

RAČUNALNIŠKE KOMUNIKACIJE

OMREŽJE

Kaj je omrežje ?

- **fizična definicija:** resurs zmožen povezovanja **velikega števila** naprav
- **storitvena definicija:** infrastruktura nudi storitve **porazdeljenim aplikacijam**

porazdeljene aplikacije:
www, VoIP, email, igre, P2P,
elektronsko poslovanje

KOMPONENTE OMREŽJA:

- **končni sistemi:** odjemalci in strežniki
- **jedro omrežja:** usmerjevalnik oz. **router**
- **komunikacijske naprave**

dostop do omrežja:

- **modemski oz. klicni dostop:** preko **telefonskega omrežja**, zasedenost telefona med uporabo
- **digital subscriber line oz. DSL:** tudi uporablja telefonsko infrastrukturo, **individualen dostop** tho!
- **kabelski dostop:** **TV infrastruktura**, več odjemalcev si deli dostop do **skupnega vozilšča**
- **optični oz. FTTHB:** skoraj **direktna** povezava do doma, zagotovljena **hitrost**
- **ethernet:** priklop preko **stikala**, običajno na **javnih zavodih**
- **WiFi:** **deljen** in neusmerjen medij
- **3G 4G LTE 5G:** centrale **mobilnih operaterjev**

končni sistemi:

- **udeleženci** v omrežju
- namizni **računalniki**, strežniki itd.
- **odjemalci**, strežniki ali **mešano** npr. P2P
- morajo **dostopati** do omrežja

PROTOKOL

- **dogovor o obliki in poteku** komunikacije
- izmenjava sporočil med udeleženci
- imamo **višenivojski** in **niženivojski**
- splošna uporabnost z **standardizacijo**
- **IETF:** standardi v obliku dokumentov **RFC**
- ima **plasti:** spremembra implementacije dela sistema je neodvisna od ostalega sistema

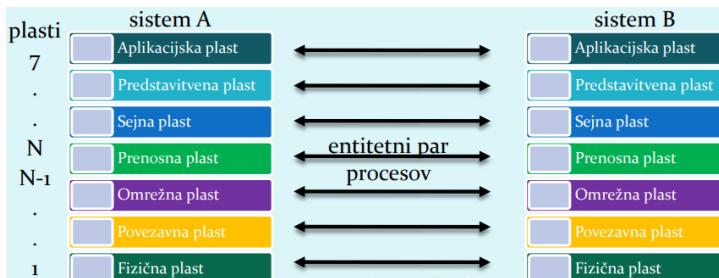
jedro omrežja:

- mreža **povezanih usmerjevalnikov**
- dva načina komunikacije
- **povezan način:** circuit switching, namenska povezava za vsak prenos, ima različne **faze**
- **nepovezan način:** prenos podatkov po kosih

niženivojski: zaporedje bitov, kontrola zamašitve, kontrola hitrosti, način potovanja paketa po omrežju itd.

višenivojski: TCP, IP, HTTP, FTP, SMTP, POP3, BitTorrent, ...

FIZIČNA PLAST



- 7 plasti definirajo sklope sorodnih funkcij komunikacijskega sistema
- plast N nudi storitve oz. **streže** plasti N + 1
- plast N zahteva storitve oz. **odjema** od plasti N – 1
- **protokol:** med istoležnima procesoma
- **entitetni par:** par procesov ki komunicira na isti plasti

1. **fizična:** prenos bitov po komunikacijskem kanalu (kodiranje, multipleksiranje)
2. **povezavna:** okvirjanje bitov, kontrola pretoka, popravljanje napak, asinhrona/sinhrona komunikacija
3. **omrežna:** usmerjanje, posredovanje, izogibanje zamaštvam
4. **transportna:** zanesljivost prenosa, učinkovitost
5. **sejna:** logično povezovanje procesov znotraj aplikacij (aplikacijsko multipleksiranje, pogosto implementirano v aplikaciji)
6. **predstavitvena:** kodiranje podatkov, kompresija, sintaksa
7. **aplikacijska:** podatki aplikacije, storitve HTTP, SMTP itd.

ISO OSI: de iure, teoretičen, sistematičen, pomanjkanje implementacij
TCP / IP: de facto, prilagodljiv, nesistematičen, fleksibilen, veliko izdelkov

prenosni sistem in kanal:

- **prenosni medij:** naprava za razširjanje valovanja
- **prenosni sistem:** uporablja prenosni **kanal**
- **prenosni kanal:** prenese bite oz. **okvir** po mediju

prenosni mediji

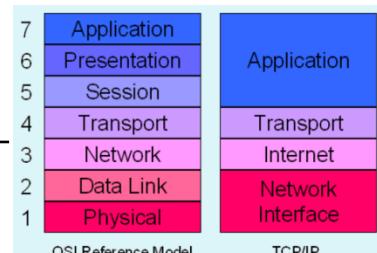
- **zvita parica** oz. **UTP:** dve vzporedni izolirani bakreni žiki, krajev razdalje
- **koaksialni kabel:** bakrena žica, izolacija, drugi vodnik, še ena izolacija, odpornost proti motnjam, ni sevanja
- **optično vlakno:** mehanska občutljivost, zahtevno spajanje, za prenos več signalov po enim vlaknu uporabimo več valovnih dolžin
- **brezžične:** radijske, mikrovalovne, IR, satelitske
- **frekvenčna karakteristika:** kakšne frekvence lahko medij prenese, posledice fizikalnih vplivov in omejitve

fizična plast:

- **kodiranje bitov:** z neko fizikalno veličino za prenos po mediju
- **prenos** posameznih bitov v **analogni** ali **digitalni** obliki
- prenos **celotnega signala**
- pretvorba signala v **obliko** za prenos po **mediju**

kvadratna modulacija

- skupaj **amplitudne** in **fazne**
- več **nivojev** amplitude
- 4 fazni koti: 0, 90, 180, 270
- posamezna spremembra signala v praksi označuje skupino 3 do 6 bitov

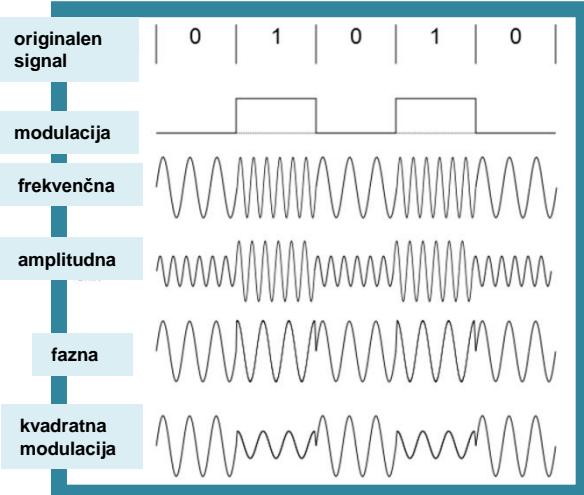


lastnostni prenosnega kanala:

- **smer:** dvosmeren ali enosmeren
- **zaporednost:** serijski in paralelni
- **število točk:** dvotočkovni in skupinski

MODULACIJA

- **analogno** kodiranje **digitalnega** signala
- **amplitudna** modulacija: glasen in tih pisk
- **frekvenčna** modulacija: visok in nizek pisk
- **fazna** modulacija: spremembe faze



Sprememba faznega kota	Amplituda	Kodirana vrednost	Diagram
0	nizka	000	Diagram showing a circle with points 000 at 0°, 010 at 90°, 100 at 180°, 011 at 270°, and 101 at 360°.
90	nizka	001	
180	nizka	010	
270	nizka	011	
0	visoka	100	
90	visoka	101	
180	visoka	110	
270	visoka	111	

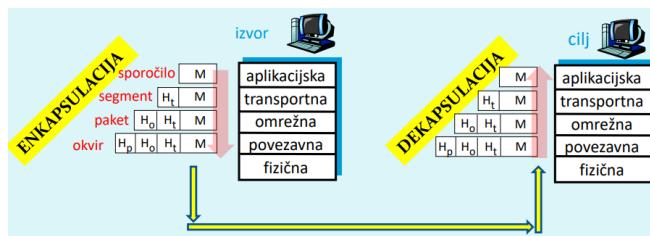
FIZIČNA PLAST

povezavna plast:

