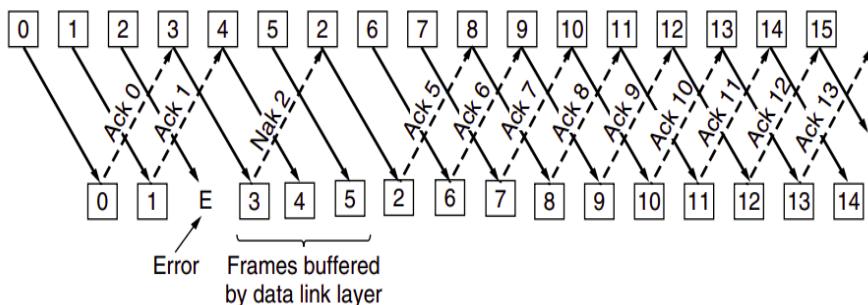
### 19.3 Protokol „Vrati se N“

Pošiljalac šalje niz uzastopnih okvira. Primalac prihvata samo okvire koji stižu redom. Ako neki okvir stigne pre reda onda se on odbacuje. Okviri sa ACK se šalju samo za neodbačene okvire. Kada pošiljaocu istekne tajmer, on opet šalje niz uzastopnih okvira. Primalac ima jednostavnu implementaciju sa baferom dimenzije 1. U najgorem slučaju moraju opet svi prozori da se šalju, što dovodi do neefikasnosti na kanalu:

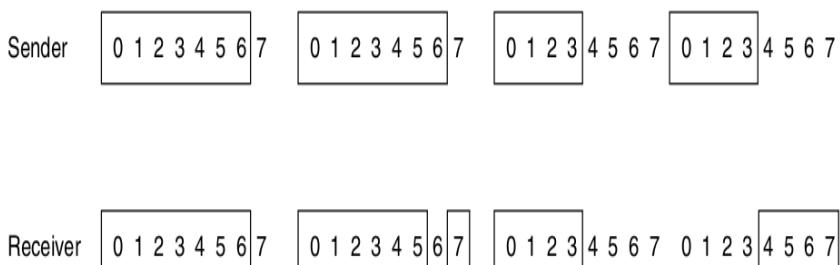


#### 19.4 Protokol „Selektivno ponavljanje“

Predstavlja generalizaciju prethodnog protokola. Primalac sada ima bafer dimenzije veće od 1 (ako je jednako 1, onda je to „Vrati se N“ protokol) i prihvata nekoliko selektivno okvira (sve okvire čiji je redni broj na baferu). Dodatno se koristi **NAK** (negativni ACK) koji informiše pošiljaoca da ponovo pošalje problematičan okvir. Tada pošiljalac šalje opet okvir koji odgovara NAK rednom broju, umesto da čeka da istekne tajmer. Ovo znatno povećava efikasnost kanala. Ovaj sistem ima složeniju implementaciju, ali ima veću efikasnost:



**Napomena:** Opseg brojeva sekvenci mora biti barem duplo veći od dimenzije bafera! Neka je dimenzija prozora  $W = 7$ , a opseg brojeva  $S = 8$ . Nakon što pošiljalac uspešno pošalje okvire sa identifikatorima od 0 do 6, primalac pomeri prozor od 7 do 7 i onda 0 do 5 (ciklično). Ako se neki ACK od 0 do 5 ne vrati nazad pošiljaocu, a pošiljalac pošalje duplikat, mrežni sloj će dobiti pogrešan paket i time ceo protokol propada.



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

Mrežni kanali se dele u dve grupe. Prva grupa su „tačka-na-tačku“ (eng. „point-to-point“) koji se odnose na komunikaciju između dva čvora. Druga grupa su **javna emitovanja** (eng. **broadcast**) koja predstavljaju malo komplikovanije sisteme. U ovim sistemima obično čvorovi smetaju jedni drugima tj. kvare signal ako šalju signal u isto vreme.

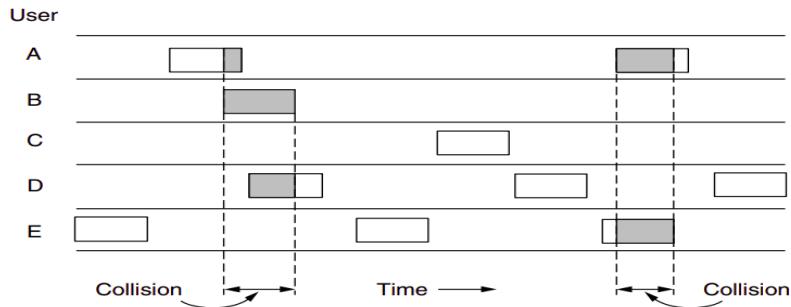
Primer: U grupnim pozivima se često dešava da krene više ljudi da govoru u isto vreme, nakon što neko završi govor. Ovakvi kanali se nazivaju i „multi access channels“ ili „random access channels“.

Na kanalima sa više čvorova protokoli su zasnovani na tome da određuju ko sledeći „dobija pravo glasa“. Ovi protokoli pripadaju **MAC** (Medium Access Control) podsloju. Primer: Lan mreže.

## 20.1 protokol ALOHA

ALOHA mreža je računarska mreža koja je povezivala havajska ostrva krajem 60-tih godina i početkom 70-tih. Povod za izgradnju ove mreže su nedostaci telefonske mreže na havajskim ostrvima. Povezivanje ostrva kablovima nije bilo realno rešenje.

**Pravi ALOHA:** Veoma jednostavan sistem: Kada čvor treba da emituje signal? - Kada ima nešto što treba da pošalje. Naravno, ovde su kolizije i oštećenja signala neizbežni. Čvor koji treba da pošalje okvir emitiše signal. Svi čvorovi koji ne šalju signal slušaju. Kada čvorovi prime signal oni ga dalje emituju. Na taj način prvobitni pošiljalac prihvata signal od nekog drugog čvora (istи okvir које је послao) i time zna da je okvir uspešno poslat. Ako je okvir oštećen onda čvor čeka neko vreme (nasumično određeno) i šalje opet signal. Problem: Svaki put kada dva čvora žele da pošalju okvir, dešava se kolizija, čime su okviri uništeni. Primer sa pet čvorova (A, B, C, D, E):



Ovaj sistem radi dobro ako je opterećenje mreže malo. Efikasnost ove mreže je 18% tj. 18 od 100 okvira ne bude uništeno.

**Diskretizovan ALOHA:** Jednostavno unapređenje sistema. Potrebno je da se čekanje za ponovno slanje diskretizuje u slotove jednakih dužina. Svaki čvor sada biraa nasumičan slot kada će opet da pošalje okvir. Pažljivim biranjem dužine slot-a efikasnost mreže može da se poveća na 36%.

## 21 CSMA, CSMA/CD, BEB.

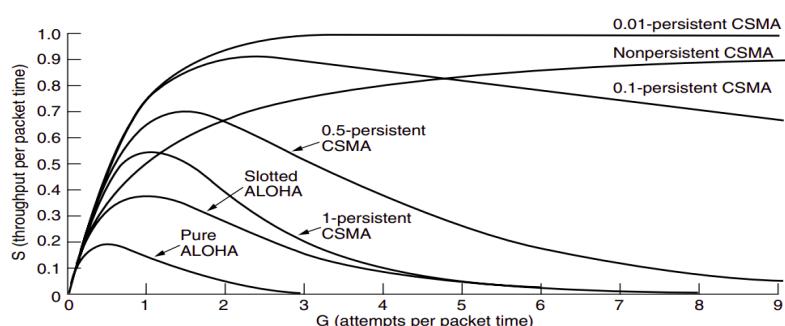
Postoji posebna grupa protokola koja osluškuje kanal pre nego što šalje signal. Osluškivanje kanala je jednostavno kod žičanih, a komplikovano kod bežičnih kanala. Moguće je da se kanal osluškuje, a da se ne čuje da neko drugi šalje signal (posledica kašnjenja).

## 21.1 protokol CSMA

Protokol CSMA (Carrier Sense Multiple Access) predstavlja poboljšanje ALOHA protokola koji osluškuje kanal pre slanja. Postoje više verzija CSMA protokola:

- **1-persistent CSMA:** Ako čvor treba da pošalje signal, prvo osluškuje kanal i onda reaguje u odnosu na situaciju:
  - Ako je kanal slobodan, šalje se signal.
  - Ako kanal nije slobodan, onda se čeka da bude slobodan.
  - Ako se desi kolizija, onda čvor čeka nasumično određeno vreme.
- Protokol se zove „1-persistent“ zato što šalje signal kada je kanal slobodan sa verovatnoćom 1. Problemi:
  - Kolizija se na ovom protokolu dešava češće nego što je to inicijalno intuitivno. Primer: Postoje tri čvora u mreži sa oznakama A, B, C. Neka čvor C trenutno šalje signal ostalim čvorovima u mreži tj. čvoru A i čvoru B. Takođe, neka u tom trenutku čvorovi A i B imaju podatke koje žele da pošalju i čekaju da C završi slanje. Kada C završi slanje, A i B će u isto vreme da krenu da šalju signal što dovodi do kolizije. Ova situacija je relativno česta u gustim mrežama.
  - Kvalitet protokola zavisi dosta od kašnjenja signala. Ako signal kasni, onda može da se desi da čvor ne čuje da neko drugi šalje signal (signal kasni) i kreće da šalje signal. Tada opet dolazi do kolizije.
- **nonpersistent CSMA:** Ako čvor treba da pošalje signal, prvo osluškuje kanal. Ako je kanal zauzet, onda umesto da uporno čeka da što pre zauzme kanal, čeka nasumično određeno vreme pre nego što pokuša opet. Rezultat je manje kolizija, ali duže čekanje.
- **p-persistent CSMA:** Ako čvor treba da pošalje signal, prvo osluškuje kanal. Ako je kanal slobodan onda se sa verovatnoćom  $p$  preuzima kanal, odnosno sa verovatnoćom  $1-p$  se čeka nasumično određeno vreme pre nego što čvor pokuša opet da preuzme kanal. Ovaj protokol predstavlja hibrid prethodna dva protokola.

**Upoređivanje protokola:**



## 21.2 protokol CSMA/CD

Protokol CSMA/CD (Carrier Sense Multiple Access With Collision Detection) predstavlja poboljšanje osnove verzije CSMA. Ra-

zlika u odnosu na klasičan CSMA je što je prenos prekida čim se primeti da je nastala kolizija. Nakon Primećenje kolizije čvorovi koji su prethodno slali podatke čekaju nasumično određeno vreme pre nego što opet pokušaju da šalju signal.

### 21.3 Binarno eksponencijalno odlaganje (BEB)

**Binary Exponential Backoff** predstavlja još jedno poboljšanje na osnovni CSMA model (CSMA/CD). Svaki čvor kada se primeti koliziju primenjuje sledeći algoritam:

- Prva kolizija: Čvor čeka 0 ili 1 vremenska okvira (nasumično odabran).
- Druga kolizija: Čvor čeka između 0 i 3 vremenska okvira (nasumično odabran).
- Treća kolizija: Čvor čeka između 0 i 7 vremenska okvira (nasumično odabran).
- ...
- N-ta kolizija: Čvor čeka između 0 i  $2^{N-1}$  vremenska okvira (nasumično odabran).

Opseg intervala brzo raste zbog čega dolazi do dobre procene njegove idealne dužine. Veoma efikasan u praksi.

## 22 MAC protokoli zasnovani na redosledu: „Token Ring“

Problem kod CSMA protokola je što je loš pod velikim opterećenjem zbog visokih dodatnih troškova (puno kolizija) i variranja vremenskog pristupa. Postoje MAC protokoli koji ne koriste slučajnost već su bazirani na redosledu. Definiše se uređenje prema kojem čvorovi šalju ako imaju nešto da pošalju (ili samo propuste prioritet ako nemaju šta da pošalju).

Prednosti protokola sa redosledom je što rade efikasnije pod velikim opterećenjem. Garantovano je da će svi čvorovi biti usluženi u unapred definisanom vremenu.

Mane protokola sa redosledom je složenost (može da se izgubi token ili slično). Takođe je relativno visok dodatni trošak pri malom opterećenju.

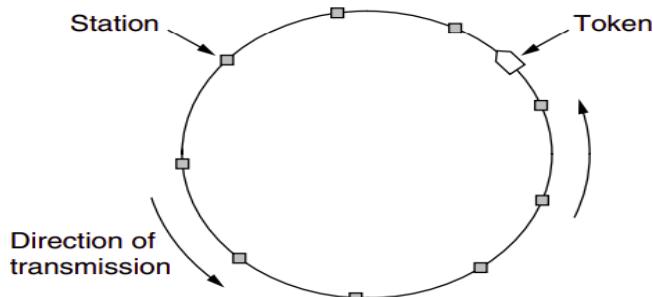
U praksi se protokoli sa redosledom obično isprobavaju kao poboljšanje. Međutim, protokoli sa slučajnošću se obično teško nadmašuju (jednostavnji i dovoljno dobri, skalabilni).

### 22.1 Protokol „Token Ring“

Koristi se topologija mreže za definisanje prioriteta po kojem čvorovi mogu da šalju podatke. Svi čvorovi su spojeni u jedan prsten. Token kruži u okviru prstena prolazeći kroz čvorove. Proces:

- Čvor prima token od svog prethodnika čime dobija prioritet da šalje okvir.
- Ako čvor ima okvir koji čeka da bude poslat, onda čvor šalje taj okvir pre nego što prosledi token (samim tim i prioritet) sledećem čvoru.

- Ako čvor nema okvir koji čeka da bude poslat, onda čvor samo pređe token sledećem čvoru.

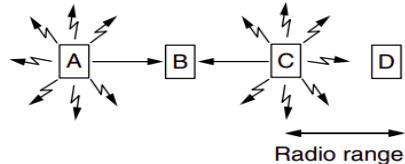


Ovde okviri putuju u istom smeru kao i token. Kako okviri ne bi putovali beskonačno u okviru mreže, oni moraju da budu sklonjeni sa mreže. Jedan način je da skine token sa mreže kada čvor (primalac) primi okvir. Drugi način je da sam čvor, pošiljalac skine sa mreže okvir nakon što taj okvir obrne krug.

Nije nužno potrebno da kanal bude fizički prsten. Umesto toga je moguće koristiti dugu magistralu gde svaki čvor šalje token sledećem predefinisano sledbeniku. Ovaj protokol se naziva „**Token Bus**“.

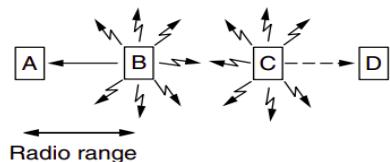
## 23 MAC protokoli za bežične mreže

Problem kod bežičnih mreža je teško detektovanje kolizije kada se ona dešava (Signal koji stiže do čvora može biti veoma slab). Zbog toga nije moguće koristiti CD dodatak uz CSMA. Nažalost, takođe nije moguće koristiti CSMA protokol. Primer: Mreža se sastoji od četiri čvora A, B, C i D:



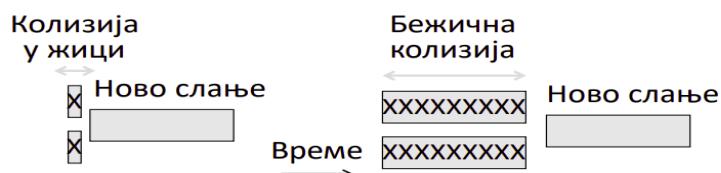
U ovom slučaju čvorovi A i C pokušavaju da šalju signal čvoru B koristeći CSMA protokol. Ako A počne da šalje signal, a C u tom trenutku sluša kanal, onda C neće čuti signal koji dolazi od A. Tada C može da pokuša da šalje signal čvoru B. Pošto A i C šalju signal u isto vreme dolazi do kolizija signala pri čemu oba okvira (od A i C) bivaju oštećena. Ovaj problem se naziva „problem sakrivenog čvora“ („hidden terminal problem“).

Drugi Primer: Mreža se sastoji od istih čvorova:



U ovom slučaju čvor B želi da šalje signal čvoru A, a čvor C želi da šalje signal čvoru D. Ako B šalje signal, a C u tom trenutku sluša da li neko šalje signal, onda će C čuti signal koji stiže od B i pogrešno će zaključiti da se signal ne može slati čvoru D (ne može oštetići signal koji stiže od B ka A, jer je A previše daleko). Ovaj problem se naziva „problem izloženog čvora“ (exposed terminal problem).

Kod žica detekcija kolizija (i rano obustavljanje) smanjuje dodatne troškove (overhead). Kod bežičnog kanala su veći dodatni troškovi, jer rano obustavljanje nije moguće.



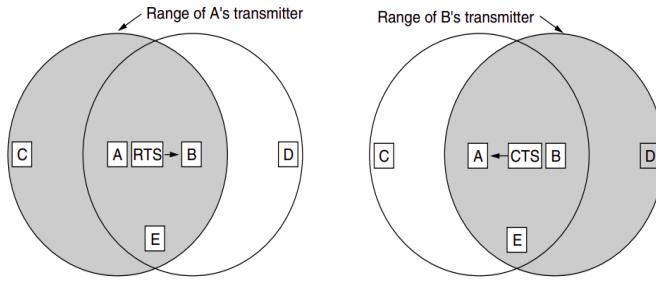
### 23.1 Protokol MACA

Protokol MACA (Multiple Access with Collision Avoidance) koristi proceduru „rukovanja“ umesto metode CSMA. U praksi se koristi poboljšana verzija MACA (protokol 802.11 tj. WiFi). Procedura MACA:

1. Pošiljalac emituje **RTS okvir (Request To Send)** sa informacijom o dužini okvira koji želi da pošalje.
2. Primalac dobija RTS i emituje **CTS okvir (Clear To Send)** sa informacijom o dužini okvira koji treba da primi (kopiran do RTS).
3. Pošiljalac prima CTS i onda počinje slanje.

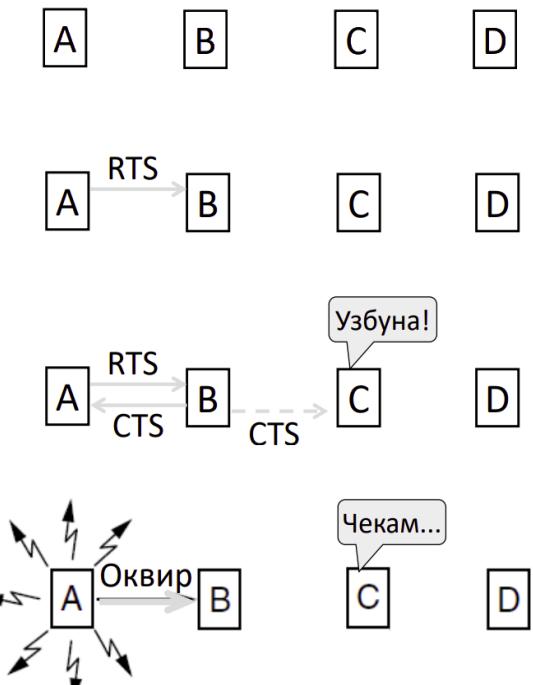
Kako drugi čvorovi reaguju na RTS i CTS?