- enota ki se prenaša na povezavni plasti je **okvir**
- prenos po **povezavi** med **sosednjima vozliščema** glede na **tip medija**

OKVIR

- **enota:** podatkov na povezavni plasti
- opredeljuje **začetek** in **konec** prenesenih podatkov
- podatkom doda glavo oz. **header** in rep oz. **trailer** ki so potrebni za uspešen prenos



1. **okvirjanje datagramov:** podatkom višje plasti se doda glava in določi struktura
2. **zaznavanje in odpravljanje napak:** z dodatnimi biti lahko zaznavamo, ali je prišlo do napake pri prenosu okvira; v določenih primerih jo lahko odpravimo
3. **dostop do medija:** če je medij deljen, se uporablja MAC protokol (*media access control*) in ustrezno naslavljanje udeležencev
4. **zagotavljanje zanesljive dostave:** uporaba potrjevanja in ponovnega pošiljanja v primeru napake pri prenosu na povezavi
5. **kontrola pretoka:** usklajevanje hitrosti pošiljanja glede na procesorske sposobnosti prejemnika

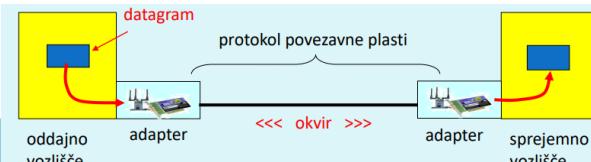
PROTOKOLI

- primeri **protokolov:** Ethernet, wireless LAN (802.11), token ring, PPP
- svoja oblika izmenjanih **podatkov** med **vozlišči**

implementacija povezavne plasti:

- **oddajnik:** enkapsulacija datagrama v okvir, detekcija, kontrola pretoka ...
- **sprejemnik:** preveri napake, pretok, dekapsulacija

- **dvotočkovna povezava:** vsaka povezava enega pošiljalnika in prejemnika
- **oddajna povezava:** deljeni medij, več vozlišč komunicira naenkrat



PROTOKOLI ZA DOSTOP DO SKUPINSKEGA MEDIJA

- **oddajni broadcast kanali:** nujen za koordinacijo dostopa
- upoštevamo principe iz **realnega sveta**
- če dve vozlišči oddajata naenkrat pride do **trka** oz. **kolizije**

idealni protokol za kanal z hitrostjo R:

- če oddaja **eno** samo vozlišče je **hitrost R**
- oddaja **M** vozlišč **povprečna hitrostjo R / M**
- protokol je **decentraliziran**
- protokol je **enostaven**
- **izkoristek, pravičnost, kolektivnost**

1. delitev kanala (ni kolizij):

kanal razdelimo na "podkanale" (frekvenčno ali časovno) in vsakega dodelimo paru vozlišč

izogibanje kolizij

2. naključni dostop (kolizije so):

vsak lahko oddaja kadarkoli, če pride do kolizije, jo razrešujemo

3. izmenični dostop (ni kolizij):

vozliščem izmenično dodelujemo pravico do pošiljanja

delitev kanala:

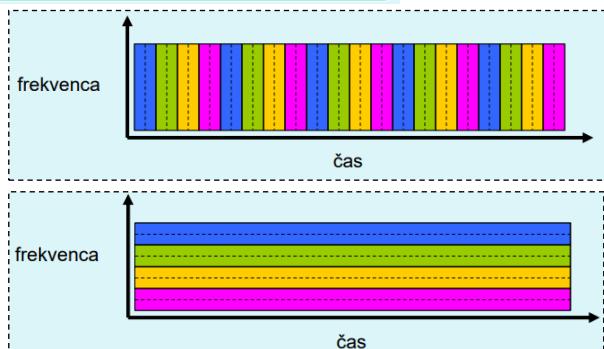
1. time division multiple access: **TDMA**
2. frequency division multiple access: **FDMA**

naključni kanala:

- določajo kako zaznati **kolizijo** in kako ukrepati
- kadar želi pošiljati uporabi polno **razpoložljivo hitrost kanala R**
- pred pošiljanjem **ni koordinacije** med vozlišči

delitev kanala je koristna pri visoki obremenitvi, pri majhnih je neizkorisčenost kanala, potrebno čakanje na vrsto

PRIMERI NAKLJUČNEGA KANALA



PROTOKOL ALOHA:

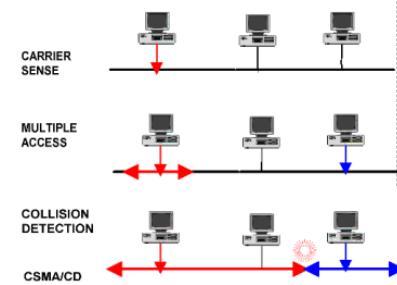
- paket je ranljiv ves čas oddajanja, je preprost in ni sinhronizacije
- obravnava kolizije: paket se prenese do konca in ponovno pošlje po preteknu naključnega intervala časa

RAZSEKANA ALOHA:

- čas je razsekana na enake časovne intervale
- vozlišča so sinhronizirana, pošiljajo samo ob začetku intervalov

CSMA:

- **vztrajni**: če je kanal zaseden, posluša dokler se ne sprosti
- **nevztrajni**: šele po času ponovno prisluhne
- **p vztrajni**: vztrajno posluša, ko se kanal sprosti, z verjetnostjo p odda paket
- gledamo ali nekdo že govori preden spregovorimo



principi iz realnega življenja so:

- daj vsakemu priložnost, da govor
- ne odgovarjaj, razen če te kdo ne ogovori
- ne izvajaj monologov
- dvigni roko, če imaš vprašanje
- ne prekinjaj nekoga, ko ta govor
- ne spi, ko ti nekdo govor

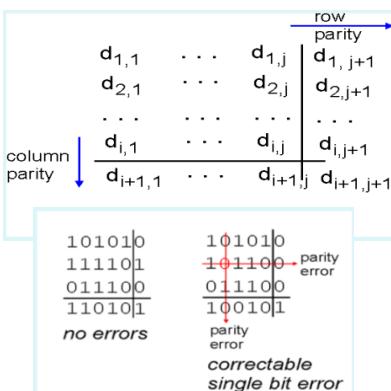
PARNOST

- **soda oz. liha** paritetna shema
- **liha paritetna** shema **0** če imamo v podatkih liho število **enic** in **1** če **sodo**
- omogoča samo zaznavanje lihega števila napak
- **parnost v 2 dimenzijah**: biti za vsako **vrstico** in **stolpec**
- zaznavanje in odpravljanje **enojnih** ali celo **dvojnih** napak



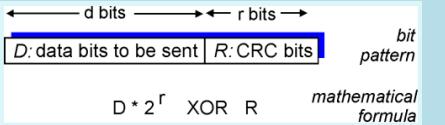
$d \text{ data bits} \rightarrow \text{parity bit}$

0111000110101011 0



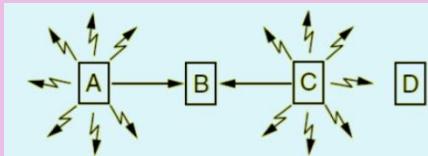
CRC

- uporaba **kontrolnih vsot**
- matematična metoda ki uporablja **polinome**
- r dodatnih bitov
- zaznava in popravi napake do $r + 1$ bitov

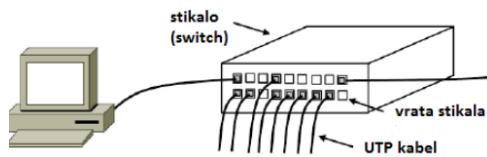


dostop do brezžičnega medija

- **CSMA / CA**: izogibanje trkom z usklajevanjem časov pošiljanj
- protokol za **naključni dostop**
- CS ne deluje dobro ker imajo dostopne točke **različna področja pokritost**



- uporabljen v **802.11 WiFi**
- uporaba **signalov za rezervacijo** medija
- imamo **2 posebni situaciji**

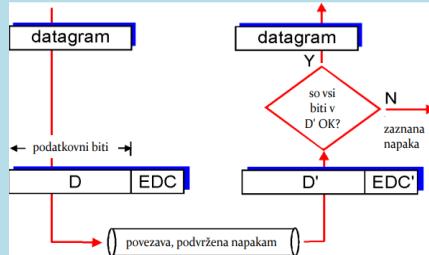


zaznavanje in odpravljanje napak:

- **motnje** na kanalu: presluh, slabljenje, šum
- signal se lahko **okvari**

EDC: Error Detection Code

- dodamo še **dodatne bite** za preverjanje pravilnosti protokol za popravljanje lahko **spregleda** napake
- **več EDC** bitov omogoča boljšo **detekcijo**



HAMMINGOVA KODA

- dodamo **kontrolne bite**
- primer: 0101 dodamo 3 in imamo **sodo paritetno** shemo

korekcija napake

→ 0 1 0 0 1 1 1

1 2 3 4 5 6 7

$p_1 = 0+0+1+1 = 0$, $p_2 = 1+0+1+1 = 1$,

$p_4 = 0+1+1+1 = 1$

Syndrome = 1 1 0, flip position 6

Data = 0 1 0 1 (correct after flip!)