- Ako neki čvor čuje RTS, onda on mora da sačeka neko vreme da bi onaj čvor koji šalje RTS uspešno prihvatio CTS bez konflikata.
- Ako neki čvor čuje CTS, onda on mora da sačeka neko vreme. Pošto se dužina podatka nalazi u okviru CTS-a, može da se odrediti koliko dugo treba da se čeka.

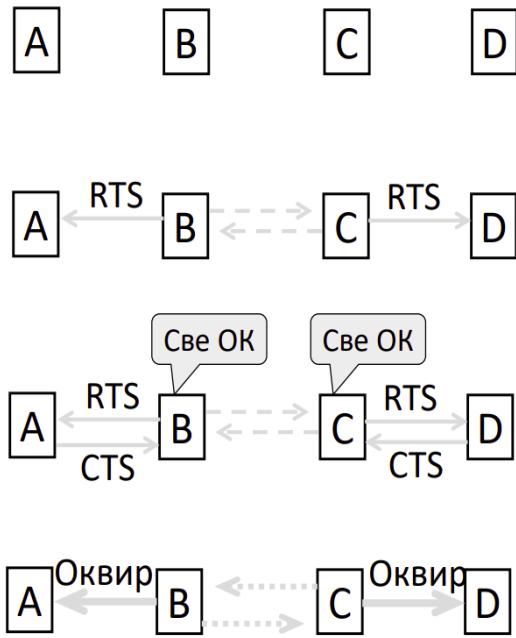


Kolizije i dalje mogu da nastanu. Primer: Čvorovi B i C u isto vreme šalju RTS okvire. U tom slučaju dolazi do kolizije i jedan čvor će morati da sačeka nasumično određeno vreme.

MACA i problem sakrivenog čvora:



MACA i problem otkrivenog čvora:



## 24 Klasični Eternet - IEEE 802.3

Najpopularniji vid organizovanja LAN mreža tokom 80-tih i 90-tih godina. Protok oko 10Mb/s preko deljenog koaksijalnog kabla. Koristi „CSMA/CD sa BEB“ model.

Razlika u odnosu na dosadašnje okvire je što se oni sastoje od adrese pošiljaoca i adrese primaoca (ovo nije potrebno za „point-to-point“ protokole). Za detekciju grešaka se koristi CRC-32. Ne koristi ACK sistem, jer se to ostavlja višim slojevima.



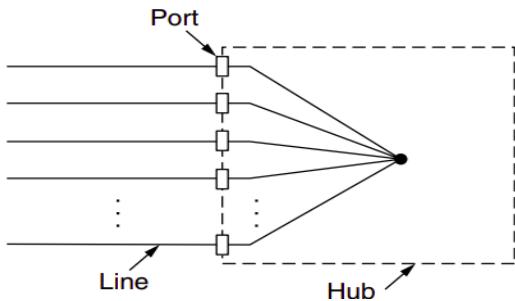
## 25 Moderni (komutirani) Eternet

Moderni Eternet ne koristi MAC, već skretnice (switches). Kanali se na hardverskom nivou razdvajaju.

## 25.1 Razvodnik (HUB)

Najlakši način da se povežu čvorovi u mreži je da se koristi jedan dovoljno dugačak kanal preko kojeg čvorovi mogu da šalju okvire. Ovo rešenje donosi mnogo problema prilikom prekida na kanalu (nije lako locirati).

Bolje rešenje se dobija koristeći razvodnik (hub). Razvodnik je kutija sa više ulaza/izlaza preko koje mogu da se povezuju čvorovi. Funkcija i implementacija razvodnika je jednostavna. Kada stigne neki okvir do njega on ga prosleđuje svim ostalim čvorovima koji su prikačeni na njega.



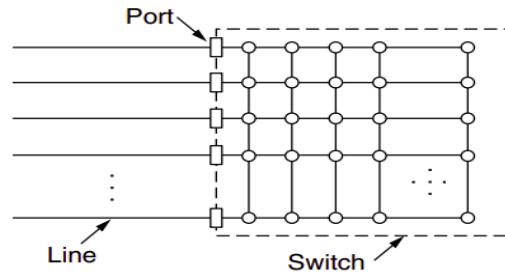
### Prednosti:

- Značajno ušteđuje na količini materijala potrebno da se spoje čvorovi (potrebna manja ukupna dužina kanala mreže).
- Dodavanje i uklanjanje čvorova je poprilično jednostavno.
- Pronalaženje prekida je lakše. Ako se izgubi kontakt sa jednim čvorom, onda je prekid na žici između čvora i razvodnika ili je čvor otkazao. Ako se izgubi kontakt na nivou cele mreže, onda je razvodnik otkazao.

Nažalost, razvodnik ne povećava kapacitet i skalabilnost mreže, jer predstavlja logički ekvivalent dugog kabla. Razvodnik i dalje zahteva protokol CSMA/CD, jer kolizije pri slanju ostaju.

## 25.2 Skretnica (Switch) - opšte

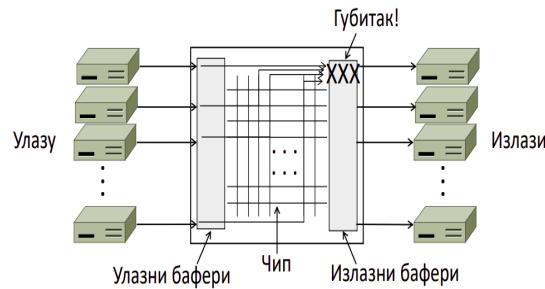
Skretnica izgleda isto kao razvodnik i čvorovi se povezuju na isti način. Prednost skretnice je u njenoj unutrašnjoj implementaciji. Umesto da se okvir šalje svim čvorovima u mreži, skretnica pamti adrese svojih čvorova i prosleđuje okvir samo primaocu. Skretnica ima odgovaraajući algoritam za pamćenje adresa. Svi čvorovi sa skretnicom su obično povezani preko punog dupleksa. To znači da svaki čvor može da šalje okvire bez brige da li će doći do kolizije. Posledica ovo je da CSMA/CD gubi svoj smisao u nedostatku kolizija.



Prednosti u odnosu na razvodnik po performansama (sve prednosti razvodnika u odnosu na jedan dugi kanal važe i za skretnicu):

- Nema kolizije i time je kapacitet bolje iskorišćen.
- Može da šalje više okvira u isto vreme.

Skretnica ima bafere za ulaz i izlaz u slučaju da ima više okvira od jednog čvora koji stižu ili u slučaju da više okvira treba da se pošalju jednom čvoru (u slučaju prelivanja bafera gube se okviri).

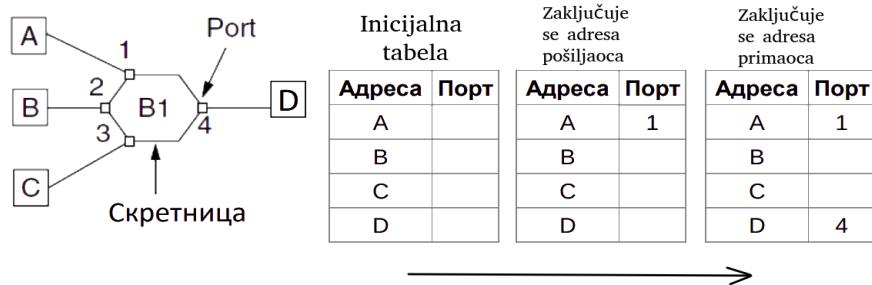


56

### 25.3 Skretnica (Switch) - učenje unazad

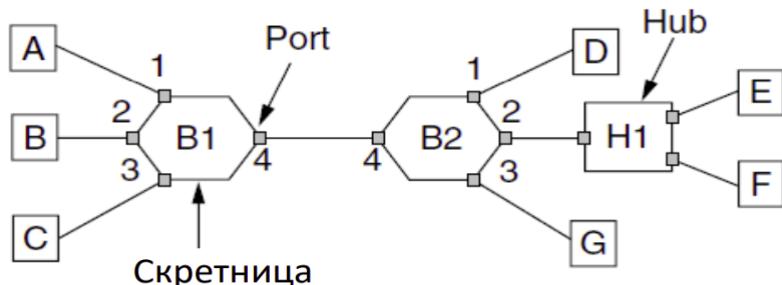
Za prosleđivanje okvira se koristi tabela relacija između broja porta i adrese okvira. Da bi se popunila tabela, posmatraju se adrese i portovi čvorova koji štampaju okvire. Ako se za zadatu adresu u tabeli nalazi pridruženi port, onda se okvir šalje samo njemu, a u suprotnom se šalje svim čvorovima.

Data je mreža sa četiri čvora. Demonstracija učenja tabele relacije. Neka čvor A šalje okvir čvoru D:



### Detaljno objašnjenje procesa sa slike:

- Inicijalno skretnica ima praznu tabelu.
- Čvor A šalje okvir čvoru D (okvir sadrži adresu primaoca i pošiljaoca).
- Okvir stiže iz čvora A. Adresi čvora A se dodeljuje port koji služi za kasnije preslikavanje.
- Problem: Skretnica ne zna koji port vodi ka čvoru D! U ovakvim situacijama se skretnica ponaša kao razvodnik i šalje okvir svim ostalim čvorovima.
- Čvorovi koji dobiju okvir od skretnice, a ne odgovara im adresa pošiljaoca, jednostavno odbacuju okvir. Čvoru D odgovara adresa pošiljaoca i on šalje odgovor čvoru A.
- Skretnici stiže okvir od čvora D. Adresi čvora D se dodeljuje port koji služi za kasnije preslikavanje.
- Skretnica prosleđuje okvir čvoru A.



Na jednoj mreži mogu da se nalaze više skretnica i razvodnika. Svaka skretnica ima svoju nezavisnu tabelu.

### 25.4 Eliminacija petlji

U slučaju komplikovanih mreža može doći do petlji. Rešenje ovog problema je formiranje razapinjuće stabla. Razapinjuće stablo nema petlje, ali povezanost u okviru mreže ostaje (ako je postojao neki put između dva čvora, onda i dalje postoji neki put između ta dva čvora, ali ne nužno isti put).



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

Jedinica podataka koja se koristi na nivou mrežnog sloja je paket. Uloga mrežnog sloja je da šalje paket od izvora do odredišta. Mrežni sloj je najniži sloj koji prosleđuje tako podatke (sloj veze prosleđuje okvire sa jednog kraja žice na drugi). Da bi se ova uloga uspešno obavljala, mrežni sloj nalazi u topologiju mreže kako bi izabrao dobre puteve, balansirao zagušenja na nivou mreže i omogućio komunikaciju mreža sa različitim tehnologijama.

**Problemi sa skretnicama:** Teoretski, koristeći veze i skretnice je dovoljno da se izgradi mreža koja može da razmenjuje podatke. Problemi:

- Loša skalabilnost: Tabela relacija kod skretnica bi postala ogromna. Takođe skretnice bi u početku se ponašale kao razvodnici dok ne nauče tabele relacija.
- Ne mogu da se kombinuju različiti tipovi mreža (različite tehnologije). Primer, Ethernets i 3G.
- Ne postoji kontrola saobraćaja koja bi smanjila zagušenje u okviru mreže.

**Rutiranje i prosleđivanje (ukratko):** Rutiranje je proces u kojem se odlučuje u kom pravcu treba poslati podatke. Prosleđivanje je proces slanja paketa na osnovu lokalne tabele.

### **26.1 Tipovi servisa na mrežnom sloju**

Uslovi koje servisi koje mrežni sloj pruža transportnom sloju treba da ispunjavaju:

1. Servisi treba da budu nezavisni od tehnologije rutera.
2. Transportni sloj ne mora da zna broj, tip ili topologiju rutera.
3. Adrese koje koristi transportni sloj bi trebalo da koriste neki ujednačeni plan za brojanje.

Postoje dva tipa servisa koje mrežni sloj nudi transportnom sloju:

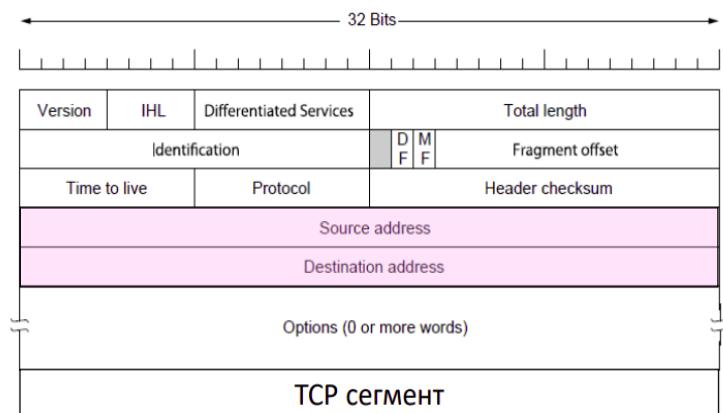
- **Datagrami (Connectionless Service):** Servis bez uspostavljenja veze (kao pošta). Svaki paket se šalje nezavisno. Paket sadrži ciljnu adresu na osnovu koje usmerivač (ruter) prosleđuje paket dalje koristeći dinamičnu (menja se vremenom) **tabelu prosleđivanja (forwarding table)**. Ukoliko jedan čvor šalje više paketa nekom drugom čvoru, onda ne mora svaki paket da putuje istim putem. **Primer:** Treba da se pošalje neki veliki podatak koji se dobija od transportnog sloja. Taj podatak može da stane u par paketa, pa se tako i šalje. Svaki od ovih paketa može da ima različitu putanju.
- **Virtuelna kola (Connection-Oriented Service):** Servis sa uspostavljenom vezom (kao fiksna telefonija). Paketi, koji se šalju koristeći

ovaj servis, putuju istom putanjom koja je određena kada je uspostavljena virtuelna konekcija. Prilikom uspostavljenja virtuelne veze svi usmerivači (ruteri) u okviru putanje trebaju da imaju informaciju o toj putanji. Paketi sadrže kratak identifikator virtuelnog kola (globalno značenje). U slučaju da dva različita čvora žele da uspostave konekciju sa jednim istim čvorom, potrebno je da svako kolo ima svoj identifikator kako bi se razlikovalo od ostalih kola.

Oba servisa koriste „sačuvaj-i-prosledi“ („store-and-forward“) tehniku, gde usmerivači (ruteri) dobijaju paket i privremeno ga čuvaju (ruteri imaju svoje bafere) sve dok ga ne proslede dalje.

## 26.2 Internet protokol (IP)

Mrežni sloj koristi datagramske servise. Paket ipv4 ima 32-bitnu adresu i obično je velik oko 1.5KB.



### Elementi:

- **Version:** verzija ipv4 (češća) ili ipv6.
- **IHL:** dužina zaglavlja.
- **Differentiated Services:** informacije o klasi servisa i zagušenju.
- **Total Length:** ukupna dužina paketa (sa zaglavljem).
- **Identification field:** informacija o tome kome paket pripada kada stigne na odredište.
- **DF, MF, Fragment offset:** dodatne informacije o fragmentu.
- **Time to Live:** definiše se životni vek paketa (u slučaju da se izgubi, da ne bi lutao).
- **Protocol:** UDP ili TCP (ili neki treći).
- **Header Checksum:** za detekciju greške.
- **Source Address:** IP adresa pošiljaoca.
- **Destination Address:** IP adresa primaoca.
- **Options:** Dodatne informacije koje zavise od verzije IP protokola.
- **TCP segment.**

### 26.3 Datagrami i virtuelna kola

Virtuelno kolo zahteva pripremu da se uspostavi veza. Ova priprema zahteva dodatne resurse. Međutim, kada se cena plati, za usmeravanja u okviru kola je dovoljno da se koristi oznaka za usmeravanja, umesto cele adrese koja se koristi u slučaju datagrama. Virtuelna kola su više osjetljiva na otkazivanje ruteru u okviru definisane putanje. U tom slučaju veza propada (datagrami nemaju ovaj problem). Takođe, datagrami omogućavaju veću fleksibilnost kod razrešavanja zagušenja u okvir mreže. Zaključak: oba servisa imaju svoje prednosti i mane.

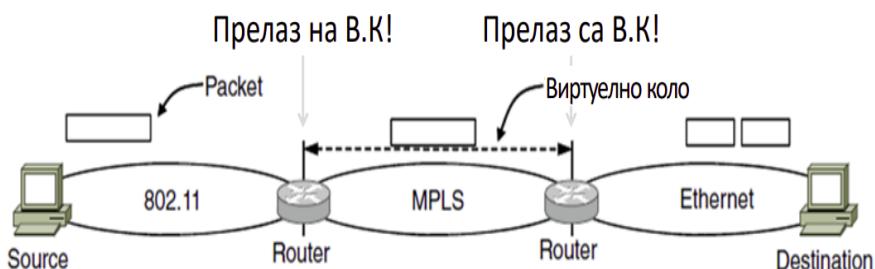
Карактеристика	Датаграми	Виртуелна кола
Припрема	Није неопходна	Неопходно
Адресирање	Пакети носе пуне адресе	Пакети носе кратку ознаку
Усмеравање	По пакету	По колу
Откази	Лакше за разрешавање	Тешко за разрешавање
Различити квалитети сервиса	Тешко за додавање	Лако за додавање

## 26.4 Povezivanje različitih mreža (međumreža)

Mreže mogu da se razlikuju u mnogo aspekata:

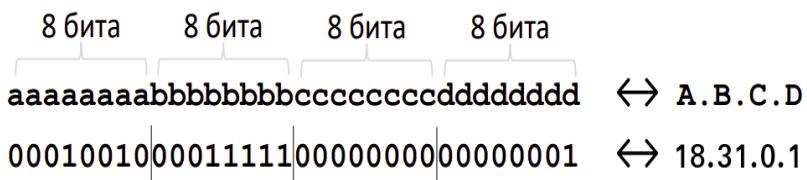
- Tip servisa (datagrami, virtuelna kola);
- Adresiranje;
- Kvalitet usluge (QOS: Quality of Service);
- Veličina paketa;
- Sigurnost (enkripcija).

Potrebno je da postoji preslikavanje između IP adresa i oznaka virtuelnog kola. Internet servis provajderi (ISP) često koriste virtuelna kola kako bi grupisali IP saobraćaj i brže preneli podatake:



## 27 IP adrese i prefiksi

Koriste se dve verzije IP adresa: IPv4 (32-bitna adresa) i IPv6 (128-bitna adresa). Adresa IPv4 se trenutno više koristi, a u procesu je prelazak na IPv6 verziju. Notacija za IPv4 adresu je u obliku kvarteta dekadnih brojeva, gde je svaki dekadni broj iz opsega [0, 255].



### 27.1 Prefiksi

Adrese se grupišu u blokove pod nazivom **prefiksi**. L-bitni prefiks je grupa adresa koje imaju isti prefiks dužine L bita. To znači da L-bitni prefiks ima  $2^{32-L}$  različitih adresa. Notacija grupa je u obliku: „IP adresa/dužina prefiksa“. Primer: 128.13.0.0/16 ima fiksiran prefiks prvih 16 bitova (prva 2 dekadna broja) tako da je opseg od 128.13.0.0 do 128.13.255.255. Primer: 128.13.0.0/32 je jedinstvena adresa.

Jedna IP adresa pripada različitim prefiksima. Što je prefiks duži, to je opseg adrese manji (specifičniji prefiks). Najspecifičniji prefiks je dužine 32 tj. konkretna adresa. Što je prefiks kraći, to je opseg adrese veći (manje

specifičan prefiks). Najmanje specifičan prefiks je dužine 0 tj. cela mreža (Internet).

#### Stari sistem za grupisanje u klase fiksne dužine:

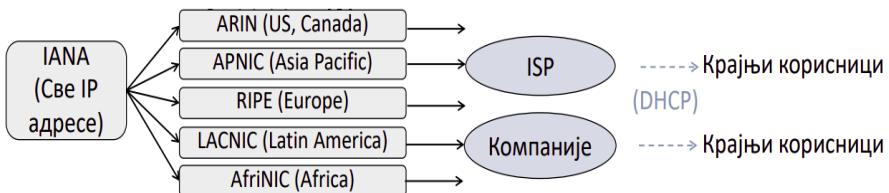
- Klasa A,  $2^{24}$  adresa.
- Klasa B,  $2^{16}$  adresa.
- Klasa C,  $2^8$  adresa (odgovara LAN mreži).



## 27.2 Privatne i javne IP adrese

- **Javna IP adresa:** Jedinstvena na nivou celog interneta. Mora se dodeliti pre upotrebe (regulatorno telo). Ove mreže su većim delom potrošene, ali polako se prelazi na IPv6 čime će neštašica biti rešena.
- **Privatna IP adresa:** Nisu jedinstvene na nivou celog interneta, ali jesu jedinstvene na nivou manjih mreža (mreža firme, kućna mreža, ...). Potrebna je bar jedna javna IP adresa i **NAT (Network Address Translation)** da bi se neko iz ovakve mreže povezao na Internet.

Svetsko regulatorno telo **IANA (Internet Assigned Numbers Authority)**, odeljenje u okviru **ICANN (Internet Corporation for Assigned Names and Numbers)** neprofitabilne korporacije, dodeljuje ceo opseg adresa regionalnim telima. Regionalna tela dodeljuju opsege kompanijama u regionu. Kompanije dodeljuju konkretnim računarima.



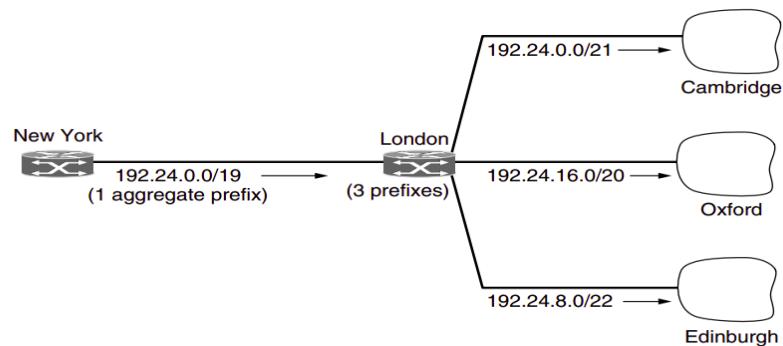
## 28 IP prosleđivanje

Internet je mreža koja ima barem milion čvorova. Jedan ruter ne može da ima tabelu svih čvorova tolike mreže (ovo je moguće za manje mreže). Odakle proizilazi problem tabela rutiranja. Jedno rešenje bi bilo da se napravi hijearhija za IP adrese: država, opština/region, grad, mreže u okviru grada i krajnji čvor. Svaki ruter treba da zna:

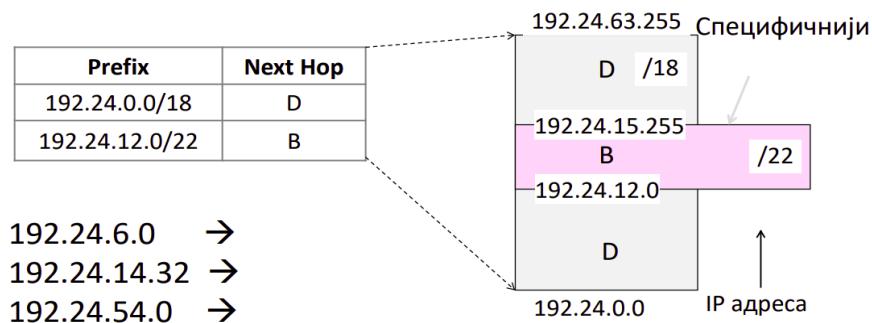
- Kako doći iz svoje u drugu državu;

- U okviru svoje države, kako doći u drugu opštinu/region;
- U okviru svoje opštine ili regiona, kako doći u drugi grad;
- U okviru svog grada, kako doći u druge podmreže grada;
- U okvir svoje podmreže grada, kako doći do drugih krajnjih čvorova.

Ovakva hijearhija bi zahtevala više od 32-bitu za realizaciju, ali postoji slična uopštenija ideja. Umesto da se manje mreže spajaju u veće mreže, manji prefiks se spajaju u veće (manje specifične) prefikse IP adresa. Ovaj proces se naziva **agregacija ruta** („route aggregation“). Svaki ruter ima tabelu prefiksa koji mogu da variraju po veličini. Ovaj sistem se naziva **CIDR** (Classless Internet Domain Routing). Ruteri imaju specifičnije prefikse za mreže koje su bliže:



Tabele rutiranja mogu da imaju prefikse koji se poklapaju. Ruter odlučuje u kom pravci šalje paket na osnovu **pravila najdužeg odgovarajućeg prefiksa**. Za svaki paket se pronalazi najduži (Najspecifičniji) prefiks koji njemu odgovara i u tom pravcu se prosleđuje.



**Kompromis vremenske i prostorne složenosti:** Kompaktnije tabele imaju podrazumevanje ponašanje (manje specifični prefiksi), zauzimaju manje prostora i manje su vremenski efikasne. Velike tabele su specifičnije (više specifični prefiksi) ponašanje, zauzimaju više prostora i vremenski su efikasnije.

## 29 ARP i DHCP

Dva problema koji se pojavljuju na mrežnom sloju su:

- Dodeljivanje IP adrese računaru u mreži (DHCP);
- i određivanje adrese u sloju veze (MAC) za ciljnu IP adresu (ARP).

Svaki računar u mreži ima jedinstvenu **MAC** (Media Access Control) adresu koja se nalazi na **NIC** (Network Interface Controller) kartici. To znači da se MAC dodeljuje hardverski svakom računaru koji treba da koristi internet usluge.

### 29.1 Dodeljivanje IP adrese (DHCP)

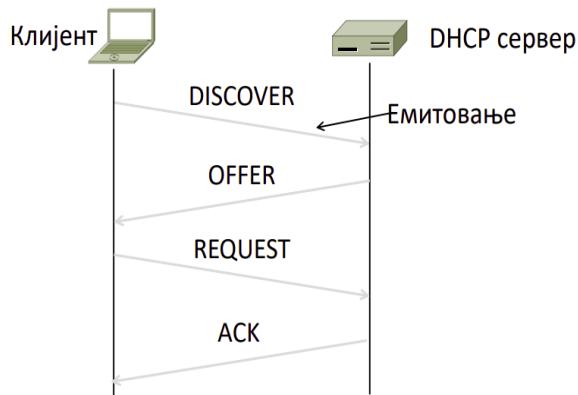
Svaki čvor u mreži mora da ima neku svoju IP adresu kako bi komunicirao na sloju mreže. Moguće je svakom čvoru dodeliti IP adresu statički ili dinamički. Dinamičko dodeljivanje adresa je uglavnom boje rešenje.

**Protokol DHCP** (Dynamic Host Configuration Protocol) dinamički dodeljuje adresu čvoru u okviru mreže. Kada se računar pokrene, on nema svoju IP adresu. Da bi dobio adresu on mora da pošalje paket DHCP serveru. Ako taj server nije direktno povezan na mrežu, onda računar emituje javnu (broadcast) poruku. Ruteri preusmeravaju DHCP zahtev DHCP serveru. Server prihvata paket sa zahtevom i dodeljuje mu slobodnu IP adresu.

Nakon nekog vremena računar napušta mrežu i IP adresa se gubi. Posledica ovoga je brzo dolaženje do nestasice IP adresa. Zbog toga DHCP protokol dodeljuje IP adresu na određeni vremenski period. Ova metoda se naziva „iznajmljivanje“ („leasing“). Pri isteku roka računar može da zaheva produženje roka. Ako se ne pošalje (uspešno) zahtev za produženje roka, onda se gubi IP adresa.

**Detaljniji proces DHCP protokola (three way handshake):**

1. **DISCOVER:** Čvor emituje paket celoj mreži (specijalna adresa za emitovanje je 255.255.255.255).
2. **OFFER:** DHCP server odgovara čvoru ciljano sa predloženom IP adresom. Ova adresa se bira na osnovu njegove MAC adrese i slobodnih IP adresa.
3. **ACCEPT:** Čvor emituje odgovor da mu odgovara predložena IP adresa (može biti više DHCP servera).
4. **ACK:** DHCP server potvrđuje i briše adresu iz spiska slobodnih IP adresa.



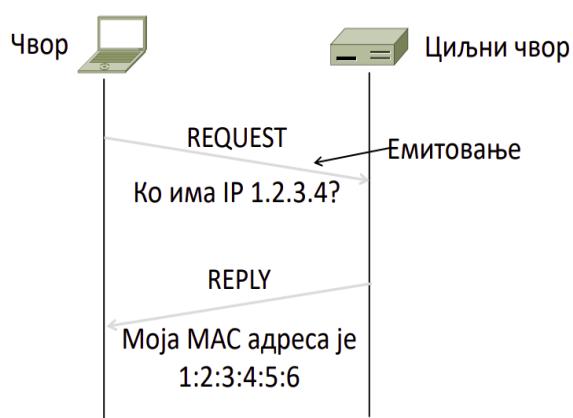
Protokol takođe omogućava paralelni rad više repliciranih DHCP servera zarađ pouzdanosti i efikasnosti. Emituje se REQUEST tako da su sihronizovani.

## 29.2 Određivanje adrese u sloju veze za ciljnu IP adresu (ARP)

Sloj veze podataka ne ume da radi sa IP adresama. Potrebno je pre slike IP adrese na adresu u sloju veze. Protokol koji vraća informaciju o MAC adresi na osnovu IP adrese se naziva **ARP** (Address Resolution Protocol). Postoji i **RARP** (Reverse Address Resolution Protocol) koji vrši inverznu funkciju.

Процес:

1. Čvor koji hoće da sazna adresu emituje ciljnu IP adresu.
2. Onaj ko ima tu IP adresu mu vraća odgovor sa svojom adresom na sloju veze.



Moguće su razne optimizacije ARP protokola. Jedna od optimizacija je da se vrši keširanje rezultata u slučaju da se opet traži ista adresa.

## 30 ICMP i NAT

### 30.1 Protokol ICMP

Operacije nad mrežom su praćene od strane rutera. Ako ruter primeti neku grešku prilikom prosleđivanja paketa, onda se taj slučaj prijavljuje pošiljaocu putem **ICMP (Internet Control Message Protocol) protokola**. Protokol ICMP se takođe koristi za testiranje Interneta (primer: *ping*). Mogu se definisati razni tipovi poruka. Svaka ICMP poruka se šalje preko paketa. Problematičan paket se odbacuje.

Paket ICMP sadrži tip greške, kod i kontrolni zbir. Postoji indikator kojim se ICMP paket razlikuje od ostalih.

Message type	Description
Destination unreachable	Packet could not be delivered
Time exceeded	Time to live field hit 0
Parameter problem	Invalid header field
Source quench	Choke packet
Redirect	Teach a router about geography
Echo and echo reply	Check if a machine is alive
Timestamp request/reply	Same as Echo, but with timestamp
Router advertisement/solicitation	Find a nearby router

## 30.2 NAT tehnika

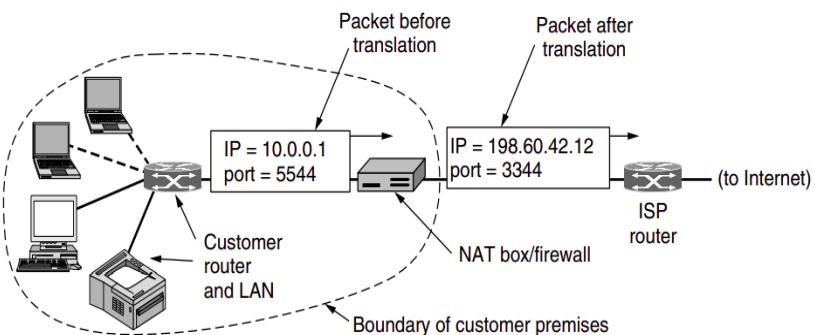
Povezivanje računara iz lokalne mreže na spoljnu mrežu (primer: Internet) se vrši preko **NAT** (Network Address Translation) tehnike. NAT se izvršava u ruterima. Protokol IPv4 omogućava par milijardni ( $2^{32}$ ) dostupnih adresa, što nije dovoljno. Nestašicom IPv4 adresa je motivisan NAT.

NAT održava tabelu preslikavanja unutrašnjih u spoljašnje (i obrnuto) adrese. To preslikavanje je zapravo IP+TCP port informacija. Portovi su ovom slučaju neophodni da bi preslikavanja bilo „1-1“.

Шта рачунар мисли		Шта ISP мисли	
Унутрашњи IP:port	Спољни IP : port	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		

Kako NAT funkcioniše:

- **Unutrašnji IP u spoljašnji IP:** Prilikom slanja podataka iz lokalne mreže, svakom IP paketu se menja adresa pošiljaoca u skladu sa zadatim preslikavanjem. („šta računar misli“ → „šta ISP misli“)
- **Spoljašnji IP u unutrašnji IP:** Prilikom prihvatanja podataka iz spoljne mreže, svakom IP paketu se se menja adresa primaoca u skladu sa zadatim preslikavanjem. („šta ISP misli“ → „šta računar misli“)



Prednosti NAT-a:

- Smanjuje potrebe za javnim IP adresama (dovoljna je jedna po domaćinstvu).
- Lako se instalira.
- Često ima u sebi neki vid zaštite od upada (firewall).

- Pomaže po pitanju privatnosti.

**Mane NAT-a:**

- Narušena je „čistoća“ slojevitosti! Radi na mrežnom sloju, a barata sa TCP portovima.
- Paketi mogu da se primaju samo ako je prethodno bilo poslatih paketa tj. mapiranje se vrši preko odlazećih paketa.
- Teško koristiti servere preko NAT-a (rešenje: port-forwarding, VPN, SSH tunneling).

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

Jedna od glavnih funkcija mrežnog sloja je rutiranje. **Rutiranje** je proces (algoritam) kojim se određuje putanja paketa tj. u kom pravcu treba poslati paket. Ključni aspekt rutiranja je alokacija protoka. Prilikom alokacije protoka treba imati na umu otkazivanje čvorova.

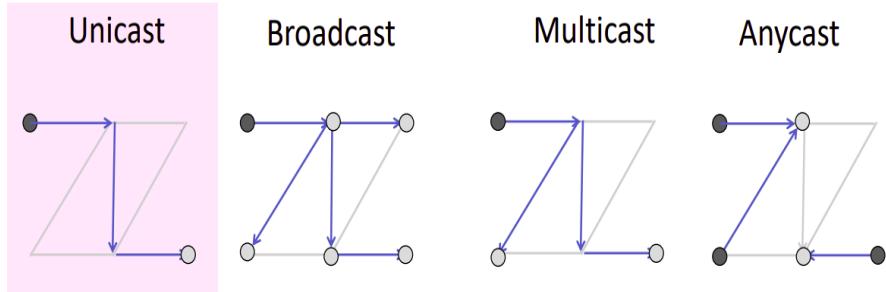
Механизам	Време реакције/ Адаптација ка
Рутирање осетљиво на оптерећење	Секунде / Критични чворови оптерећења
Рутирање	Минути / Откази чворова
Обликовање протока	Сати / Оптерећење мреже
Резервација протока	Месеци / Корисници мреже

**Статично rutiranje (neprilagodavajući algoritmi):** Putevi od čvora A do čvora B su unapred definisani (umesto da se određuju na osnovu promene topologije i zagrušenja mreže). Ovaj način rutiranja je jako osećljiv na otkaze čvorova. Koriste se samo kada je očigledno kako treba rutirati.

**Dinamičko rutiranje (prilagodavajući algoritmi):** Suprotno od statickog rutiranja, prilagodavajući algoritmi uzimaju u obzir promene topologije i zagrušenja mreže (neki jednostavniji modeli uzimaju u obzir samo promene topologije).

### 31.1 Modeli isporuke

Koriste se razlažiti algoritmi rutiranja za različite modele:



### 31.2 Ciljevi rutiranja

- **Tačnost:** Pronalazi se putanja koja radi.
- **Efikasnost:** Racionalno se troši protok.
- **Ravnopravnost:** Podjednaka prava čvorova tj. izbegava se izglađivanje čvorova.
- **Brza konvergencija:** Brz oporavak nakon promena (novi čvor, otkazan čvor).
- **Skalabilnost:** Radi dobro i kada mreža raste.

### 31.3 Principi dizajna algoritama rutiranja

Decentralizovani i distribuirani:

1. Čvorovi su ruteri. Korisnički računari se ne razmatraju (podaci od poslednjeg rutera do računara se razmatra u okviru sloja veze).
2. Svi čvorovi su ravnopravni tj. nema bitnijih čvorova.
3. Čvorovi saznaju ukupno stanje mreže tako što razmenjuju poruke sa susedima.
4. Čvorovi rade konkurentno.
5. Mogu da se dese otkazi čvorova, otkazi veza ili gubljenje poruka.

### 31.4 Rutiranje sa najkraćim putevima (najmanjim troškom)

**Princip optimalnosti:** Ako se čvor C nalazi na najkraćem putu od A do B, onda je deo tog puta od A do C najkraći put od čvora A do čvora C i deo tog puta od C do B je najkraći put od čvora C do čvora B.