- 7 bit code, check bit positions 1, 2, 4

- Check 1 covers positions 1, 3, 5, 7

- Check 2 covers positions 2, 3, 6, 7

- Check 4 covers positions 4, 5, 6, 7

0 1 0 0 1 0 1 →

1 2 3 4 5 6 7

$p_1 = 0+1+1 = 0$, $p_2 = 0+0+1 = 1$, $p_4 = 1+0+1 = 0$

Syndrome = 000, no error

Data = 0 1 0 1

- rezervacija s **žetonom** oz. **token** ki **kreži**
- zakasnitev
- enotna točka odpovedi: **žeton**

protokoli za izmenični dostop

- namesto faze boja za medij je faza **rezervacije**
- imamo **2 pristopa**



→ rezervacija s **centralnim vozliščem**

→ **zakasnitev**

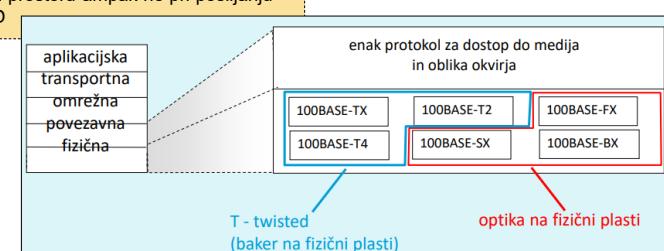
→ enotna točka odpovedi: **centralno vozlišče**

ETHERNET

- RTS: request to send
- CTS: clear to send

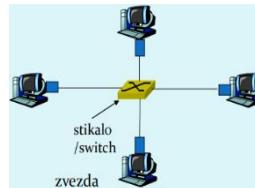
- **skriti terminali**: A in C in B in D na sliki na primer, ker terminala nista v dosegu ampak lahko vseeno ustvarita kolizijo in točki B
- **izpostavljeni terminali**: B in C sta si izpostavljeni, terminala sta v dosegu in ustvarjata trke v vmesnem prostoru ampak ne pri pošiljanju iz B v A in C v D

- hiter, poceni in enostaven
- različni **standardi** in **fizični mediji** npr. baker / optika
- ista oblika **okvirja**
- isti **MAC protokol**



TOPOLOGIJA ETHERNETA

- včasih vodilo
- vsi vmesniki so bili na isti kolizijski domeni
- najprej koaksialno vodilo
- nato topologija zvezda: naprava hub
- zvezda: stikalo v centru
- ločene kolizijske domene
- možne različne različice na vsakem kraku



OKVIR ETHERNET

- preamble: $7 \cdot 10101010$ in $1 \cdot 10101011$
- namenjeno sinhronizaciji ur oddajnika in prejemnika
- MTU: max transmission unit

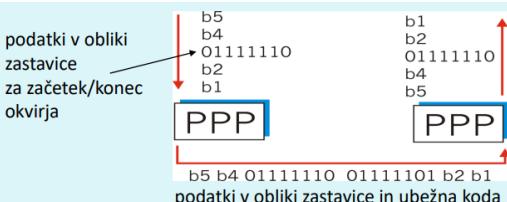
kaj ponuja Ethernet?

- nepovezavna storitev: vmesnik pošlje brez faze vzpostavljanja
- nezanesljiva storitev: kontrola pravilnosti je CRC, potrebovanje in ponovno pošiljanje se ne uporablja, omrežna in transportna plast skrbita za vrstni red in prenos vseh podatkov
- uporablja CSMA / CD: posluša pred oddajo, v primeru trka preneha, uporablja eksponentno povečevanje čakanja na naslednjo oddajo
- pri koliziji se uporablja jam signal, za oznanitev trka drugim

PPP PROTOKOL

- drugi povezavni protokol
- point to point: povezave med dvema točkama
- se ne uporablja na broadcast povezavah
- en pošiljatelj in en prejemnik
- ni potrebe po protokolu za dostop do medija in MAC naslovi

- podatki v okvirju PPP lahko poljubne oblike
- primer: če podatki vsebujejo niz 01111110 ki je začetek oz konec okvirja, uporabljamo vrvanje oz stuffing
- pred 01111110 vrinemo ubežno kodo oz. escape sequence
- ta koda opozori prejemnika da zaporedje, ki sledi, ni zastavica
- prejemnik ubežno kodo odstrani pri sprejemu



ADRESS RESOLUTION PROTOCOL

- vsako vozlišče ima tabelo ARP
- vsebuje 3 podatke <naslov IP | naslov MAC | TTL (20 min)>
- ARP deluje samo na lokalnem podomrežju in ne v celiem internetu

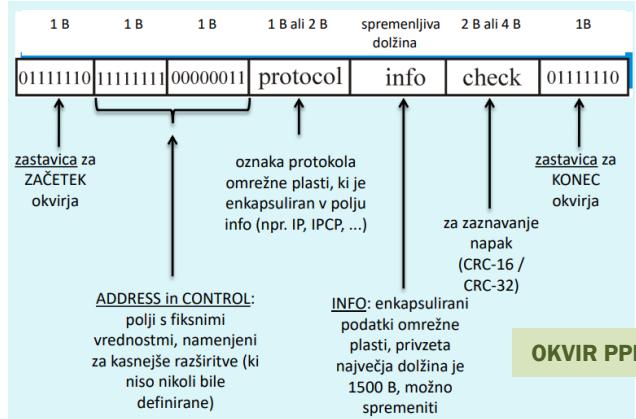
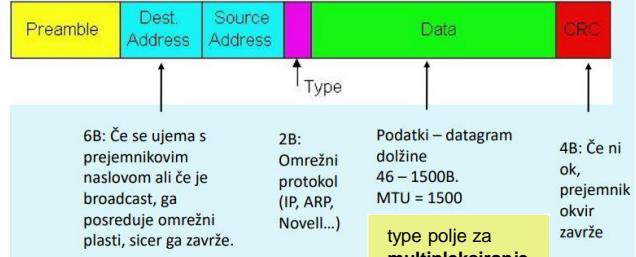
nastanek ARP tabele:

- A želi poslati datagram vozlišču B
- naslova MAC še nima v tabeli ARP
- A pošlje paket za ARP poizvedbo
- ga naslovi na vse vmesnike – broadcast
- premejo jo vsa vozlišča priključena na medij
- iz naslova IP poizvedbe vmesnik B zazna da paket sprašuje po njem
- B odgovori vmesniku A z odgovorom ARP
- pošlje svoj naslov MAC samo vozlišču A
- A shrani novi podatek v svojo tabelo
- princip plug and play

POPLAVLJANJE na VSA vrata: če ne vemo kje je prejemnik

POSREDOVANJE na IZBRANA vrata: če vemo kje je prejemnik

FILTRIRANJE: okvir je namenjen istim vratom ga zavremo

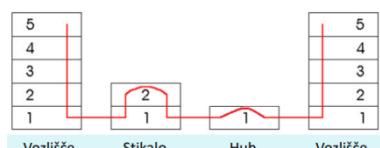


OKVIR PPP

NASLAVLJANJE NAPRAV

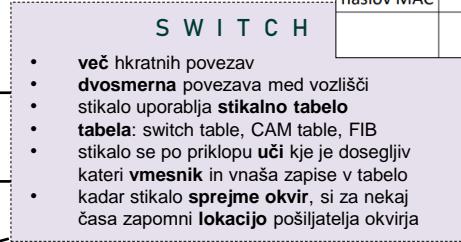
- naprave imajo naslove, ki so sestavljeni iz 48 bitov: fizični naslov ali MAC naslov zapišemo z 12 HEX znaki torej možnih 2^{48} naslovov
- prva polovica je ID proizvajalca, druga polovica ID adapterja
- MAC naslov je zapečen v adapterju in je unikaten
- naprave morajo razpoznati, ali je okvir na mediju namenjen njim
- naslov FF-FF-FF-FF-FF-FF je broadcast prejemniki so vse naprave naslavljanje računalnikov v Internetu poteka z uporabo IP naslovov
- MAC naslovi so fizični in stalni za napravo
- IP naslovi so logični in zamenljivi oziroma odvisni od lokacije priklopa

primerjava	Hub	Stikalo
Izolacija prometa	Ne	Da
Potrebna konfiguracija?	Ne	Ne



aktivna oprema:

- repeater: ponavljalec oz. ojačevalce opreme na fizični plasti
- hub: razdelilec oz. zvezdišče ponavlja signal na vseh ostalih vratih iz isto hitrostjo, ne shranjuje okvirjev, ista kolizijska domena
- switch: omrežno stikalo, preklaplja med priključenimi segmenti, shranjuje okvirje in aktivno ukrepa na podlagi vsebine, posredovanje, poplavljanie in filtriranje



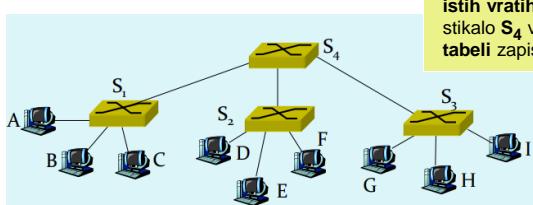
- prvi sprejem okvirja:**
- stikalo shrani lokacijo pošiljalnika A
 - stikalo ne ve kje je cilj zato poplavi oz. ang. flood okvir na vsa vrata

nan prejemnik:

- stikalo prejme okvir za znanega prejemnika
- posreduje okvir na prava vrata

povezovanje stikal:

- pravilnega posredovanja se naučimo z učenjem **stikalnih tabel**



filtriranje: stikalo **S₄** bo **zavrglo** okvir ki ga A pošlje C ker je na istih vratih če ima stikalo **S₄** v **stikalni tabeli** zapis za C

OMREŽNA PLAST

USMERJEVALNIK

- transport **datagrama** po **jedru** omrežja
- povezave med različnimi **mediji** in **protokoli**
- izvajajo **usmerjanje** in **posredovanje**
- posredovanje:** prenos paketa iz **vhodnega** na **izhodni** vmesnik usmerjevalnika in poteka znotraj usmerjevalnikov saj ima ta **posredovalno tabelo**
- usmerjanje:** določitev poti paketov od **izvora** do **cilja** sodelujejo vse omrežne naprave na poti

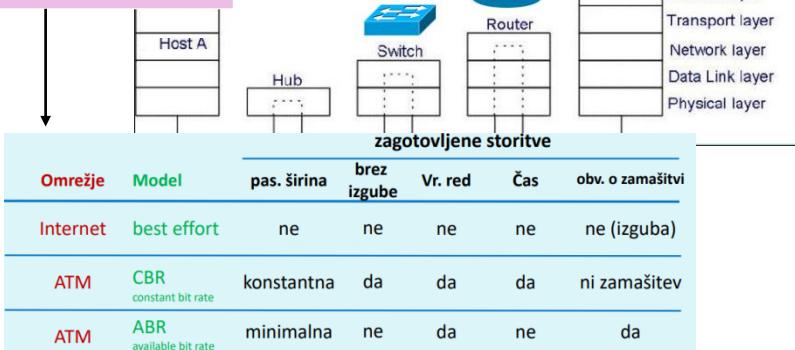
- zagotovljena **dostava** paketov
- dostava v **zagotovljenem času**
- dostava v **pravem zaporedju**
- zagotovljena spodnja meja **pasovne širine**
- največja **varianca zakasnitve**
- zagotavlja **varno komunikacijo**

$$t_{\text{pošiljanja}}(P_2) - t_{\text{pošiljanja}}(P_1) \approx t_{\text{prejetja}}(P_2) - t_{\text{prejetja}}(P_1)$$

naloga 1: transport segmenta od **končnega pošiljalnika** do prejemnika

naloga 2: enkapsulacija segmentov transportne plasti v **pakete** na strani pošiljalnika

prisotna v vseh omrežnih napravah in v **jedru** omrežja oz. **usmerjevalnikih**



p o v e z a v n a i n n e p o v e z a v n a o m r e ž j a

omogočajo vzpostavitev **zvez** v omrežni infrastrukturi med **pošiljaljem** in **prejemnikom**

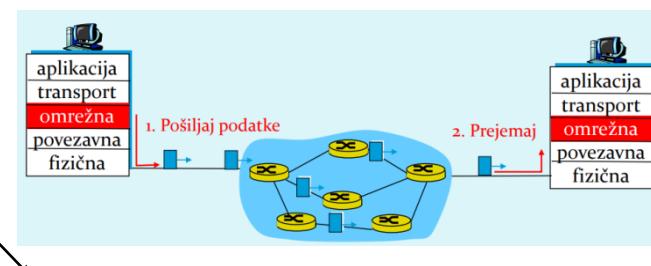
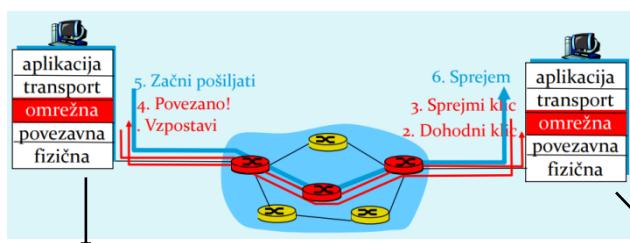
omogočajo **posredovanje paketov** skozi infrastrukturo brez vzpostavljene povezave

NAVIDEZNI VODI

- kot **telefonske zvez**
- faze: **vzpostavitev, tok podatkov, rušenje**
- številke vodov **neodvisne** kar pomeni lažjo konfiguracijo
- usmerjevalniki** usmerjajo pakete glede na **številke** vodov
- internet tega ne uporablja !
- uporaba: **ATM, X.25, MPLS, Frame Relay**

DATAGRAMSKA OMREŽJA

- ni faze vzpostavljanja povezave
- usmerjevalniki **ne hranijo** podatkov o **končnih** povezavah
- paket se **doda naslov cilja** in se ga "vrže" v omrežje
- usmerjevalniki posredujejo glede na **ciljni naslov** v paketu
- paket lahko potuje po **različnih** poteh
- imamo **posredovalne** tabele: 32 bitni naslovi
- tabele bi bile ogromne zato združujemo dele naslovov v **range** ali pa posredujemo na podlagi **predpome**
- zapis z **predponami** je krajši in bolj učinkovit
- če jih ustreza več uporabimo ujemanje **najdaljše**
- usmerjevalniki hranijo tabele in **stanje o povezavah**



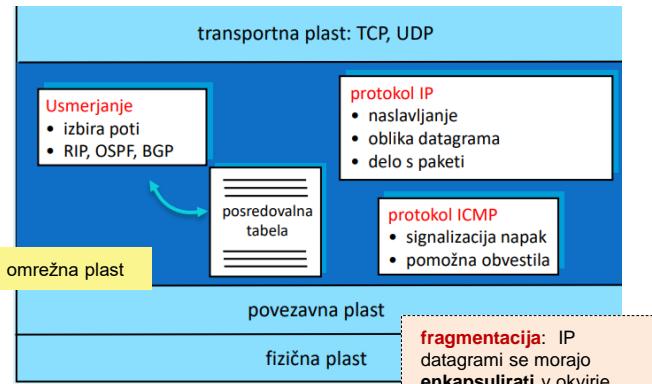
Vhodni vmesnik	Vhodna št. voda	Izhodni vmesnik	Izhodna št. voda
1	12	3	22
2	63	1	18
3	7	2	17
...

Vhodni vmesnik	Vhodna št. voda	Izhodni vmesnik	Izhodna št. voda
1	22	3	32
1	34	2	23
2	4	1	55
...