Posledica principa optimalnosti: Unija optimalnih puteva u grafu čini drvo. Ovo drvo (graf) se naziva „sink tree“. Drvo ne mora da bude jedinstveno tj. moguće je da postoji više optimalnih puteva od A do B. Ako se uključe i duplikati, u slučaju da ima više optimalnih puteva od čvora A do čvora B, onda se dobija uopštenija struktura tj. aciklični graf (DAG - Directed Acyclic Graph). Teoretski, pošto je graf optimalnih puteva drvo, onda se može očekivati da svaki paket bude prosleđen konačan broj puta tj. ima konačan broj „hopova“, jer ne postoji ciklusi.

Model se može modelirati jednostavnije kao neusmereni graf (paket i idu u oba smera) i troškovi su simetrični. Princip u ovom slučaju i dalje važi.

Postavlja se pitanje kako se definiše „najkraći put“ tj. kako se mere troškovi. Troškovi se mogu meriti po dužini, ceni, kašnjenju ili kombinacijom različitih mera:

- **Kašnjenje:** Izbegavaju se zaobilazni putevi;
- **Protok:** Izbegavaju se spore veze;
- **Novac:** Izbegavaju se skupe veze;
- **Broj hopova:** Smanjuje se iskorišćenost komunikacione opreme.

### 31.5 Dijkstrin algoritam

**Uopšteno o algoritmu:**

- **Namena:** Računa najkraće puteve od zadatog čvora do svih ostalih čvorova. Rezultat je drvo (usmereni graf) ili DAG, ako se pamte svi optimalni putevi u slučaju da nisu jedinstveni (ovo je korisno prilikom zagušenja mreže ili otkazivanja čvorova).
- **Algoritam:**
  - Inicijalizuje se tabela udaljenosti tako da su svi čvorovi beskonačno udaljeni, sem korena *root* (početni čvor) tj. čvora od kojeg se gleda rastojanje (smatra se da je rastojanje čvora od samog sebe nula). Ako tabela ima  $N$  čvorova, onda treba da ima  $N-1$  „puteva“ beskonačne dužine i jedan „put“ dužine 0.
  - U svakoj iteraciji se uzima čvor koji je najbliži korenju i nije posećen.
  - U prvoj iteraciji će biti izabran koren. Ažuriraju se udaljenosti: Neka su  $s_1, s_2, \dots, s_k$  susedi najbližeg neposećenog čvora *root*. Tada se ažurira njihovo rastojanje po formuli:  

$$d_{min}(s_i) = \min(d_{min}(s_i), d_{min}(\text{root}) + d(\text{root}, s_k)),$$
gde je  $d_{min}(\text{node})$  trenutno najkraće rastojanje od korena do čvora *node*, a  $d(\text{src}, \text{dest})$  rastojanje čvora *src* do čvora *dest* (težina grane).
  - Neka je *node* sledeći čvor koji je najbliži korenju od svih neposećenih čvorova. Ažuriramo udaljenosti do suseda  $s_1, s_2, \dots, s_k$  sledećom formulom:  

$$d_{min}(s_i) = \min(d_{min}(s_i), d_{min}(\text{node}) + d(\text{root}, s_k)).$$
  - Algoritam je završen ako su svi čvorovi posećeni.
  - Vremenska složenost:  $O(E \cdot \log V)$  ili  $O(V^2)$  u zavisnosti od implementacije, gde je  $E$  broj grana, a  $V$  broj čvorova.
- **Opravdanje algoritma:** Princip optimalnosti.

## 32 Rutiranje zasnovano na vektoru razdaljine. (Distance Vector Routing)

Svaki ruter ima svoji vektor (tabelu) koji čuva najbolje poznate puteve. Ove tabele se ažuriraju komunicirajući sa susedima. Posle izvesnog vremena svi ruteri znaju sve najbolje puteve. Drugi naziv za ovaj algoritam je **Distribuirani Belman Fordov algoritam**, dobijen po autorima. **Algoritam (svaki čvor radi sledeće):**

- Inicijalizuje se udaljenost do samog sebe na 0, a do svih ostalih čvorova na beskonačno.

- Čvorovi periodičnu prosleđuju vektore udaljenosti susedima.
- Vektori udaljenosti u čvorovima se ažuriraju na osnovu vektora udaljenosti dobijenih od suseda.

Nakon prvog korak su naučeni svi putevi dužine 1. Nakon k koraka su naučeni svi putevi dužine k.

**Karakteristike:**

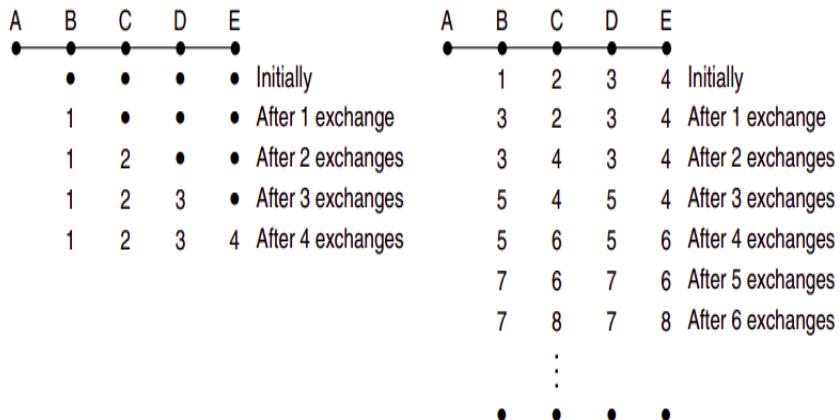
- **Dodavanje nove veze ili čvorova:** Vest putuje jedan hop po razmeni (nije preterano brzo).
- **Uklanjanje veza ili čvorova:** Susedi ne obavljaju više razmenu sa njim i posle nekog vremena zaborave da postoji. Može se uvesti maksimalna starost razmenjenog vektora.
- **Razbijanje mreže na dva dela je ozbiljan problem! Problem brojanja do beskonačnosti.**

ARPANET je koristio rutiranje zasnovano na vektoru razdaljine.

### 32.1 Problem brojanja do beskonačnosti:

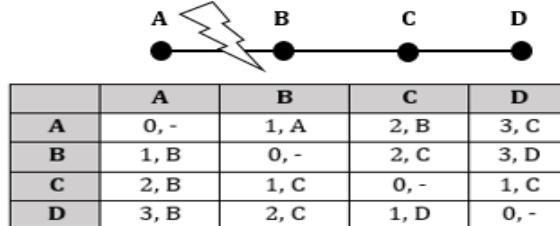
„Dobra vest putuje brzo, a loša spor...“

Algoritam zasnovan na vektoru razdaljine konvergira i time svi ruteri znaju optimalne puteve. Ova konvergencija može biti jako spora. Suština ovog problema se nalazi u činjenici da kada čvor A prosledi čvoru B neki put, čvor B ne može znati da li se on sam nalazi na tom putu.



Primer: Neka je dat jednostavan graf sa slike...

**Link Between A & B is Broken**



- Prekida se veza između čvora A i B. Čvor B zna za prekid, ali čvor C ne zna za prekid, jer nije sused.
- Nakon nekog vremena, čvor C šalje svoju DV tabelu čvoru B. Čvor B vidi da „postoji“ veza između čvora C i A, i ažurira svoju tabelu.
- Nakon nekog vremena (x2), čvor B šalje svoju DV tabelu čvoru C. Čvor C vidi da je „veza između A i B“ ažurirana i ažurira svoju tabelu.
- ...

### 33 Plavljenje (Flooding)

Svim čvorovima se emituje poruka pomoću tehnike **plavljenja (flooding)**. Jednostavan mehanizam koji nije preterano efikasan. Svaki paket se prosleđuje svim susedima. Ovo pravi mnogo duplikata i to relativno brzo (eksponencijalno). Zbog toga se paketima dodeljuje životni vek po broju hopova. Paket initialno ima K preostalih hopova, gde se svakim prosleđivanjem ovaj broj dekrementira (umanjuje za jedan). Paket se odbacuje kada broj preostalih hopova postane 0.

Koristi se lista koja čuva redne brojeve paketa koji su već pristigli u taj čvor. Paket se odbacuje ako je njegov redni broj u listi. Takođe je moguće omogućiti i ARQ kako bi plavljenje bilo pouzdanije.

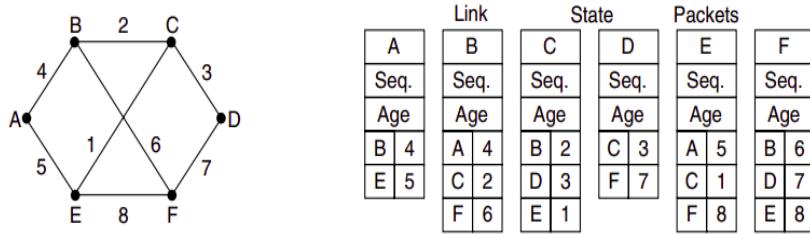
### 34 Rutiranje zasnovano na stanju veza (Link state routing)

Problem algoritma zasnovanog na vektoru razdaljine je to što je potrebno previše vremena da iskonvergira. Postoji bolji algoritam koji je zasnovan na stanju veza.

#### Algoritam (Link State Routing):

1. **Identifikacija suseda:** Kada se ruter pridruži mreži potrebno je da prvo identificuje svoje susede. Čvor šalje kroz svaku vezu HELLO paket koji primalac vraća nazad sa informacijom o njegovom globalno jedinstvenom imenu.
2. **Postavljanje težina grana:** Postavljaju se odgovarajuće težine za veze u okviru mreže. Jedan način da se odrede težine je da se računa težina koja je obrnuto proporcionalna u odnosu na protok (veći protok ima manju cenu tj. manji protok ima veću cenu).

3. **Pravljenje paket stanja (LSP - Link State Package):** Potrebno je sada da se napravi paket u kojem će biti smeštene informacije koje su sakupljene u prethodna dva koraka. Paket sadrži identifikator pošiljaoca, starost i listu suseda (sa cenzom prelaska). Primer (Paketi svih čvorova):



4. **Razmena paketa:** Osnovna ideja je da se koristi plavljenje za dejanje paketa. Da bi se razlikovale nove informacije od starih, čuva se uređeni par [identifikator paketa, redni broj]:

- Ako stigne novi paket, on se beleži i šalje se svim ostalim susedima (sem onom koji je prvo bitno poslao paket).
- Ako je paket duplikat, on se odbacuje.
- Ako je redni broj paketa manji od najvećeg viđenog rednog broja do sad, opet se odbacuje.

Ova tehnika ima mane. Kada ruter otkaze on gubi sve informacije (redni broj pre svega). Ako se opet priključi onda je njegov redni broj 0, zbog čega se svi njegovi paketi odbijaju. Takođe, ako dođe do greške u rednom broju (primer: promeni se jedan bit i time se broj promeni iz 4 u 65540) onda će svi paketi sa rednim brojem između 4 i 65540 dalje biti odbijeni.

Rešenje za ovaj problem je da se pamti starost paketa (treći korak). Starost je brojač koji se dekrementira. Ruter odbacuje informaciju kada starost dođe na nulu. Postoji još par detalja koji mogu da se promene/dodaju da algoritam radi još bolje (primer: pametnije odbacivanje duplikata).

5. **Računanje optimalnih puteva:** Sada svaka veza ima svoju cenu (cene ne moraju da budu iste u oba smera). Koristi se Dijkstrin algoritam da bi se izračunali optimalni putevi.

#### Poređenje sa DV algoritmom:

- LS ima bolju konvergenciju od DV-a, jer je plavljenje brzo.
- DV ima bolju skalabilnost od LS-a, jer čvorovi računaju rešenje potproblema, dok LS čvor računa uvek rešenje celog problema.

## 35 Višeciljno rutiranje sa najkraćim putevima (ECMP)

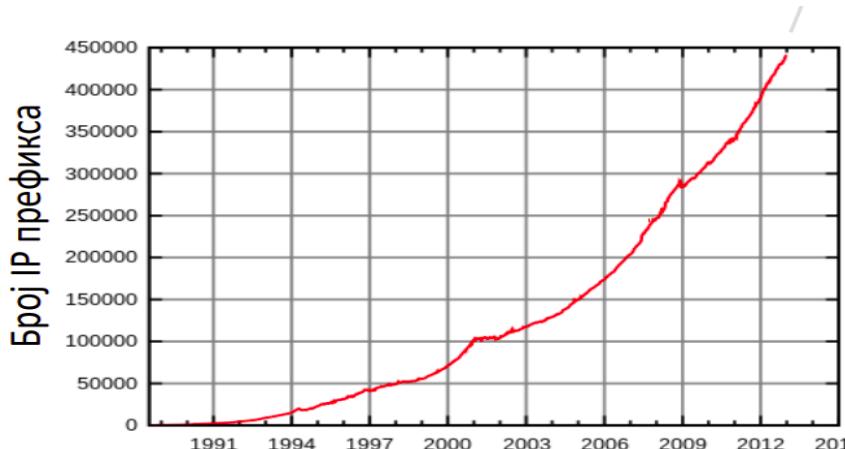
Tehnika **ECMP** (Equal-Cost Multipath Routing) podrazumeva da se čuvaju svi optimalni putevi. U tom slučaju se dobija DAG umesto drveta (spomenuto ranije). Koristi se i dalje Dijkstrin algoritam, samo što

se čuvaju svi najbolji putevi. Prednost čuvanja svih optimalnih puteva je što može da se smanji zagušenost mreže. Ako se nasumično bira optimalan put za prenos podataka, onda se dobija balansiranje opterećenja. Paketi mogu da imaju različito kašnjenje, što je loše za prenos u realnom vremenu. Bolje rešenje je fiksirati izbor na osnovu izvora i cilja. Slanje je i dalje nasumično, ali isto na nivou izvora i cilja. Opterećenje se i dalje balansira na isti način, a eliminise se upotreba različitih putanja za pakete iz istih logičkih celina (datoteka, video).

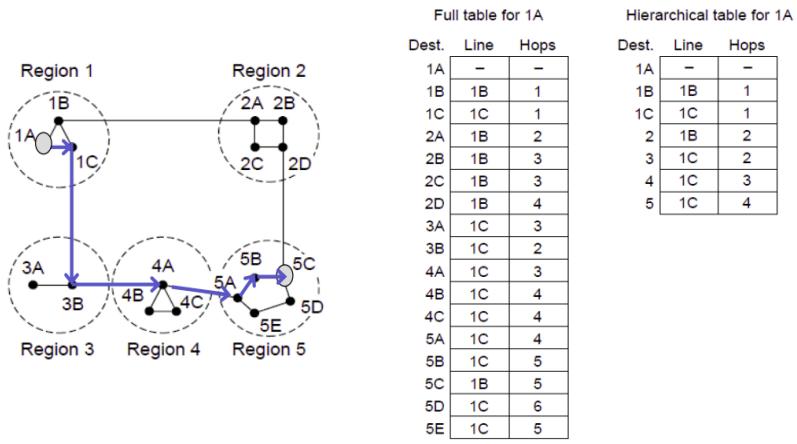
## 36 Hijearhijsko nasleđivanje

Rutiranje po svakom pojedinačnom čvoru se ne skalira dobro. Bolje rešenje je da se čvorovi grupišu u regione, a potom u okviru regiona se vrši specifičnije rutiranje.

Nema smisla praviti unos u tabeli prosleđivanja za svaki ruter na svetu:



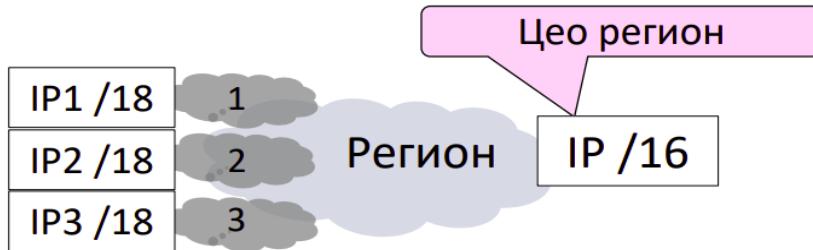
Globalna mreža se deli na manje jedinice, a te jedinice mogu da se dele na još manje jedinice (primer: država, region, ...). Konačna putanja se dobija tako što se prvo rutira paket u okviru najmanje jedinice mreže (primer: ISP). Potom se, pri izlasku, koristi rutiranje na nivou veće jedinice (region ili država). Onda je potrebno ponovo ući iz većih jedinica mreže u manje jedinice mreže gde se nalazi ciljno odredište.



Mana je što se ne pronađe uvek najkraća putanja, ali to je kompromis za bolju skalabilnost.

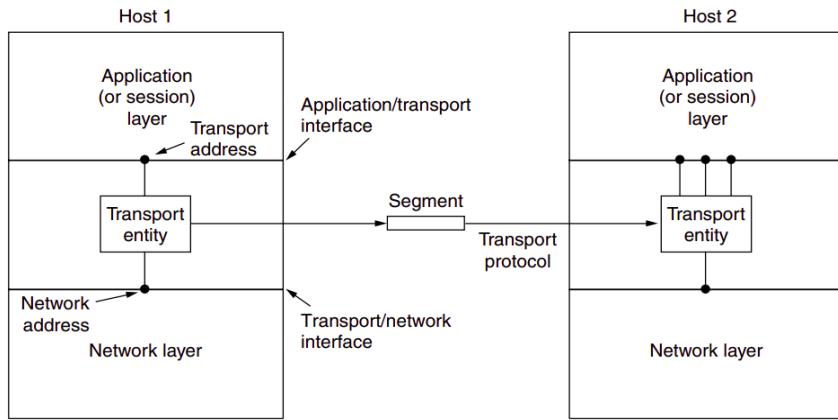
### 36.1 Podela i sažimanje IP prefiksa

Rutiranje po regionima je i dalje teško za izračunavanje. Uvodi se dodatni podnivo u vidu IP prefiksa. Unutrašnji čvorovi nisu vidljivi spolja. Moguće je i spoljne sažimanje nepovezanih ustanova (ovo obično rade ISP).



## 37 Transportni sloj: Uloga, tipovi servisa i njihovo poređenje

Glavni zadatak transportnog sloja je da pruža efikasan i pouzdan prenos podataka aplikativnom sloju. Ovaj zadatak se ispunjava koristeći servis nižeg sloja tj. mrežnog sloja. Hardver i softver (operativni sistem) za transportni sloj se nazivaju **transportna jedinica**.



Jedinica podataka na transportnom sloju se naziva segment. Segment se ugrađuje u paket, a paket u okvir.

**Zašto su transportni i mrežni sloj podeljeni na dva dela?**  
Transportni sloj radi na korisničkim mašinama, a mrežni sloj radi uglavnom na ruterima. Jednostavno mrežni sloj ne pruža dovoljno dobar servis (mogu često da se gube paketi, ruteri mogu da otkazuju, ...). Mrežni sloj nije dovoljno razrađen da bi pružao dobar servis korisnicima.

### 37.1 Tipovi servisa

Isto kao što postoje dva mrežna servisa, servis sa uspostavljenom vezom i servis bez uspostavljene veze, tako imaju i dva transportna servisa tj. servis sa uspostavljenom vezom i servis bez uspostavljene veze. Odgovarajući servis transportnog sloja imaju sličnosti sa odgovarajućim servisom mrežnog sloja. Moguće je imati servis sa uspostavljenom vezom na transportnom sloju koristeći servis bez uspostavljene veze na mrežnom sloju (ili obrnuto), ali to ne mora biti baš tako lako. Dva osnovna tipa servisa (protokola):

- **UDP (User Datagram Protocol):** nepouzdane poruke.
- **TCP (Transmission Control Protocol):** pouzdan tok podataka.

### 37.2 Poređenje servisa (TCP i UDP)

TCP je ozbiljno razrađen mehanizam (ne predstavlja nadogradnju virtuelnog kola). UDP praktično koristi datagram iz mrežnog sloja.

TCP (Токови)	UDP (Датаграми)
Остваривање везе	Датаграми
Бајтови се испоручују једном, поуздано и по реду	Поруке се могу изгубити, помешати, дуплирати
Произвољна дужина тока	Ограничена дужина поруке
Контрола тока се прилагођава пошиљаоцу и примаоцу	Шаље се без обзира на стање примаоца
Контрола загушења се прилагођава стању мреже	Шаље се без обзира на стање мреже

## 38 Socket API, primer jednostavnog klijent-server (uopšteno), portovi

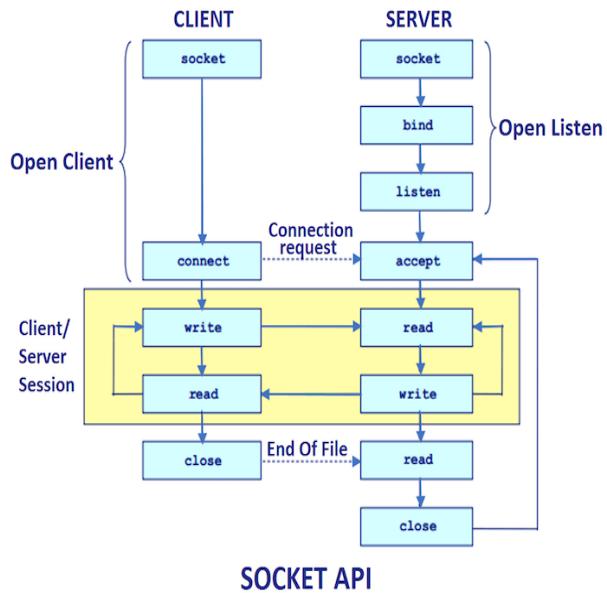
### 38.1 Socket API

Socket API je apstrakcija za upotrebu mrežnih usluga. Često se kaže i „mrežni API“, ali zapravo je u pitanju upotreba transportnih servisa. Ovo je deo svih bitnijih operativnih sistema i programskih jezika. Podržava tokove i datagrame (isti API).

Soketi omogućavaju procesima da se povezuju na lokalnu mrežu putem različitih portova. Funkcije:

- **SOCKET:** Kreira se novi „endpoint“ u alocira se potrebna memorija.
- **BIND:** Kreirani soketi nemaju adresu. Funkcijom BIND se socketu dodeljuje adresu. Kada server ima adresu, klijenti mogu da mu pristupaju
- **LISTEN:** Alocira se prostor za red u slučaju da više klijenata pokuša da se prikači u isto vreme na server (koristi se samo kod tokova).
- **ACCEPT:** Upostavlja se pasivna veza, gde ova funkcija vraća dekriptor datoteke (file descriptor) (koristi se samo kod tokova).
- **CONNECT:** Upostavlja se aktivna veza gde oba čvora mogu da komuniciraju preko SEND i RECEIVE na punom dupleksu.
- **SEND:** Šalje podatke preko konekcije.
- **RECEIVE:** Prihvatanje podataka preko konekcije.
- **CLOSE:** Prekida se konekcija.

Dijagram aktivnosti:



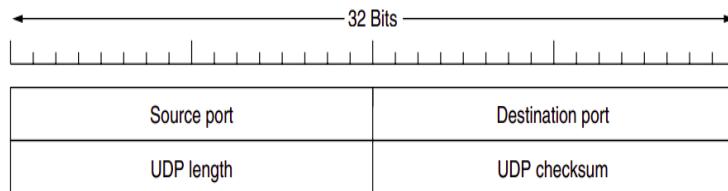
## 38.2 Portovi

Procesi se identificuju uređenom trojkom: IP adresa, protokol, port). Portovi su 16-bitni pozitivni celi brojevi. Serveri se obično vezuju za opšte poznate portove (< 1024). Klijent obično koristi nasumične portove koje bira operativni sistem (ne moraju biti opšte poznati, jer ih niko ne zna). Primeri:

- **FTP (20, 21):** prenos podataka.
- **SSH (22):** udaljeni pristup.
- **SMTP (25):** slanje i primanje elektronske pošte.
- **HTTP (80):** pristup WWW.
- **POP-3 (110):** primanje elektronske pošte.
- **IMAP (143):** slanje elektronske pošte.
- **HTTPS (443):** sigurni HTTP.
- **RTSP (543):** prenos tokova u realnom vremenu.
- **IPP (631):** deljenje stampača.

## 39 UDP

Protokol na transportnom sloju koji šalje podatke bez uspostavljene veze se naziva **UDP (User Datagram Protocol)**. UDP prosleđuje segmente koji se sastoje iz paketa i zaglavlja dužine 8 bajta:



#### Elementi zaglavnja:

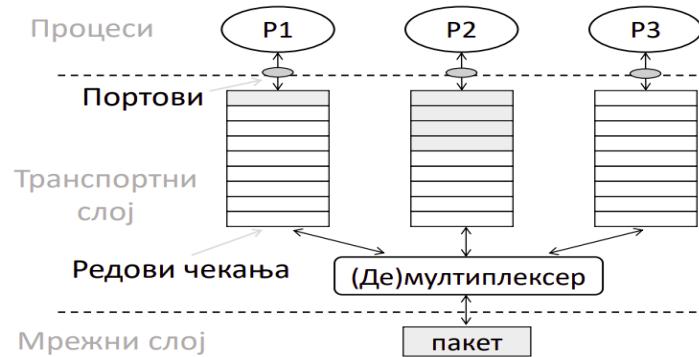
- **Source Port:** Port pošiljaoca (u slučaju da treba da se vrati neka informacija);
- **Destination Port:** Port primaoca;
- **UDP length:** Dimenzija segmenta (8 do 65515 bajtova);
- **UDP checksum:** Suma za proveru greške.

UDP ne vrši kontrolu toka, kontrolu zagušenja ili retransmisiju u slučaju lošeg segmenta. Zbog toga se UDP koristi kod aplikacija kojima nije preterano bitna pouzdanost poruka. Primer (server-klijent): Klijent šalje kratku poruku serveru i očekuje kratak odgovor. Ako za zahtev ili odgovor izgube, klijent može jednostavno da ima tajmer i pošalje opet zahtev. Posledica je jednostavniji kod i manji broj razmenjenih poruka. Primeri:

- Voice-over-IP (nepouzdano)
- DNS, RPC (zasnovano na porukama)
- DHCP (zasnovano na porukama)



Koriste se UDP baferi (redovi čekanja) u slučaju da više paketa stiže sa mrežnog sloja. Svaki port ima svoj odvojeni bafer. Kapacitet bafera može da se definiše u okviru operativnog sistema ili u okviru programa:



## 40 Uspostava i prekid mreže na transportnom sloju (uopšteno)

Pre bilo kakvog slanja ili primanja podataka, kranji čvorovi moraju biti svesni uspostave veze (moraju se dogovoriti o skupu parametara kao što je na primer maksimalna dužina segmenta). Uspostava veze podrazumeva podešavanje stanja krajnjih čvorova, usaglašavanje početnih brojeva segmentata krajnjih čvorova za korišćenje kliznih prozora. U ovom slučaju se klizni prozori koriste između krajnjih tačaka, dok se u sloju veze koristio između susednih tačaka.

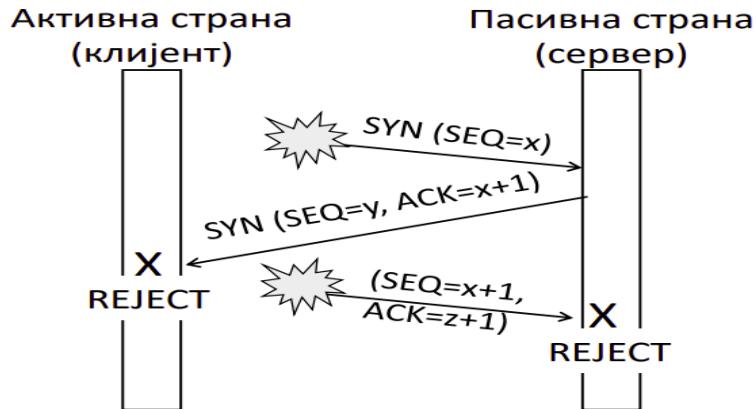
### 40.1 Proces uspostave veze

**Trofazno rukovanje (Three-way-handshake):** TCP koristi trofazno rukovanje za uspostavu veze. Proces:

1. Klijent bira neki broj  $x$  i šalje SYN segment ( $x$  je početni redni broj segmenta, obično slučajan broj iz nekog opsega):  $\text{SYN}(\text{seq} = x)$
2. Server odgovara tako što šalje broj koji sledeći put očekuje i šalje svoj početni broj  $y$ .  $\text{SYN}(\text{seq} = y, \text{ack} = x+1)$
3. Klijent potvrđuje broj.  $\text{SYN}(\text{seq} = x+1, \text{ack} = y+1)$ .



Prethodni primer predstavlja idealnu situaciju. Nažalost, neke stvari mogu da krenu naopako. Slučaj kada stiže zakasneli duplikat: Redni brojevi će se razlikovati, pa će sa obe strane duplikati biti odbijeni.



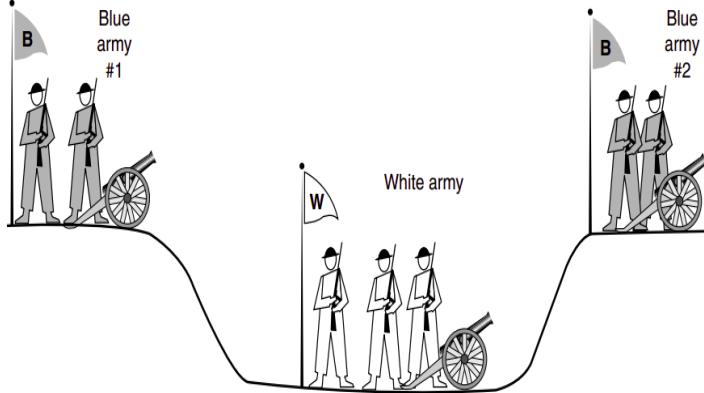
## 40.2 Proces prekidanja veze

U ovom slučaju je potrebna simetrična obustava veze (obe strane treba da prekinu mrežu). Ne sme se desiti da se jedna strana zatvori a druga ne. Jedna strana je uvek inicijator prekida (aktivna), a druga strana je pasivna strana. Ne mora klijent da bude aktivna strana, već može i server da zahteva prekid.

„Problem dve armije“:

- Bela armija sa nalazi između dva brda.
- Polovima plave armije se nalazi na jednom brdu, a druga polovina na drugom brdu.
- Jedna polovina plave armije je slabija od bele armije i sigurno gubi bitku, ali cela plava armija je jača od bele armije i sigurno dobija bitku.
- Plava armija treba da sihronizuje napad tako da napadnu obe polovine u isto vreme.
- Svaka polovina plave armije ima svog komandanta i može da pošalje pismenoš koj možda stiže na drugu stranu, a možda i ne.
- Kako se dogovoriti oko trenutka kada treba izvršiti sihronizovani napad?

Ovaj problem je analogan problemu obustave veze. Polovine plave armije su čvorovi, bela armija je kanal, a napad je prekidanje veze.



#### Potencijalna rešenja:

1. Prvi komandant šalje vreme napada i vrši kada to vreme dođe. Problem: Pismonoša je možda sankcionisan od strane bele armije pre nego što je stigao na drugu stranu.
2. Prvi komandant šalje vreme napada i čeka da mu drugi odgovori. Ako mu ne odgovori onda šalje opet pismonošu. Problem: Drugi komandant ne može da zna da li je pismonoša potvrdio poruku.
3. Trofazno rukovanje: Prvi komandant šalje vreme napada, čeka odgovor i onda šalje potvrdu da je odgovor stigao. Problem: Prvi komandant ne može da zna da li je potvrda stigla.
4. Četvorofazno rukovanje...

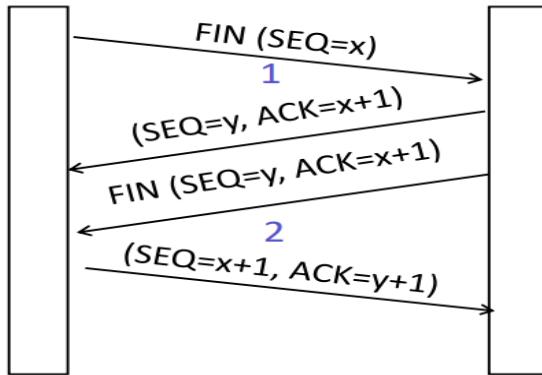
Ovaj problem nema rešenje. Jedan način da se u praksi reši problem obustave veze je da se veza automatski prekine nakon nekog vremena ako do tad nije stigao nijedan segment.

Čvorovi mogu da pokušaju da „lepo“ zatvore vezu kroz dva koraka:

1. Aktivna strana šalje FIN(x), a pasivna potvrđuje sa ACK(x+1).
2. Pasivna šalje FIN(y), a aktivna potvđuje sa ACK(y+1).

Potrebno je poslati FIN opet ako se izgubi. Svaka strana gasi svoju stranu veze nakon slanja FIN i dobijanja ACK-a.

## Активна страна                            Пасивна страна



U praksi je ovaj problem znatno lakši, jer klijent i server uglavnom unapred znaju pravila komunikacije. Server može da očekuje kada će klijent prekinuti vezu.

## 41 Protokoli kliznih prozora

**Zašto je potrebno da se i na transportnom sloju koriste klizni prozori?** U okviru mreže paketi mogu da se izgube ili pokvare. Zbog toga se vrši retransmisija i šalju duplikati. Takođe, neki paketi mogu da „zalutaju“, pri čemu se šalje novi paket koji uspešno stiže na odredište. Posle nekog vremena zalutali paket takođe stiže na odredište. Potrebno je identifikovati ovaj paket kao duplikat i odbaciti ga. U suštini, problemi su donekle slični kao na sloju veze. Glavna razlika je u tome što se ovde komunikacija odnosi na komunikaciju dva krajnja čvora preko cele putanje umesto na komunikaciju dva susedna čvora preko jedne veze.

Povećana pouzdanost na nižim nivoima doprinosi efikasnost na višim nivoima, ali nije nužno imati te mehanizme na nižim nivoima. U ekstremnom slučaju bi moglo sve da radi na transportnom sloju, ali cena retransmisija na transportnom sloju je skuplja nego na nivou sloja veze.

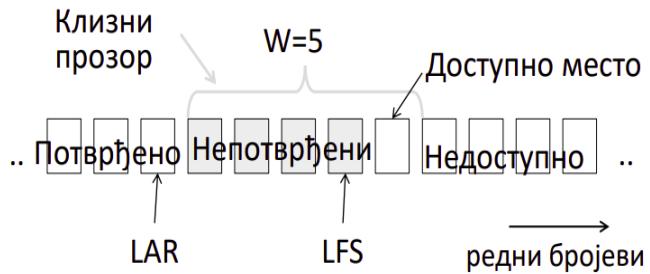
Postoji dosta varijacija ovih protokola u zavisnosti od baferisanja, potvrđivanja poruka i retransmisija. Primeri: **Vrati se N** (jednostavan, manje efikasan) i **Selektivno ponavljanje** (složeniji, više efikasan). Cilj svakog od ovih protokola je da se od aplikativnog sloja dobijaju segmenti po redu i da se aplikativnom sloju šalju segmenti po redu.

### 41.1 Pošiljalac

Pošiljalac baferiše najviše  $W$  segmenata dok ne stignu potvrde za njih:

- **LFS (Last Frame Sent):** Poslednji poslat segment.
- **LAR (Last ACK Received):** Poslednji potvrđen segment pre kojeg su svi potvrđeni.

Pošiljalac šalje segmente sve dok važi:  $LFS - LAR \leq W$



- Novi segment koji je prihvачen od aplikativnog sloja se šalje samo ako je prethodni uslov ispunjen.
- Kada se potvrdi naredni segment, prozor se pomera za jedno mesto tj. LAR se inkrementira. Time je jedno mesto u baferu oslobođeno i može da se šalje dodatni segment.

### 41.2 Primalac

**Varijanta „Vrati se N“:** Tada je **LAS** (Last Ack Sent) redni broj poslednjeg segmenta prosleđenog aplikativnom sloju. Kada stigne novi ispravan segment (redni broj LAS+1), on ga prihvati, prosledi aplikativnom sloju, inkrementira LAS i pošalje potvrdu nazad. Ako segment nije ispravan, on se odbacuje. Pošiljalac u ovom slučaju treba da ima jedan tajmer. Kada tajmer istekne, svi baferisani segmenti počev od LAS+1 se opet šalju.