Ciljni naslov	Vmesnik povezave	ujemanje po predponi	Vmesnik povezave
Od 11001000 00010111 00010000 00000000	0	Ciljni naslov	
Do 11001000 00010111 00010111 11111111		11001000 00010111 00010	0
Od 11001000 00010111 00011000 00000000	1	11001000 00010111 00011000	1
Do 11001000 00010111 00011000 11111111		11001000 00010111 00011	2
Od 11001000 00010111 00011001 00000000	2	sicer	
Do 11001000 00010111 00011111 11111111			3
sicer	3	zdrževanje v range	

Internet (datagramsko)	ATM (VC omrežje)																
usmerjanje glede na ciljni naslov	usmerjanje glede na ID voda																
komunikacija med računalniki : zato so dovoljene elastične storitve, kjer čas ni tako pomemben	izvira iz telefonije : zakasnitev in zanesljivost sta pomembna																
težka zagotovila kakovosti	preprosta zagotovila kakovosti																
<ul style="list-style-type: none"> končni sistemi so "pametni", znajo sami popravljati napake in izvajati manjkajoče storitve omrežje je preprosto 	<ul style="list-style-type: none"> končni sistemi so "neumni" omrežje je kompleksno, saj mora zagotavljati storitve kakovosti 																
preprosto dodajanje novih storitev (aplikacij) in povezovanje heterogenih omrežij	težje dodajanje novih storitev, pogojeno z infrastrukturno omrežja																
32 bitov																	
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td>ver</td> <td>header length</td> <td>type of service</td> <td colspan="2">length</td> </tr> <tr> <td></td> <td></td> <td></td> <td>identifier</td> <td>flgs</td> <td>fragment offset</td> </tr> <tr> <td>time to live</td> <td>upper layer</td> <td colspan="4">Internet checksum</td> </tr> </table>	ver	header length	type of service	length					identifier	flgs	fragment offset	time to live	upper layer	Internet checksum			
ver	header length	type of service	length														
			identifier	flgs	fragment offset												
time to live	upper layer	Internet checksum															
IP naslov izvora																	
IP naslov cilja																	
opcije																	
podatki - spremenljiva dolžina (ponavadi TCP ali UDP segment)																	

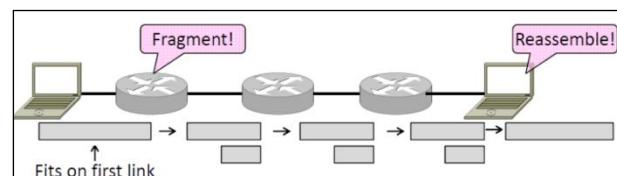
INTERNET PROTOCOL : IP



primerjava obeh

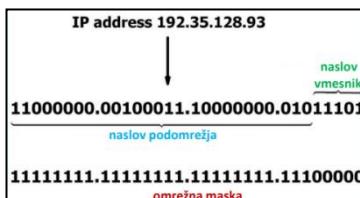
KOMPONENTE:

- **VER**: verzija IP protokola – **4 biti**
- **HEADER LENGTH**: poda kje se **začnejo** podatki – **4 biti**
- **TYPE OF SERVICE**: razlikovanje datagramov z **posebno obravnavo** – **8 bitov**
- **LENGTH**: skupna dolžina celega datagrama in bytes – **16 bitov**
- **ID FLAG in OFFSET**: potrebno za IP **fragmentacijo** – **32 bitov**
- **TTL**: za **preprečitev ciklanja** datagramov vsak usmerjevalnik zmanjša vrednost za 1 – **8 bitov**
- **UPPER LAYER**: številka **enkapsuliranega protokola** v podatkih – **8 bitov**
- **CHECKSUM**: kontrolna enota glave datagrama preračuna jo vsak usmerjevalnik – **16 bitov**
- **IP NASLOVI**: naslovi **izvora** in **cilja** – **32 bitov**
- **OPCIJE**: za možne **razširitve** glave datagrama – **32 bitov**
- **PODATKI**: poljubne dolžine



IPV4 NASLAVLANJE:

- dolžina **32 bitov**
- računalniki imajo običajno **en vmesnik**
- usmerjevalniki **več vmesnikov**
- cca. 4 milijarde IP naslovov
- naslovi morajo biti **globalno unikatni**

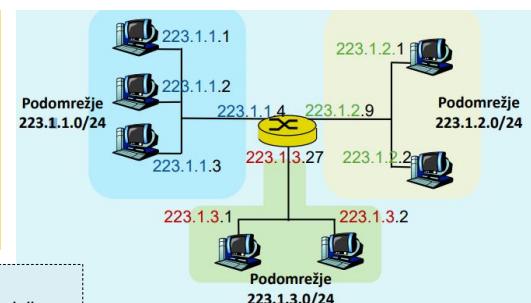


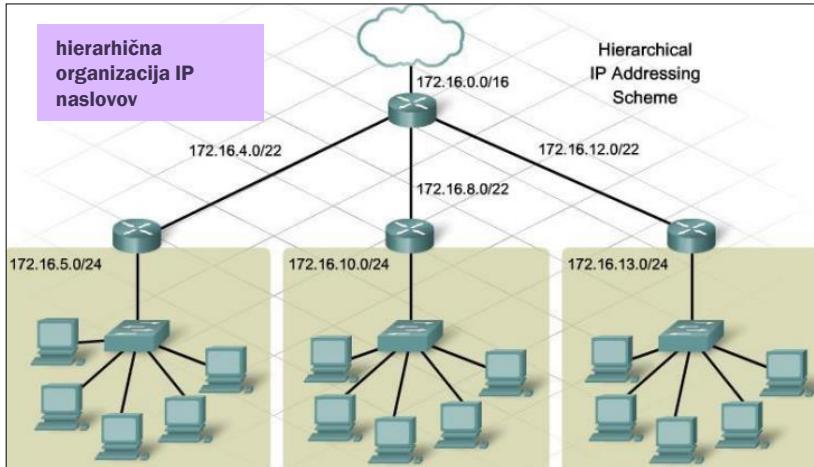
PODOMREŽJE:

- **predpona** predstavlja naslov podomrežja
- **2 dela**: naslov **omrežja** in naslov **naprave**
- podomrežje ima **enak naslov omrežja**
- **maska** podomrežja: določa dolžino naslova podomrežja in jo okrajšamo z številom enic
- usmerjevalnik ima na vsakem vmesniku **drugo podomrežje**
- najprej smo imeli **razrede A** 8 bitov in **B** 16 bitov in **C** 24 bitov za maske
- potem prefiksna oz **CIDR notacija**
- poseben **broadcast** naslov: naslov **naprave** so same ene

Kako določimo IP naslove?

administrator naprave vpše **fiksen naslov** ali pa **DHCP strežnik** dodeli **dinamičnega**, nato ponudnik dostopa do interneta **ISP** dodeli del s svojega naslovnega prostora ki mu ga dodeli **ICANN**





NAT: Network Address Translation

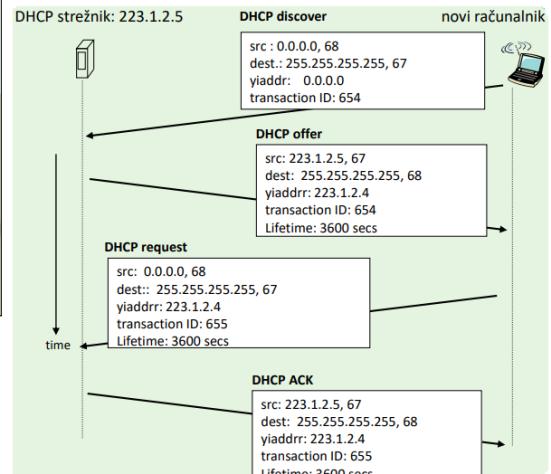
- zaradi pomanjkanja IPv4 naslovnega prostora
- lokalni naslovi ki so lahko ponovljivi ker niso v javnem internetu

Naslovi	Omrežje/maska	Št. naslovov
10.0.0.0 - 10.255.255.255	10.0.0.0/8	2^{24}
172.16.0.0 - 172.31.255.255	172.16.0.0/12	2^{20}
192.168.0.0 - 192.168.255.255	192.168.0.0/16	2^{16}