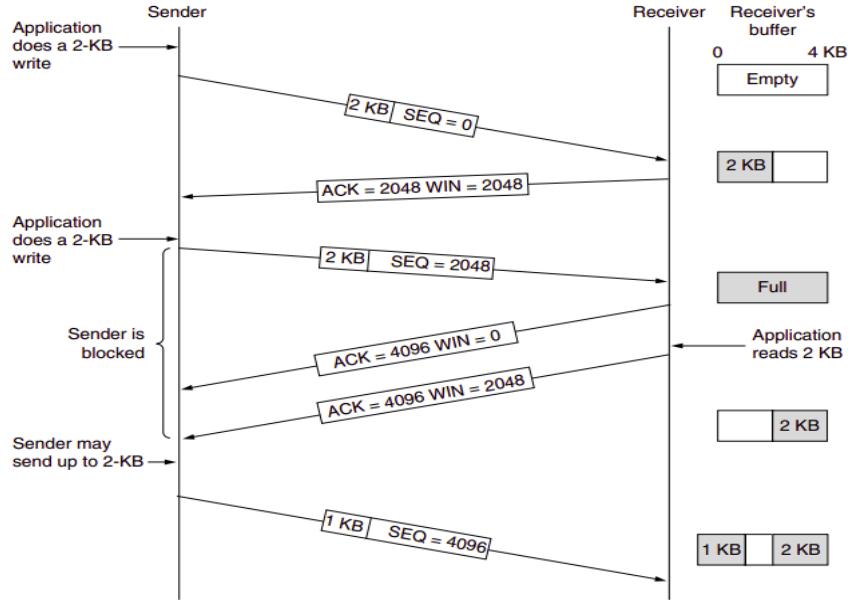
**Varijanta „Selektivno ponavljanje“:** Primalac i dalje prosleđuje segmente po redu kao što je to u prethodnoj varijatni (što je zajedničko za svaki protokol ovog tipa). Međutim, primalac koristi bafer gde može da čuva segmente koji su „došli pre reda“ (neka je bafer veličine W). Putem ACK segmenta, primalac potvrđuje najviši uređeni segment i dodatno šalje informaciju o segmentima koji nisu po redu. Bafer funkcioniše slično kao kod pošiljaoca. Prihvataju se segmenti iz opsega [LAS + 1, LAS + W], a ostali se odbacuju. Prozor se pomera kada stigne LAS+1 segment. Pošiljalac ima tajmer za svaki nepotvrđeni segment. Po isteku tajmera se taj segment šalje opet. TCP protokol koristi ovaj pristup.

## 42 Kontrola toka podataka na transportnom sloju

U slučaju da primalac sporo prima podatke u odnosu na brzinu slanja podataka od pošiljaoca (ovo se odnosi na maksimalnu brzinu) je potrebno usaglasiti tu brzinu slanja. Aplikativni sloj može da sporo prihvata segmente koji pristižu.

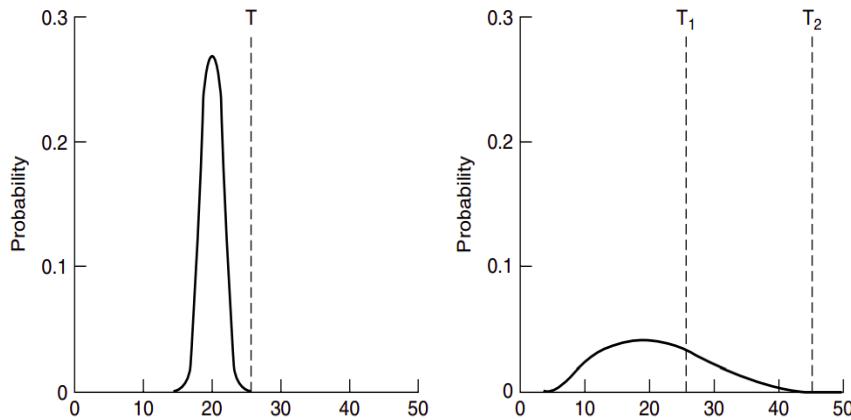
Neka primalac ima bafer dimenzije W. Inicijalno je taj bafer prazan (prethodni deo je već poslat aplikativnom sloju). Kada stignu neki validni segmenti LAS vrednost se inkrementira, ali prozor se ne pomera dok aplikativni sloj ne prihvati te segmente. Ako je pristiglo K segmenata onda bafer ima mesta za još W-K segmenata. Bafer je popunjén kada je W = K. Da bi se izbegao gubitak na strani primaoca, primalac šalje pošilja-

cu dostupno stanje bafer:  $WIN = W - K$ . Pošiljalac tada koristi WIN vrednost kao efektivnu informaciju o veličini prozora.



### 43 Retransmisija i prilagodljive pauze (tajmauti) na transportnom sloju

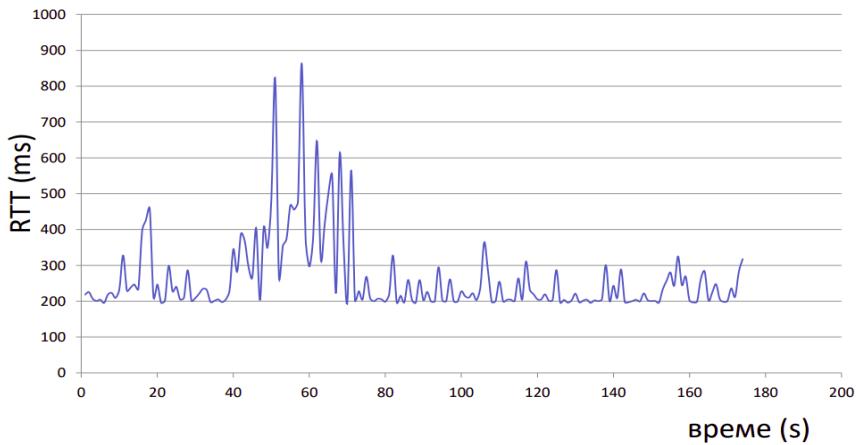
Tajmeri za retransmisiju idejno funkcionišu slično kao na sloju veze. Segment se šalje opet kada tajmer istekne. Ovaj tajmer se naziva **RTO (Retransmission TimeOut)**. Očekivano kašnjenje na sloju veze se meri u mikrosekundama i lako je predvideti zbog malog odstupanja. Na transportnom sloju za TCP je ovo odstupanje mnogo veće i teže predvideti kašnjenje (slika: levo je sloj veze, desno je transportni sloj):



Ako je tajmer previše kratak (T1), onda će nastati nepotrebne retransmisije, a ako je tajmer previše dugačak (T2), onda su lošije performanse svaki put kada se paket zagubi (slika desno).

### 43.1 Prilagodljive pauze

Dodata problem nastaje zbog promene zagušenja mreže ili putanje kojim putuje paketi zbog koje **RTT** (**Round-Trip Time**) varira tokom vremena:



Rešenje ovog problema je konstantna adaptacija dužine tajmera tj. korišćenje dinamičkog algoritma. Ideja je da oceni kratkoročni RTT i njegova varijansa. U svakoj iteraciji se ažurira **SRTT** (**Smoothed Round-Trip Time**) promenljiva koja predstavlja trenutnu procenu RTT-a sa glatkim prelazima:

$$SRTT_{n+1} = \alpha \cdot SRTT_n + (1 - \alpha) \cdot RTT_{n+1}$$

Preko faktora  $\alpha$  se definije koliko se brzo zaboravlja stari RTT (primer:  $\alpha = 0.9$ , obično se uzima  $\alpha = \frac{7}{8}$ ).

Problem: velike promene varijanse. Rešenje: Postavi se tajmer da zavisi od varijanse i od apsolutne razlike RTT i SRTT:

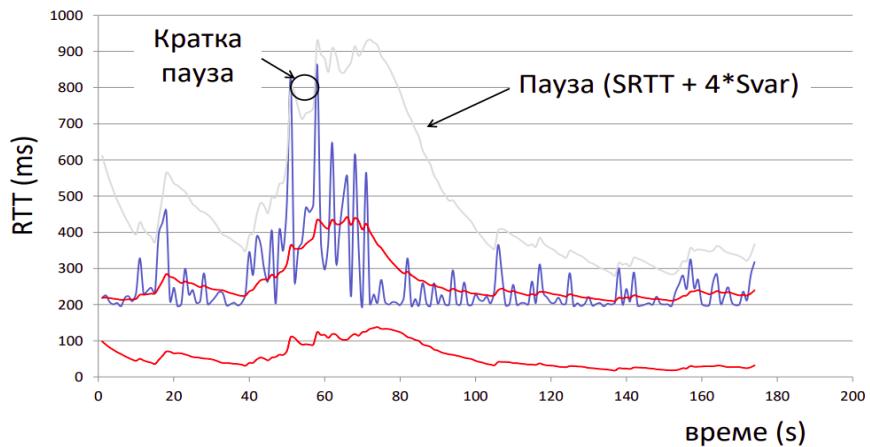
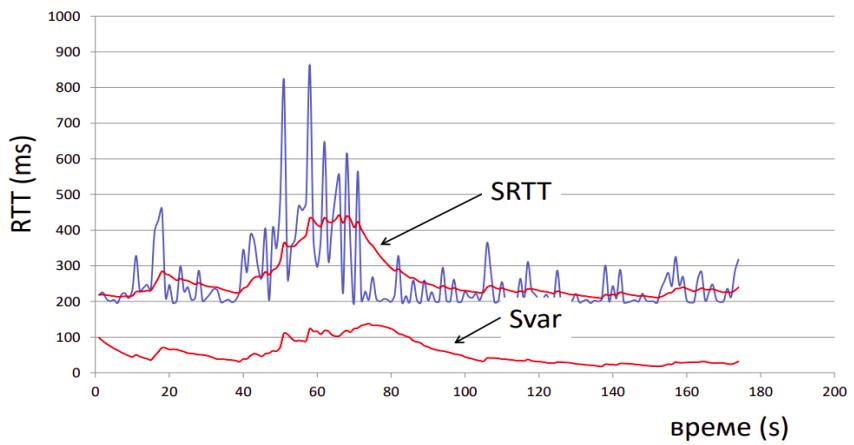
$$SVAR_{n+1} = \beta \cdot SVAR_n + (1 - \beta) \cdot |RTT_{n+1} - SRTT_{n+1}|$$

(obično se uzima  $\beta = \frac{3}{4}$ )

Konačno RTO se obično računa kao:

$$RTO = SRTT + 4 \cdot SVAR$$

Što je veća varijansa, to se teže može predvideti potrebno vreme, pa je gornja granica više udaljena.



#### 44 TCP, svojstva, zaglavlje, realizacija klinih prozora, uspostava i prekid veze (specifično).

**Napomena:** Uspostava i prekid veze su obrađeni u prethodnom pitanju gde piše uopšteno!

##### Svojstva:

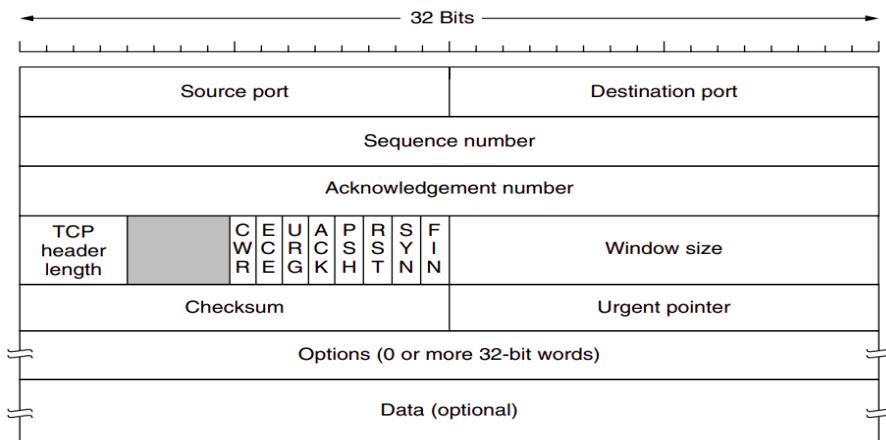
- Pouzdan tok bajtova;
- Zasnovan na vezama;
- Koriste se klizni prozori zarad pouzdanosti (sa prilagodljivim pauzama);
- Postoji kontrola toka za spore primaocve.

**Pouzdan tok bajtova:** Segmenti se mogu kretati neuređeno i nepouzdano kroz mrežni sloj i ostale niže slojeve. Međutim, transportni sloj ih

uređuje, proverava i šalje aplikativnom sloju pouzdano po odgovarajućem redosledu.

Obično se između dva čvora vrši slanje i primanje podataka. Kontrolne informacije (primer: ACK) se često šalju kao delovi dolaznih segmenata za podatke iz drugog smera (šlepanje tj. „piggybacking“).

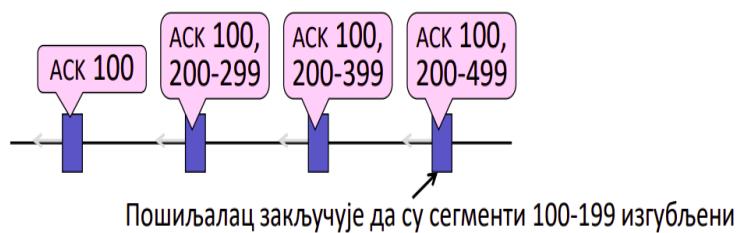
#### 44.1 TCP zaglavljje



- **Source Port:** Port izvora ( $IP + PORT = 48$ bita).
- **Destination Port:** Port odredišta.
- **Sequence Number:** Serijski broj.
- **Acknowledgement Number:** broj potvrde.
- **TCP Header Length:** Dužina zaglavlja kod TCP segmenta nije fiksne dužine (zbog opcija).
- **Osenčen deo:** Neiskorišćenih 4 bita (već 30 godina).
- **CWR i ECE:** Signali za zagušenje (1 bit).
- **ACK:** Zastavica (1 bit) koja govori da li segment sadrži ACK broj. Ako ova zastavica ima vrednost 0, onda se ACK broj u zaglavljiju ignoriše.
- **PSH:** Zastavica koja govori primaocu da mora (ako je 1) da odmah pošalje segment aplikativnom sloju bez baferovanja.
- **RST:** Zahtev za resetovanje konekcije.
- **SYN:** Koristi se za uspostavu konekcije.
- **FIN:** Koristi se za obustavu konekcije.
- **Window size:** Govori koliko bajtova može biti poslato (ako je 0, onda primalac ne želi više podataka u tom trenutku).
- **Checksum:** Za proveru greške.
- **Options:** Dodatne opcije koje se možda ne nalaze u okviru klasičnog zaglavlja.

## 44.2 Realizacija kliznih prozora

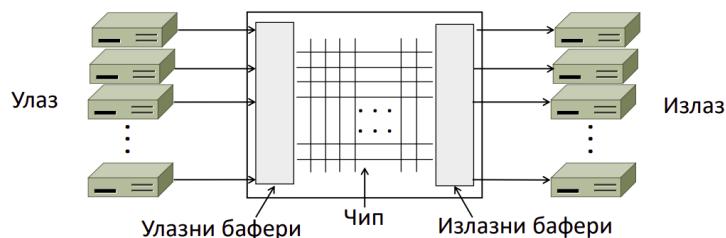
**Kumulativni ACK** govori koji je sledeći bajt koji primalac očekuje od pošiljaoca ( $LAS+1$ ). Opcionalno se koriste i **selektivni ACK-ovi (SACK)** zarad optimizacije (pošiljaocu se šalje koji segment primalac želi na osnovu onih koji je već primio). Koristi se prilagodljiva pauza za retransmisiju segmenata koji počinju od  $LAS+1$ . Koristi heurstiku kako bi brže zaključio koji su segmenti izgubljeni i time izbegao istek pauze. Primer heuristike: „Tri duplirana ACK-a“ impliciraju gubitak:



## 45 Zagušenje na transportnom sloju, opis problema i mehanizma za rešavanje (AIMD).

Kontrola zagušenja je posao moraju da obavljaju i mrežni i transportni sloj (zajednička odgovornost). Zagušenje se dešava na ruterima i to detektuje mrežni sloj, ali to zagušenje na mrežnom sloju pravi transportni sloj. Zbog toga je potrebno kontrolisati koliko se paketa ubacuje u mrežu od strane transportnog sloja.

**Priroda zagušenja:** Ruteri i skretnice koriste bafere zarad poboljšanja performansi. Organizacija bafera je FIFO (redovi čekanja). Redovi čekanja pomažu pri apsorbovanju kratkoročnih skokova u saobraćaju (Traffic Burst). Međutim, oni nisu dizajnirani za dugoročna ili srednjoročna stanja u kojima je ulazni saobraćaj veći od izlaznog. Ovakva stanja se nazivaju **загушење**.

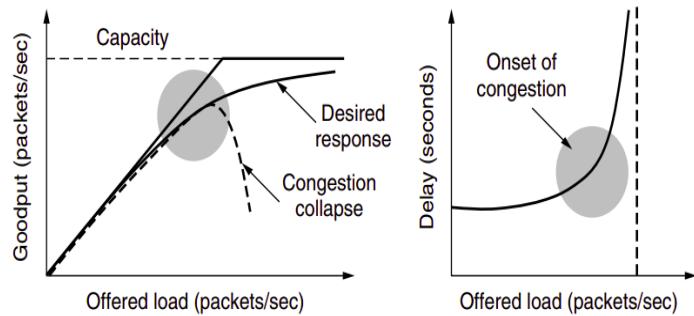


**Principi prilikom dodeljivanja kapacitet pošiljaocima:**

- **Efikasnost:** Podrazumeva da je skoro ceo kapacitet upotrebljen, ako nema zagušenja.
- **Ravnopravnost:** Podrazumeva da svaki pošiljalac dobija racionalni udeo protoka.

## 45.1 Efekti zagušenja

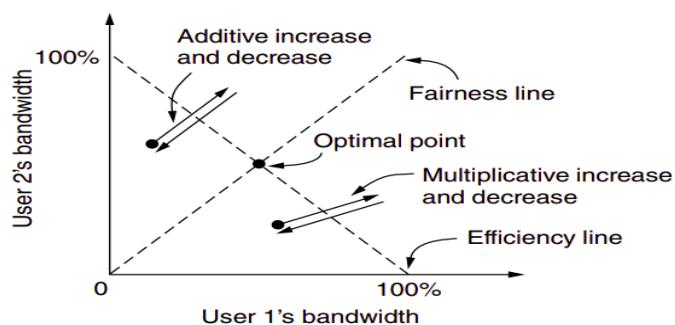
Primer: Neka je maksimalan protok na mreži 100Mbps i postoje 5 entiteta koja koriste mrežu. Svaki od entiteta bi trebalo da dobije protok 20Mbps (po prethodno navedenim principima), ali oni verovatno dobijaju manje od 20Mbps.



**„Goodput“:** Procenat korisnih paketa koji su pristigli do odredišta. Kako se broj paketa povećava, prvo se „goodput“ povećava istom brzinom, a kasnije zaostaje (leva slika). Ako je protokol loše dizajniran dolazi do **kolapsa**. Slična priča je i za kašnjenje koje u jednom trenutku počinje veoma brzo da raste (desna slika).

## 45.2 AIMD

AIMD (Additive Increase Multiplicative Decrease) je kontrolni mehanizam koji omogućava dostizanje dobre alokacije. Ideja je jednostavna, pošiljaoci aditivno povećavaju brzinu slanja podataka dok mreža ne postane zagušena, a ako je mreža zagušena, onda se brzina slanja podataka umnoženo smanjuje. Primer: Mreža ima dva čvora koji koriste ovu tehniku. Prvi čvor predstavlja X-osu, a drugi čvor predstavlja Y-osu:



Kada tačka pređe liniju koja odgovara ukupnom kapacitetu (100%), oba čvora bivaju „kažnjena“ tako što im se protok multiplikativno umanji (primer: duplo). Neka se protok čvorova duplo umanjuje kada postoji zagušenje, a povećava za 5% kada ne postoji zagušenje. Neka prvi čvor ima početnu iskorišćenost 75%, a drugi 25%. Proces:

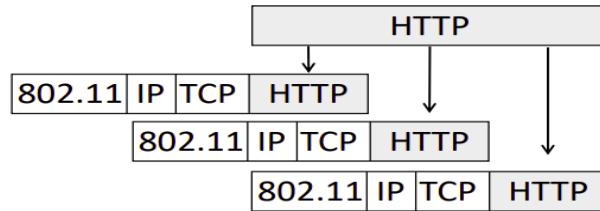
iteracija	prvi čvor	drugi čvor	ukupno	razlika
1	75%	25%	100%	50%
2	80%	30%	110%	50%
3	40%	15%	55%	35%
4	45%	20%	65%	35%
5	50%	25%	75%	35%
6	55%	30%	85%	35%
7	60%	35%	95%	35%
8	65%	40%	105%	35%
9	32.5%	20%	52.5%	12.5%

Odavde se vidi da se izbegava zagušenje i da se konvergira ka potpunoj ravnopravnosti (razlika 0%). Kada tačka dođe na  $y = x$  liniju, odatle se kreće direktno do optimalne tačke.

Pošiljaoci ne mogu direktno da komuniraju. Potrebno je da ruter signalizira čvorove o zagušenju. Ruter šalje binarno 0 ili 1 tj. da li ne postoji ili da li postoji zagušenje.

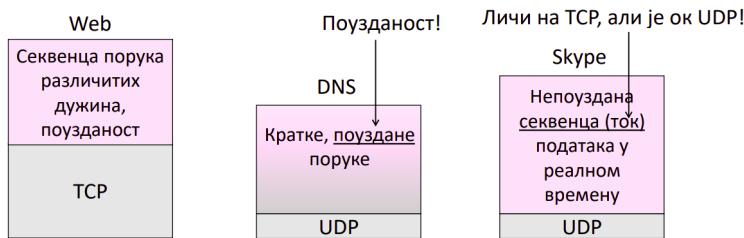
## 46 Aplikativni sloj: uloga, interakcija sa slojem ispod, pregled Internet aplikacija.

Protokoli aplikativnog sloja su često deo aplikacije (programa). Ne moraju nužno da imaju **GUI (Graphical User Interface)** (primer: DNS). Poruke aplikativnog sloja se uglavnom dele na segmente, a ti segmenti se uokviruju u pakete (poruka aplikativnog sloja se deli u više paketa).

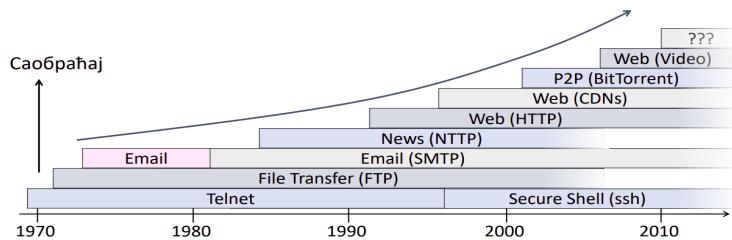


### 46.1 Interakcija sa slojem ispod

Aplikacijama treba „svašta“. Nešto od zahtevnih funkcionalnosti i ne postoji na trasportnom sloju.



## 46.2 pregled internet aplikacija



## 47 DNS: uloga, raniji pristupi, moderni pristup, TLD, slogovi

Teoretski, svi programi mogu da referišu na web strane, poštansko sanduće i ostale resurse koristeći samo IP adresu računara na kojem se odgovarajući resursi nalaze. Za prve verzije „Interneta“ (ARPANET), dok je bilo malo čvorova, ovo je bio validan način da se pristupa resursima.

Rastom Interneta došlo je do potrebe da se koriste ljudski čitljivi nazivi (imena) koja se prevode u mašinski čitljive IP adrese. Termin koji se najčešće koristi za ime koje se prevodi IP adresu je **domen**.

### 47.1 Raniji pristup

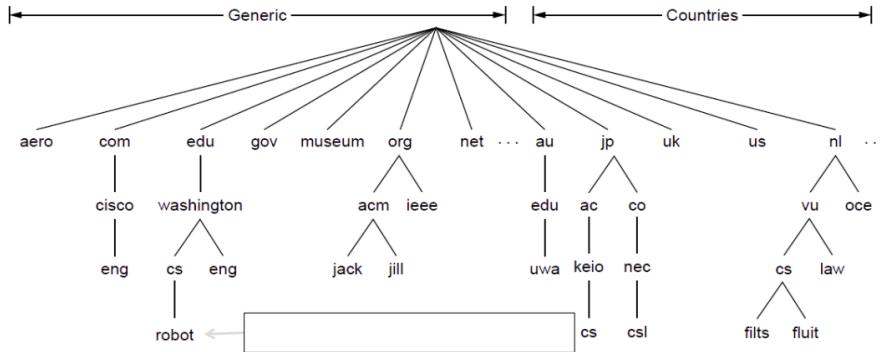
ARPANET je koristio zvaničan tekstualni fajl **hosts.txt** koji se nalazio na odgovarajućoj javnoj adresi. Svi čvorovi mreže su morali redovno (jednom dnevno) da ažuriraju lokalnu kopiju tako što pristupaju datoteci hosts.txt preko odgovarajuće adrese.

**Problemi:**

- **Loša skalabilnost:** Tekstualni fajl za ceo Internet bi bio ogroman.
- **Konflikti:** Potrebno je održavanje fajla u smislu provere konflikata, što nije praktično za toliki tekstualni fajl.

### 47.2 Moderni pristup

**DNS (Domain Name System)** je sistem (aplikacija) koja vrši preslikanje imena u IP adresu. Primer: [www.uwa.edu.au](http://www.uwa.edu.au) → 130.95.128.140. DNS sistem koristi hijearhiju domena i distribuiranu bazu podataka (domena). Ukratko, funkcioniše tako što program poziva proceduru iz biblioteke pod imenom **resolver** koja uzima ime (domen) kao argument (*gethostbyname*). Tada se šalje upit DNS serveru koji vraća odgovor. Za ovaj proces se uglavnom koristi UDP paketi.



### 47.3 TLD (Top-Level Domains)

Vrh hijearhije se održava od strane **ICANN (Internet Corporation for Assigned Names and Numbers)** organizacije. Koren čvor drveta hijearhije predstavlja Internet. Internet se deli na preko 250 **krovnih imena** tj. **TLD (Top-Level Domains)**. Postoji preko 22 osnovna TLD (.com, .edu, .gov, .mil, .org, .net, ...) i preko 250 nacionalnih TLD (.rs, .us, ...). **Zone** su osnov za dalju distribuciju. Zona je neprekidno parče prostora imena i može da se deli na podzone (delegira).

Primer: matf.bg.ac.rs:

- „rs“ je TLD koji se odnosi na Srbiju;
- „ac“ („ac.rs“) se odnosi na univerzitete u Srbiji;
- „bg“ („bg.ac.rs“) se odnosi na Beograd tj. univerzitete Beograda;
- i „matf“ („matf.bg.ac.rs“) se odnosi na Matematički fakultet.

**Napomena:** Čita se sa desna na levo.

### 47.4 DNS slogovi (DNS records)

Svaki domen (list, TLD ili unutrašnji) se čuva kao **DNS slog** u DNS bazi podataka. Za list je prirodno pronaći IP adresu kao jedan od slogova, ali tu mogu da se nađu i drugi tipovi slogova.

**Elementi:**

- **Domain Name:** Domen (može više slogova da postoji za isti domen).
- **Time To Live:** Koliko je slog stabilan. Za stabline slogove obično стоји 86400 (dani), a za nestabilne obično стоји 60 (minuti).
- **Class:** Uglavnom se koristi IN (Internet information).
- **Type:** Postoje različiti tipovi:
- **Value:** Domen ili niska u zavisnosti od tipa sloga.

Type	Meaning	Value
SOA	Start of authority	Parameters for this zone
A	IPv4 address of a host	32-Bit integer
AAAA	IPv6 address of a host	128-Bit integer
MX	Mail exchange	Priority, domain willing to accept email
NS	Name server	Name of a server for this domain
CNAME	Canonical name	Domain name
PTR	Pointer	Alias for an IP address
SPF	Sender policy framework	Text encoding of mail sending policy
SRV	Service	Host that provides it
TXT	Text	Descriptive ASCII text

```
; Authoritative data for cs.vu.nl
cs.vu.nl.      86400  IN  SOA   star boss (9527,7200,7200,241920,86400)
cs.vu.nl.      86400  IN  MX    1 zephyr
cs.vu.nl.      86400  IN  MX    2 top
cs.vu.nl.      86400  IN  NS    star          ← Сервер имена

star           86400  IN  A     130.37.56.205
zephyr         86400  IN  A     130.37.20.10
top            86400  IN  A     130.37.20.11 ← IP адресе
www            86400  IN  CNAME star.cs.vu.nl
ftp             86400  IN  CNAME zephyr.cs.vu.nl
               |           |
               flits        86400  IN  A     130.37.16.112
               flits        86400  IN  A     192.31.231.165
               flits        86400  IN  MX    1 flits
               flits        86400  IN  MX    2 zephyr
               flits        86400  IN  MX    3 top
               |           |
               rowboat      IN  A     130.37.56.201
               |           |
               |           IN  MX    1 rowboat
               |           |
               |           IN  MX    2 zephyr          ← Мейл сервер

little-sister   IN  A     130.37.62.23
laserjet        IN  A     192.31.231.216
```

## 48 DNS: zone, opis mehanizma određivanja adresa

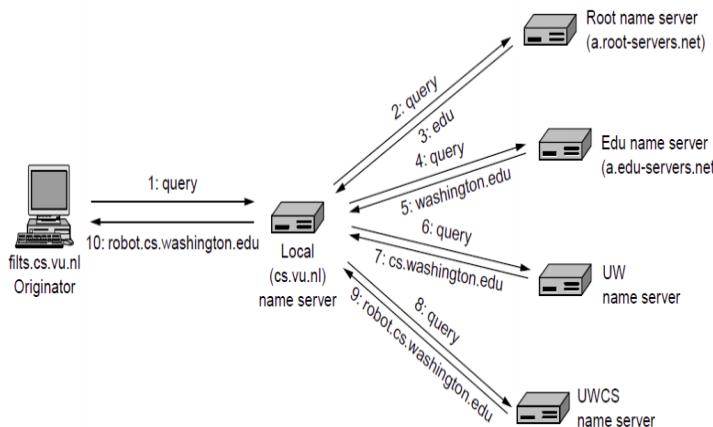
**Opšta ideja:** Računar (klijent) kontaktira svoj najbliži DNS server (lokalni DNS). Ako lokalni server ne zna traženo ime od ranije (nije keširano) on mora da traži odgovor od korenih servera domena, a onda nastavlja pretragu niže u hijearhiji. Ovaj proces se naziva „name resolution“.

Primer: Zahteva se IP adresa domena robot.cs.washington.edu od strane flits.cs.vu.nl. U slučaju da odgovor nije već keširan, mora da se pokrene niz upita.

- Prvi korak je kontakt lokalnog servera imena, cs.vu.nl (query 1). Tada cs.vu.nl postavlja upite od vrha hijearhije do dna.
- Prvi upit se postavlja korenom serveru a.root-servers.net (query 2).

Ovakvih korenih servera ima 13 (od a.root-servers.net do m.root-servers.net).

- Koreni server verovatno ne zna adresu, ali zna ime servera koji ima .edu domen (query 3), preko kojeg može da se nastavi pretraga.
- Tada lokalni server šalje upit .edu serveru imena (query 4).
- Server imena .edu mu ne daje konkretnog odgovora, ali mu daje korisne informacije za dalju pretragu pomoću UW servera imena (query 5).
- Lokalni server imena šalje upit UW serveru imena (query 6).
- Server imena UW vraća odgovor gde dalje da se vrši pretraga tj. preko UWCS servera imena (query 7).
- Lokalni server imena šalje upit UWCS serveru imena (query 8).
- Konačno dobija potpun odgovor (query 9) sa IP adresom.



### 48.1 Tipovi DNS servera

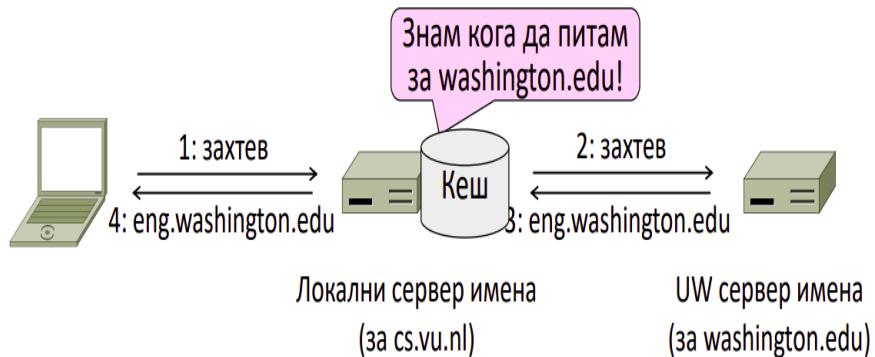
- **Rekurzivan DNS:** Rekurzivni DNS završava ceo posao za klijenta. On rekurzivno traži odgovor preko upita i vraća IP adresu klijentu. U prethodnom primeru je to lokalni DNS server. Rekurzivni DNS serveri smanjuju opterećenje onome kome treba informacija. Imaju zapamćenje (keširane) rezultate prethodnih upita, što dosta daje na performansi.
- **Iterativni DNS:** Ako ne zna za ime, iterativni DNS server samo vraća referencu na druge DNS servere gde se može pronaći direktno ili indirektno odgovor. U prethodnom primeru su to svi DNS serveri sa kojim je lokalni DNS server komunicirao. Iterativni DNS server ne zahtevaju mnogo memorije, već samo pamćenje direktnih potomaka (servera imena nižeg nivoa). Predstavljaju jedini izbor ukoliko se DNS „bombarduje“ zahtevima (kao u slučaju korenih DNS servera).

### 48.2 Keširanje

Ako je ime keširano u DNS serveru, onda on može momentalno da odgovori bez dodatnih pitanja drugih DNS servera. Takođe, mogu se čuvati

informacije o delu imena. Odgovori od DNS u sebi sadrže TTL (Time To Live) za keširanje.

Prethodni primer: Ako flits.cs.vu.nl sada traži eng.washington.edu, onda lokalni DNS server (ako je rekurzivan) će znati dobar deo puta.



### 48.3 Lokalni i koreni DNS serveri

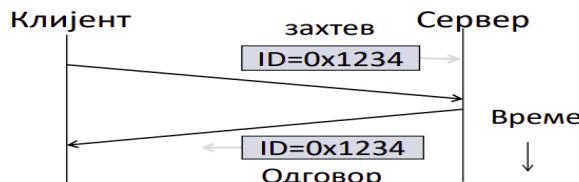
**Lokalni DNS server:** Obično u vlasništvu firme ili ISP, ali mogu biti i na klijentu ili AP (pristupna tačka). Postoje i javno dostupni kao što je na primer Google javni DNS.

Klijenti moraju da znaju ko im je lokalni DNS. Ovo se obično podešava putem DHCP u istom trenutku kada se računaru u mreži šalje sopstvena IP adresa.

**Koreni DNS server:** Prethodno navedeno, ima ih 13 različitih (a.root-servers.net do m.root-servers.net), ali su replicirani više puta na različitim mašinama (preko 250). Ovi serveri su vrlo pouzdani i efikasni.

### 48.4 DNS poruke

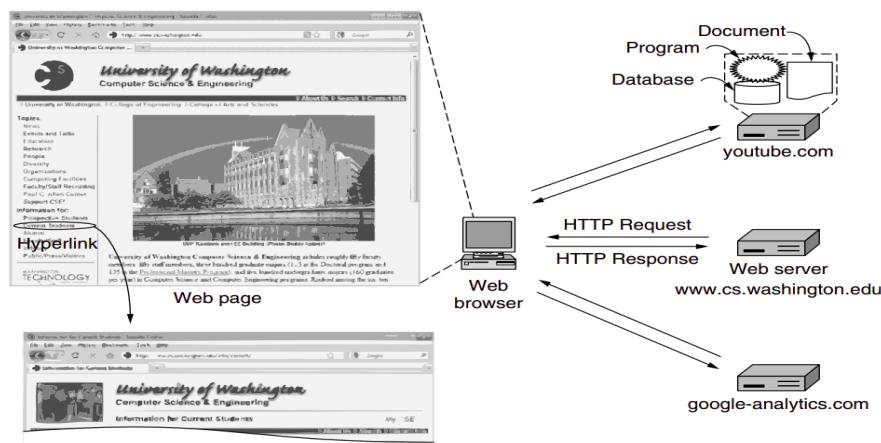
Za komunikaciju se koriste UDP sa portom 53. Koristi se ARQ radi pouzdanosti. Šalje se segment sa zahtevom (upitom) i vraća se isti segment sa odgovorom. Poruke se identificuju 16-bitnom oznakom.



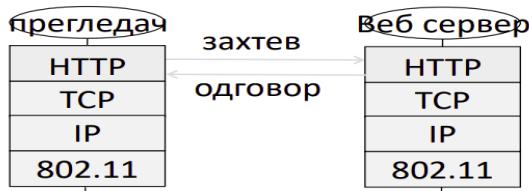
Obično se koriste višestruki DNS serveri za jednu istu zonu. Klijent se obraća jednom od njih (nekad drugom u slučaju otkaza). Ovo podiže performanse i pouzdanost (pomaže i u balansiranju protoka).

## 49 HTTP protokol: preuzimanje veb dokumenta

Iz ugla korisnika, Veb se sastoji iz kolekcije sadržaja u obliku veb stranica. Korisnik može da prelazi sa jedne stranice na drugu stranicu (ova ideja se naziva „hypertext“). Stranice se prikazuju pomoću veb pregledača (browser) čiji je posao da korisniku prikaže stranicu. U okviru stranice postoje veze preko kojih se može preći na drugu stranicu („hyperlink“).



**HTTP (HyperText Transfer Protocol)** je zadužen za dobavljanje stranica sa nekog servera. HTTP koristi TCP protokol sa transportnog sloja i obično koristi port 80. U ovoj razmeni podataka imamo dva čvora: Klijenta koji koristi pregledač za slanje HTTP zahteva i Veb server koji šalje HTTP odgovor sa odgovarajućim statusom (da li je došlo do greške).



### Tipovi dokumenata:

- **Statička stranica (Static page):** Ovaj tip dokumenta je uvek isti.
- **Dinamička stranica (Dynamic page):** Ovaj tip dokumenta je menja vremenom (potrebno je svaki put dobiti stranicu u slučaju promene ili čuvati vreme trajanja verzije kao optimizacija).