→ usmerjevalnik z NAT lokalni naslov preslika v globalni

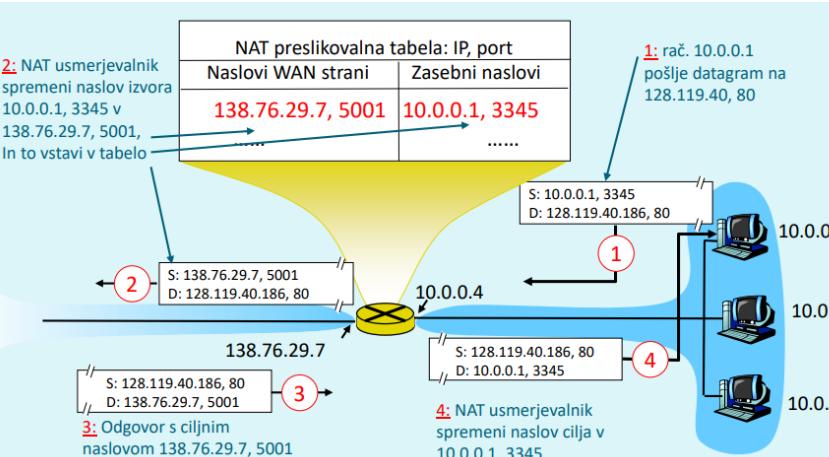
dinamično dodeljevanje DHCP:

- ob priklopu naprava nima naslova IP
- potrebna dodelitev osnovnih omrežnih nastavitev
- dodeli naslov v 4 fazah:
DISCOVER OFFER REQUEST ACK



PREDNOSTI

- zadošča samo en javni naslov za dostop celega omrežja do Interneta
- spremiščamo neodvisno od zunanjega naslova
- večja varnost notranjih naprav
- 16 bitno polje za vrata evidentira cca. 60 000 povezav do naprav



SLABOSTI

- usmerjevalniki naj bi delali na 3. plasti
- vrata oz. port so del 4. plasti
- vrata so naslavljajo procese ne računalnikov
- krši princip končnih sistemov: zahteva da je omrežje transparentno problem pri P2P
- bolje uporabiti IPv6

ICMP

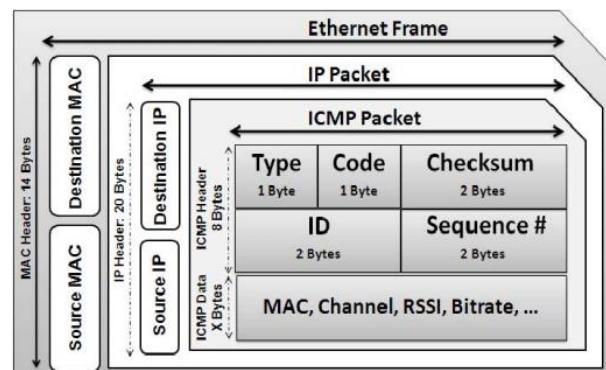
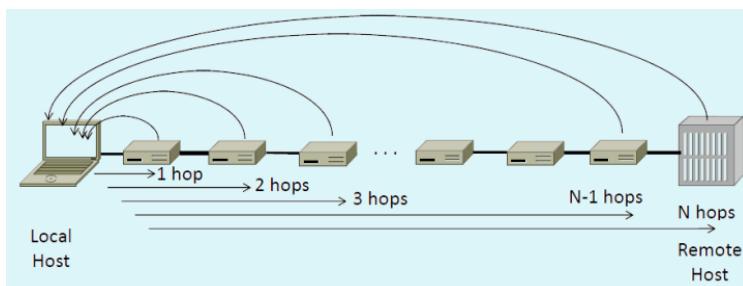
- sporočilo enkapsulirano **znotraj paketa IP**
- Internet Control Message Protocol
- traceroute: analizira po kateri poti gre promet do cilja
- koračni protokol preizkuša komunikacijo na posameznih korakih

delovanje aplikacije Traceroute: izvor pošilja serijo paketov cilju prvi ima TTL 1 drugi TTL 2 itd.

usmerjevalnik pogleda vrednost TTL če ni 0 potem zmanjša vrednost za 1 in če je 0 paket zavrne in napiše TTL expired

za vsako prejetjo ICMP sporočilo izvor izračuna tudi čas vrnitve in statistike

Tip	Koda	Pomen
0	0	echo reply (ping)
3	0	dest network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header



INTERNET PROTOCOL: IPv6

zakaj ga potrebujemo?

- potrebujemo **večji naslovni prostor**
- **hitrejše** usmerjanje: fragmentacija ni dovoljena
- zagotavljanje **kakovosti storitev** za posebne tokove podatkov: **QoS**

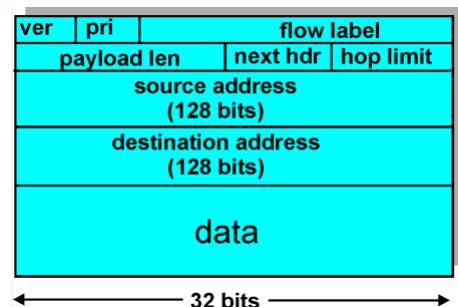


hitrejše usmerjanje:

- fragmentacije ni več pakete delita **pošiljalitelj** in **sprejemnik**
- usmerjevalnik lahko **zavrže** paket in sporoči pošiljalju "Packet Too Big"
- zato nova verzija protokola ICMP: **ICMPv6**
- **glava** nima **kontrolne vsote** ta je v enkapsuliranih protokolih znotraj IP paketa
- **glava** nima polja za **opcije**, implementacija kot poseben enkapsuliran protokol

SINTAKSA

- 128 bitov razdeljenih v 8 blokov po 16 bitov
- zapišemo **šesnajstiško**
- krajšamo **vodilne** niče
- najdaljše zaporedje ničel krajšamo z ::
- to lahko storimo samo **enkrat**: vedno tisto zaporedje ki je **najdaljše**, če sta tako dva okrajšamo **prvega**
- IPv4 naslovom spredaj dodamo ničle
- IPv4 naslovi v **dva bloka** po 16 bitov ali pa pustimo pike



IPv4 Header

Version	IHL	Type of Service	Total Length	
Identification		Flags	Fragment Offset	
Time to Live	Protocol	Header Checksum		
Source Address				
Destination Address				
Options		Padding		

Legend

- Field's Name Kept from IPv4 to IPv6
- Fields Not Kept in IPv6
- Name and Position Changed in IPv6
- New Field in IPv6

IPv6 Header

Version	Traffic Class	Flow Label
Payload Length		Next Header
Source Address		
Destination Address		
Padding		

KOMPONENTE:

- VER: **verzija** IP protokola – 4 biti
- PRI oz. TRAFFIC CLASS: podobno kot **type of service** – 8 bitov
- FLOW LABEL: oznaka **toka** podatkov ki zagotavlja **kakovost** storitve – 20 bitov
- PAYLOAD LENGTH: velikost podatkov ki sledijo glavi – 16 bitov
- NEXT HDR : tip **enkapsuliranega** protokola – 8 bitov
- HOP LIMIT: enako kot **TTL** – 8 bitov

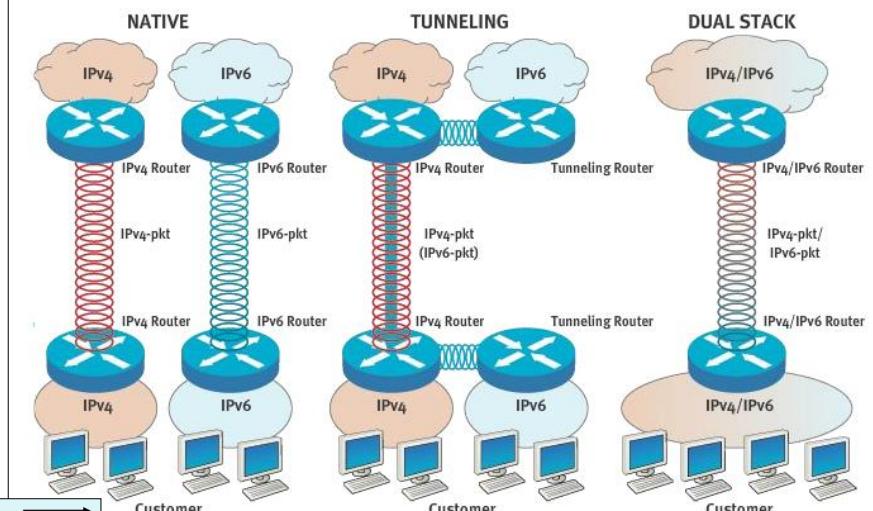
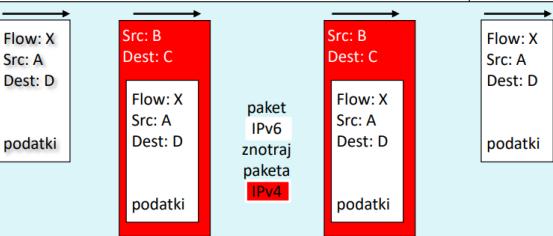
Prehod iz IPv4 na IPv6

- vseh naprav ni mogoče nadgraditi naenkrat
- rešitev: **dvojni sklad**
- rešitev: **tunneling** kjer IPv6 v IPv4 kot data

DUAL STACK:

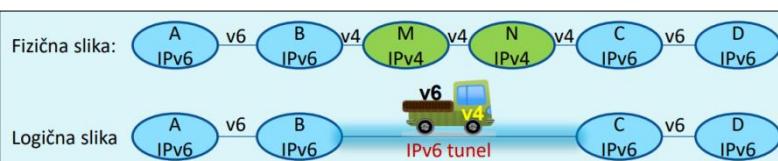
- usmerjevalnik z vozlišči ki uporabljajo **IPv6** komunicira s tem protokolom
- z ostalimi z **IPv4**
- **DNS strežnik** vrne IPv6 ali IPv6 naslov
- če na poti nek **IPv4 naslov** se bo promet vmes pretvarjal v **IPv4**
- zgubijo se **specifična polja**

TUNNELING:



Varnost na omrežni plasti: IPSec

- komunikacija poteka **nevarovano**
- možna so **ponarejanja** izvornih naslovov, **prisluškovanje** in **ponarejanje** komunikacije
- **IPSec** skrbí za varno komunikacijo



STORITVE

- dogovor o uporabljenih **criptografskih algoritmih** in **ključih**
- **enkripcija** in **dekripcija**
- **integriteta** podatkov
- **avtentifikacija** izvora

USMERJANJE

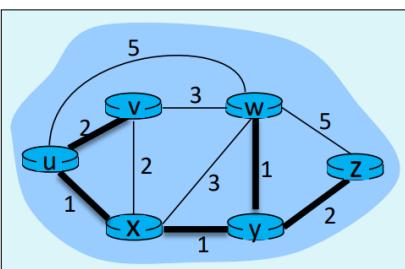
- po kateri poti naj paket potuje od **izvora** do **cilja**?
- omrežja modeliramo z **teorijo grafov**
- imamo **usmerjevalne protokole** ki konfigurirajo **posredovalne tabele**
- vzpostavijo **najcenejšo pot**
- **cena**: učenje povezave lahko različna čas, razdalja...

VRSTE USMERJEVALNIH ALGORITMOV

- **centralizirani**: dostopne podatke o stanju povezav v **celem** omrežju
- **decentralizirani**: podatke samo o **neposredno** priključenih povezavah
- **prilagodljivi**: avtomatsko prilagajajo cene glede na zasičenost povezav
- **neprilagodljivi**

centralizirani:

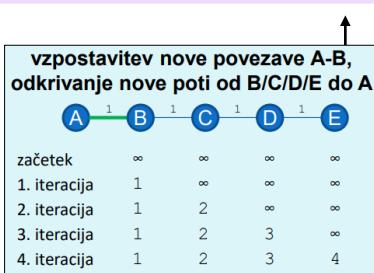
- uporaba **centralnega vozlišča**
- **vozlišče** koordinira **usmerjanje** ali neodvisno izračunavanje posameznih vozlišč
- **link state** algoritmi
- vsako vozlišče zase izračuna **drevo najkrajših poti**
- **Dijkstra algoritem**: rezultat posredovalna tabela



prilaganje tabele na spremembe:

good news travel fast: podatek o znižanju cene povezav se hitro razširi

bad news travel slow: podatek o povišanju cene povezav se širi počasi, kar lahko privede do "štetja do neskončnosti"



hierarhično usmerjanje:

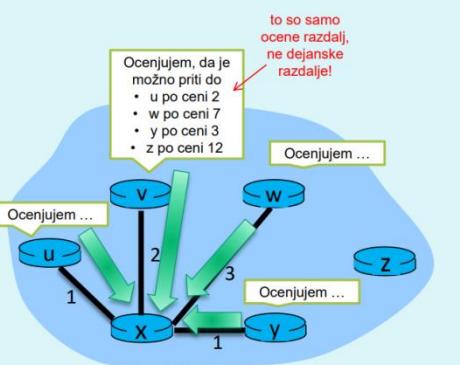
problem: imeli bi **velike** usmerjevalne tabele in **administrator** vsakega usmerjevalnega omrežja želi administrirati omrežje po svoje

decentralizirani:

- usmerjanje z **vektorjem razdalj**
- posredovalna tabela iz **lokalnih podatkov**
- usmerjanje je **iterativno**
- tabelo izračunamo po **korakih**

najcenejša pot: imamo **vozlišče s** z sosedi, potrebujemo **ceno povezave** od izvora **x** do **sosedov** in **oceno** cene najcenejše poti od soseda **s** do **cilja y**

porazdeljeno usmerjanje:



usmerjanje z vektorjem razdalj:

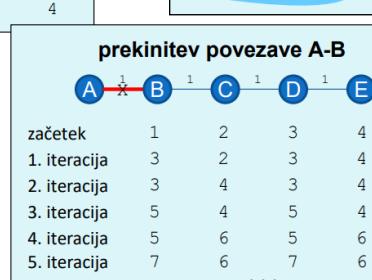
vsako vozlišče hrani svoj **vektor razdalj**, vektorje razdalj vseh **svojih sosedov** in posredovalno **tabelo**

vozlišče prejme vektor razdalj **od sosedja**, izračuna svoj novi vektor razdalj

$$D(x, y) \leftarrow \min_{s \in S} \{c(x, s) + D(s, y)\} \forall y$$

vsako vozlišče **pošlje** svoj vektor **sosedom** kadar zazna **spremembo** v svojem vektorju razdalj

sčasoma vse posredovalne tabele vsebujejo dejanske **najmanjše cene**



$$D_x(y) = \min\{c(x, y) + D(y, y), c(x, z) + D(z, y)\} = \min\{2+0, 7+1\} = 2$$

$$D_x(z) = \min\{c(x, y) + D(y, z), c(x, z) + D(z, z)\} = \min\{2+1, 7+0\} = 3$$

cena do vozlišče x

x y z

x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

cena do vozlišče y

x y z

x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

cena do vozlišče z

x y z

x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

cena do vozlišče x

x y z

x	0	2	7
y	2	0	1
z	3	1	0

cena do vozlišče y

x y z

x	0	2	3
y	2	0	1
z	3	1	0

cena do vozlišče z

x y z

x	0	2	3
y	2	0	1
z	3	1	0

cena do vozlišče x

x y z

x	0	2	3
y	2	0	1
z	3	1	0

cena do vozlišče y

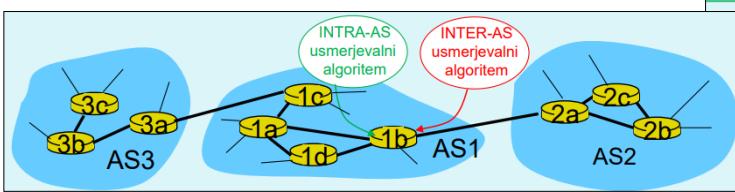
x y z

x	0	2	3
y	2	0	1
z	3	1	0

cena do vozlišče z

x y z

x	0	2	3
y	2	0	1
z	3	1	0



INTRA - A S :
 → zanj skrbijo **IGP** protokoli
 → **RIP**: Routing Information Protocol
 → **IGRP**: Interior Gateway Routing Protocol
 → **OSPF**: Open Shortest Path First

→ usmerjanje z vektorjem razdalj
 → algoritem optimizira število hopov
 → cena vsake povezave je 1
 → največja dovoljena cena je 15
 → vektor razdalj se razpošilja na 30 s
 → povezava prekinjena po 180 s