### 49.1 Proces preuzimanje veb dokumenta (klijent)

**URL (Uniform Resource Locator):** Svaka stranica na Vebu ima svoj URL koji se sastoji iz tri dela:

1. Protokol: Koji se protokol koristi (http, https, ftp, ...).

2. Domen: Ime servera na kojem se stranica nalazi.
3. Relativna putanja do specifičnog dokumenta u okviru stranice.

**Opšti oblik:** [PROTOCOL]://[DOMEN]/[RELATIVNA\_PUTANJA\_DO\_DOKUMENTA]  
 Primer: http://www.cs.washington.edu/index.html

#### Proces dohvatanja veb dokumenta:

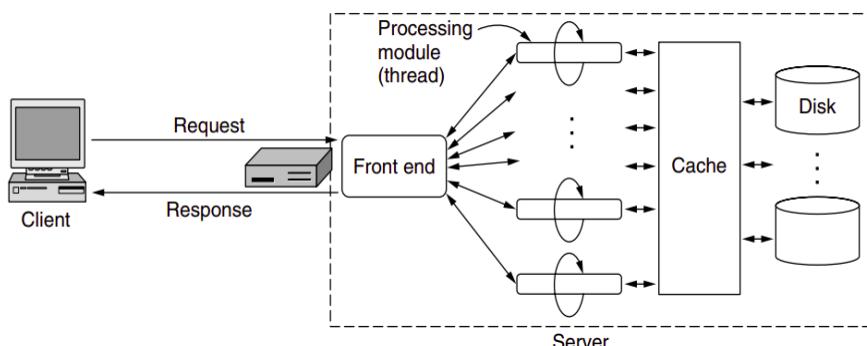
1. Pretraživač dobija URL od klijenta;
2. Pretraživač zahteva od DNS servera IP adresu koja odgovara datom domenu;
3. DNS server šalje odgovor na zahtev;
4. Pretraživač uspostavlja TCP vezu ka dobijenoj IP adresi na portu 80;
5. Šalje se HTTP zahtev serveru na dатој IP adresi (ako nije specifirano ime dokumenta, onda se traži index.html dokument);
6. Server šalje stranicu (dokument) kao HTTP odgovor sa odgovarajućim statusom;
7. Ako je za prikaz stranice potrebno još dokumenata, onda se na isti način dobavljaju ostale stranice/dokumenti sa odgovarajućim URL-om koji se koristi u okviru stranice (primer: umetnuti youtube snimak);
8. Pretraživač prikazuje stranicu korisniku, koju je dobio kao odgovor;
9. Prekida se TCP konekcija (osim ako nema još zahteva u kraćem periodu).

## 49.2 Proces preuzimanja veb dokumenta (server)

#### Pojednostavljeni proces dohvatanja veb dokumenta:

1. Prihvata se TCP konekcija od strane klijenta;
2. Određuje se dokument koji je potrebno poslati;
3. Čita se dokument sa diska;
4. Dokument se šalje klijentu;
5. Prekida se TCP konekcija.

Veb serveri uglavnom koriste po jednu nit za svakog klijenta (višenitni serveri). U tom slučaju se server sastoji iz modula koji prihvata klijente i K modula koji procesiraju zahteve:



### 49.3 Komande, kodovi i dodatne opcije HTTP protokola

Pri formiraju zahteva se koristi različite komande (prethodni proces preuzimanja se najviše odnosi na GET request, ali koraci su slični).

	Метода	Опис
Преузми документ	GET	Чита и враћа садржај Веб документа
Пошаљи податке	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

**Напомена:** „GET“ захтев може да се користи за додавање података на веб документ, где се подаци дефинишу као параметри URL-а. Овакав начин додавања података на веб документ се углавном не практикује.

Уз одговор се враћају кодови који означавају статус захтева. Уколико је све прошло како треба, онда се очекује код 200 тј. 2xx.

Код	Значење	Примери
1xx	Информација	100 = сервер прихвата 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

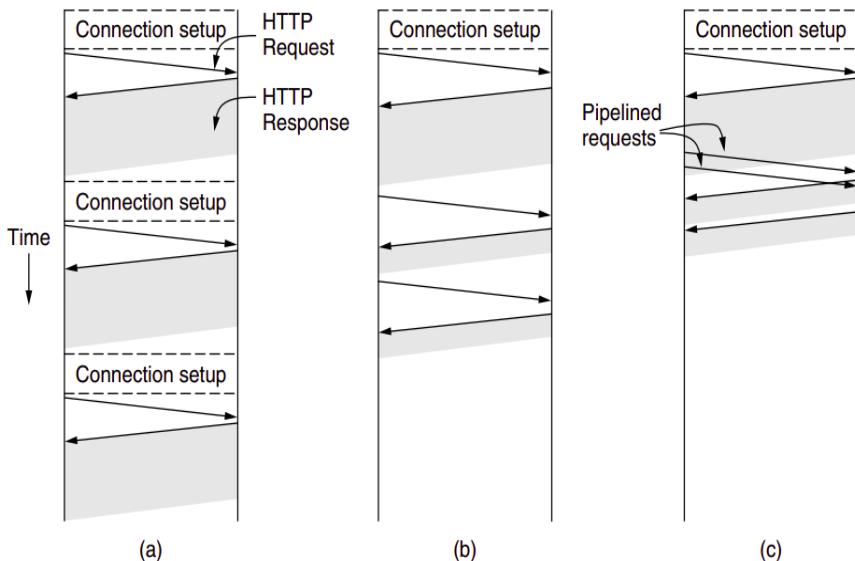
Dodatna zaglavlja zahteva ili odgovora:

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

## 50 HTTP performanse

Performanse HTTP se mere preko **vremena učitavanja dokumenta PLT (Page Load Time)** tj. vreme proteklo od klika do prikazivanja rezultata. Čak i malo povećanje PLT ima veliki uticaj na smanjenje zadovoljstva korisnika. PLT zavisi od mnogo faktora: struktura dokumenta, verzija HTTP i TCP protokola, mrežnog protoka i RTT (Round-Trip Time).

**Protokol HTTP/1.0** (ranije verzija) je koristio jednu TCP vezu za jedan zahtev. Nakon što se zahtev obradi, veza se prekida. Ima jednostavnu implementaciju, ali veoma loš PLT (slika: a).



**Protokol HTTP/1.1** podržava dugoročnu vezu tj. moguće je uspostaviti TCP vezu, poslati zahtev i dobiti odgovor i onda opet poslati zahtev i dobiti odgovor par puta pre nego što se TCP veza prekine (slika: b). Ova strategija se naziva **ponovna upotreba veze (connection reuse)**. Takođe je moguće slati paralelno zahteve i dobijati paralelno odgovore tj. poslati više zahteva bez čekanja odgovora (slika: c).

Sa slike se vidi jasna razlika između ova tri pristupa. Razlozi su intuitivni:

- Ne troši se dodatno vreme za uspostavu konekcije.
- TCP kontrola zagušenja (protok se polako povećava).

**Ostali načini za poboljšanje PLT:**

- Smanjivanje poslogog sadržaja: Manje slike, *gzip* kompresija.
- Pomeranje sadržaja bliže korisniku: CDN.

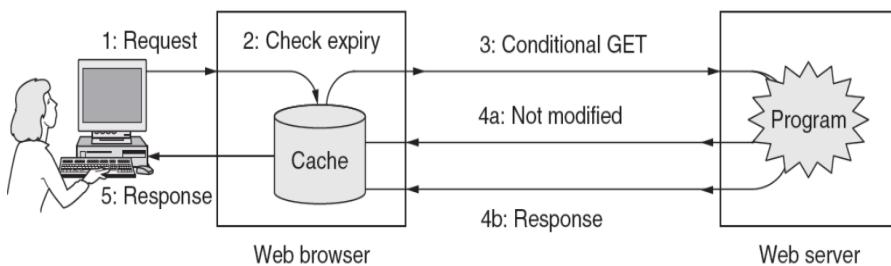
## 51 HTTP keširanje i HTTP proksiji

### 51.1 HTTP keširanje

Korisnici često zahtevaju stranice koje su već koristili. Ove stranice mogu da imaju razne slike i slične dokumente koji su skupi za prenos. Veoma je rasipnički da se ovi podaci šalju svaki put kada se zahteva data stranica.

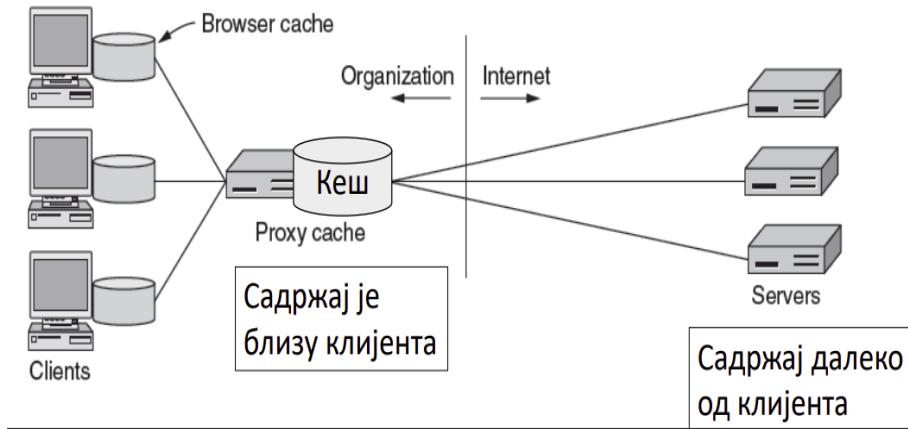
Proces čuvanja stranica koje su već korišćene za kasnije se naziva **keširanje (caching)**. Pretraživač čuva određenje, već posećene stranice za kasnije (koristi dodatnu memoriju, kompromis vremenske i memoriske složenosti). Ovim se značajno smanjuje kašnjenje u mreži.

**Kako znati da li je lokalna kopija ista kao i trenutna verzija stranice?** Kopija je ispravna ako nije istekla. Server u HTTP odgovoru dostavlja vreme isticaja sadržaja u vidu dodatnog polja (expires). Ako je kopija istekla, potrebno je proveriti da li je ispravna. Šalje se GET tj. „**Conditional GET**“ zahtev u kojem su informacije o lokalnoj kopiji. Server proverava da li je kopija validna i u skladu sa tim šalje ili ne šalje novi sadržaj. Ako ne šalje sadržaj, onda šalje kratku poruku kako bi potvrdio da je kopija validna. Moguća je i provera zasnovana na „**Etag**“ zaglavljtu (neki vid heš koda).



### 51.2 HTTP proksiji

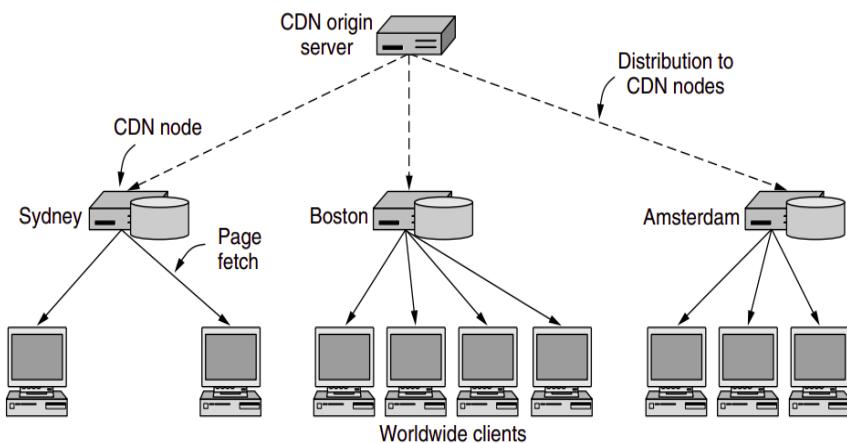
**Proksi (proxy)** je posrednik između grupe klijenata i veb servera. Različiti klijenti takođe mogu da koriste isti sadržaj tako da ima smisla keširati sadržaj i za grupu klijenata, a ne samo za pojedinca. Dodatno, u proksi se mogu ugraditi i sigurnosni mehanizmi, zabrana sadržaja (cenzura) i slično. Klijent kontaktira proksi, a proksi potom potencijalno kontaktira server:



Nekada se koristi i hijearhija proksija ukoliko se to isplati. Vrlo često je jedan proksi sasvim dovoljan posrednik. Ukoliko se koristi hijearhija proksija, proksi koji je niži u hijearhiji koristi usluge proksija koji je viši u hijearhiji tj. proksi koji je viši u hijearhiji kešira podatke za proksi koji je niži u hijearhiji.

## 52 CDN (Content Delivery Networks)

CDN (Content Delivery Networks) predstavlja dodatnu optimizaciju za efikasniju isporuku sadržaja koji se često koristi. Umesto da klijent traži kopiju tražene stranice, provajder je taj ko postavlja kopiju stranice u čvorovima koji su blizu klijenta i koji se ponašaju kao serveri. Primer:



Moguće je imati više nivoa CDN-a u slučaju da je to potrebno. CDN značajno smanjuje zagruženje mreže. Primer bez korišćenja CDN-a (gornja slika, 12 hopova) i sa korišćenjem CDN-a (donja slika, 6 hopova):

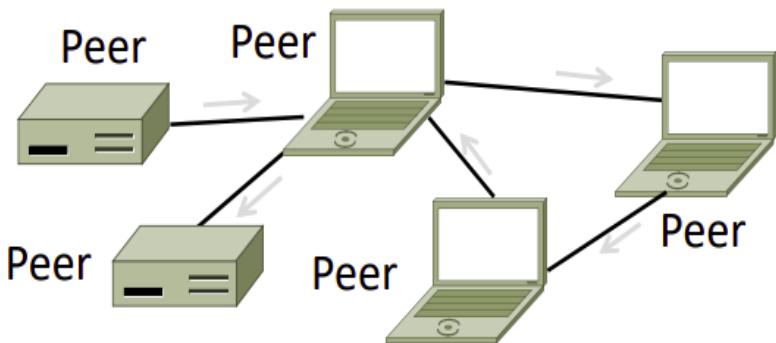


Kopije se mogu postaviti geografski ciljano i može se svesti na upotrebu DNS-a. DNS za određivanje adrese može da daje različite odgovore različitim klijentima. Svakom Klijentu se daje IP adresa najbliže kopije.

Većina kompanija ne pravi svoje CDN, već koristi servise provajdera. Najpoznatiji CDN provajder je *Akamai*.

### 53 P2P

**P2P (Peer-To-Peer)** mreža za deljenje dokumenata predstavlja alternativu za CDN, gde je sadržaj distribuiran. U ovom modelu nisu potrebni bitniji, centralizovani čvorovi, već je svako i klijent i server (isporučuje i dostavlja). Svi čvorovi su ravnopravni. Prva mreža ovog tipa je **Napster**, gde su se ilegalno delio muzički zapis. Primer: BitTorrent.



Cilj P2P je isporuka sadržaja bez centralizovanih mehanizama:

- Efikasno se ponaša i sa povećanjem broja čvorova.
- Može postići i visoku pouzdanost.

### Izazovi P2P:

- **Podsticaj za učešće čvorova:** Zašto bi čvorovi pomagali jedni drugima? Čvor isporučuje i dobavlja sadržaj. Čvor preuzima da bi pomogao sebi i šalje da bi pomogao drugima. „Ja će poslati tebi, ako ti pošaljes meni..“.
- **Decentralizacija:** Kako se pronalazi sadržaj bez centralnog registra (indeksa)? Ne postoji URL preko kojeg možemo pristupiti podacima kao u klasičnom slučaju. Čvor mora da nauči gde se nalazi sadržaj. Za to se koriste **distribuirane heš tabele** tj. **DHT (Distributed Hash Tables)**. DHT koriste algoritme za rad sa distribuiranim indeksom. Indeks je distribuiran preko svih čvorova. Indeks za traženi sadržaj daje spisak svih čvorova koji ga traže. Svaki čvor ima pristup indeksu.

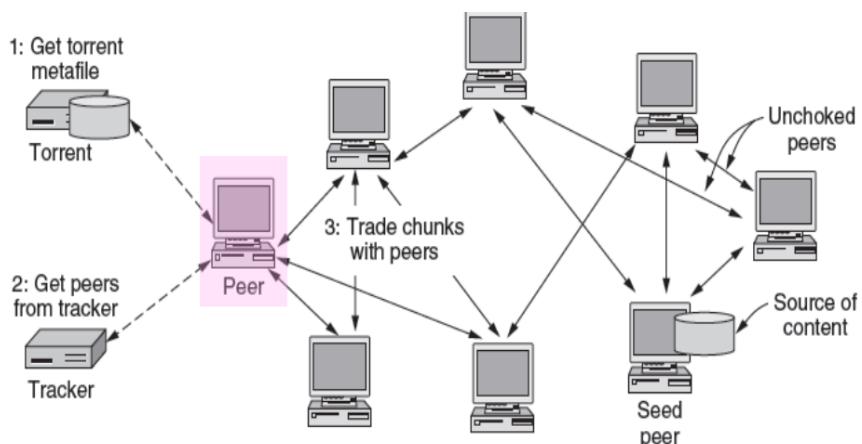
### 53.1 BitTorrent protokol

BitTorrent je standardni P2P sistem koji se danas koristi. Ima veoma brz rast i mogućnost prenosa velikih datoteka. Čini veliki deo Internet saobraćaja danas. Koristi se za legalni i ilegalni saobraćaj.

Podaci se dostavljaju posredstvom „**torrentata**“. Datoteka .torrent sadrži meta podatke, poput spiska datoteka, veličine, itd. Sadržaj se šalje u delovima zbog paralelizma. Ovi delovi se ne uzimaju redom. Spisak čvorova ne mora da bude DHT, on može biti naveden tzv. „**tracker**“ serverima (Računarama čija je namena da pamte spiskove čvorova koji poseduju određenu datoteku. Danas se sve manje koristi zbog pojave DHT).

**Proces (koraci u radu protokola):**

1. Započinje sa torrent datotekom: U njoj su ili lokacije „tracker“ servera ili potrebne informacije da bi se kontaktirao DHT.
2. Razmena podataka sa različitim čvorovima: Čvor ima delove sadržaja koji treba drugim čvorovima, a drugi čvorovi imaju delove sa sadržaja koji treba tom čvoru (trgovina).
3. Što se više šalje, to se više i dobija: Odlazni saobraćaj prema nekom čvoru će pratiti dolazni saobraćaj sa njega (obustavlja se rad prema čvorovima koji ne sarađuju).



**Neke činjenice:**

- Čvorovi koji imaju sadržaj ili žele da ga preuzmu se povezuju i za-počinju razmenu...
- Delovi koji se šalju su obično nasumično odabrani. Ako bi se slalo redom, mogućnost trgovine bi se značajno smanjila.
- Obustavlja se saobraćaj prema čvorovima koji ne sarađuju.