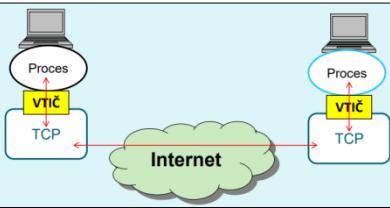
INTER - A S :
 omogoča omrežjem oglaševanje svoje prisotnosti drugim omrežjem

pošiljalci: poročilo razbije v **segmente** in jih posreduje v **enapsulacijo** omrežni plasti

→ usmerjanje glede na stanje povezav
 → obvestila se s **poplavljanjem** posredujejo celotnemu sistemu
 → preračuna **najkrajše** poti
 → varnost, usmerjanje po več poteh, razpošiljanje, hierarhično usmerjanje

razlika med plastema:
 → **omrežna plast:** logična povezava med končnimi sistemmi
 → **transportna plast:** logična povezava med procesi

kako komuniciramo s procesom / aplikacijo ?
 → vsak proces ima **vstopno** točko
 → vstopno točko imenujemo **vtič**
 → **vtič:** vmesnik med **aplikacijsko** in **transportno** plastjo
 → če teče **več** procesov ima vsak od njih svoj vtič
 → preko vtiča proces **sprejema** in **oddaja** sporočila



→ **Ciscova** izboljšava protokola **RIP**
 → usmerjanje z vektorjem razdalj
 → cena se izračuna kot utezena **vsota pasovne širine, zakasnitve, obremenitev, MTU in zanesljivosti**

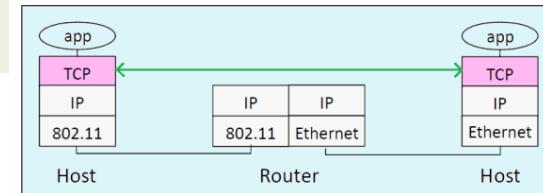
prejemnik: dekapsulira **segmente** iz paketov sestavljenih segmente **zdrži** v sporočila in jih posreduje **aplikacijski** plasti

kako nasloviti proces na drugi strani ?
 → naslov **vmesnika** naprave: IP številka
 → naslov **procesa:** številka vrat

znane številke vrat 0-1023
 • spletni strežnik **HTTP : 80**
 • poštni strežnik **SMTP : 25**
 • imenski strežnik **DNS : 53**
 • oddaljen dostop oz. **telnet : 23**
 • pogovorni strežnik **IRC : 194**

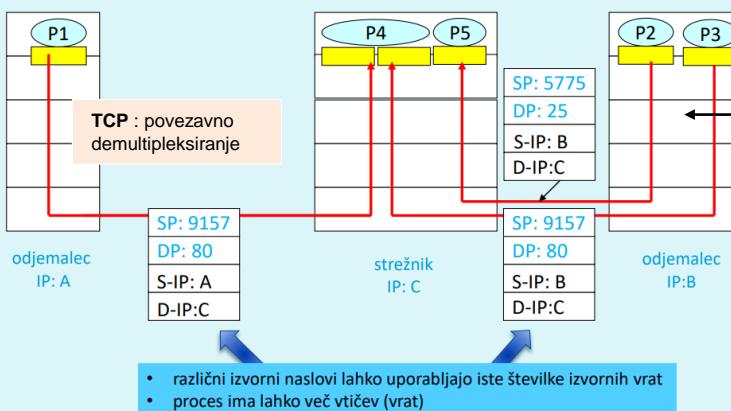
TRANSPORTNA PLAST

tasks :
 • povezovanje dveh oddaljenih procesov
 • **multipleksiranje / demultipleksiranje** komunikacije
 • zanesljiv **prenos** podatkov
 • kontrola pretoka in zasičenja
 • logična komunikacija med **aplikacijskimi** procesi
 • protokola **TCP** in **UDP**



kako poteka demultipleksiranje ?

→ vsak transportni **segment** potuje znotraj svojega **paketa IP**
 → transportni segment ima **16 bitna** podatka
 → številka vrat **izvora in ponora**
 → **pošiljalci:** pobira podatke z več vtičev
 → **prejemnik:** segmente razdeli ustreznim vtičem



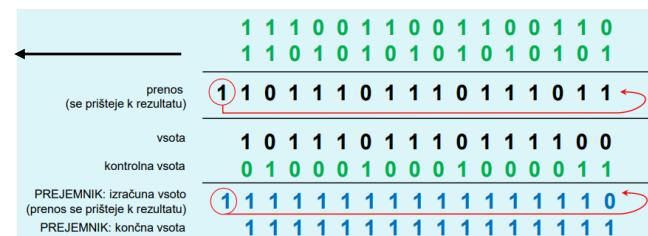
nepovezavni transport : UDP

UDP datagram:

- namenjen uporabi v **okoljih** kjer lahko toleriramo **izgube**
- pomembna **hitrost pošiljanja**
- multimedija, DNS, SNMP, usmerjevalni protokoli
- **zanesljivost** moramo zagotoviti na **aplikacijski** plasti

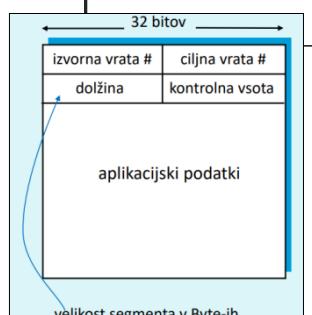
internetna kontrolna vsota:

- algoritem za izračun imenujemo **internetna kontrolna vsota**
- **pošiljalci** sešteje 16 bitne besede in shrani eniški komplement ki mu rečemo **kontrolna vsota**
- **prejemnik** sešteje 16 bitne besede skupaj s kontrolno vsoto
- dobiti mora same **enice**



zakaj ? ni zagotovila, da protokoli na drugih plasteh zagotavljajo **zaznavanje** in **odpravljanje** napak

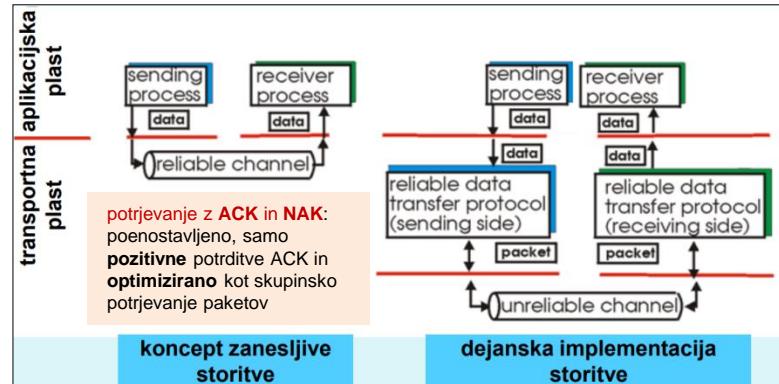
do napak lahko pride tudi pri **hranjenju segmenta v spominu** usmerjevalnika in ne nujno pri prenosu



UDP kontrolna vsota je namenjena preverjanju **pravilnosti** med **izvornim** in **ciljnim** procesom, ne pa pri potovanju po **posameznih povezavah**

PROTOKOL TCP :

- zagotavlja **zanesljivo** dostavo z uporabo **nezanesljivega** kanala
- podatki se ne **okvarijo**
- podatki se ne **izgubljojo**
- podatki so dostavljeni v pravilnem **zaporedu**
- **osnovna funkcionalnost:** pošiljanje in prejemanje
- reševanje **napak** pri prenosu
- obravnavo **izgubljenih** paketov



1. enostavni protokol

- **ideja:** enostavni protokol nadgradimo v zanesljivega predstavimo s **končnim avtomatom**

2. reševanje napak pri prenosu

- vsebina se lahko **okvari** to preverjamo z **kontrolno vsoto**
- dodamo **funkcionalnosti:** zaznavanje napak
- **ACK:** paket pravilno sprejet
- **NAK:** paket nepravilno sprejet

3. izguba paketov ali potrditev

- paketi se lahko **izgubijo** in ne pridejo do prejemnika
- pošiljalatelj počaka določen **interval časa** na **ACK** če ga ne dobi pošlje paket **ponovno**
- možna tudi **izguba potrditve**
- možnost **prekratkega časovnega intervala**
- težavo rešimo s **številčenjem** paketov in zaznavanjem istih številk

4. posredno potrjevanje

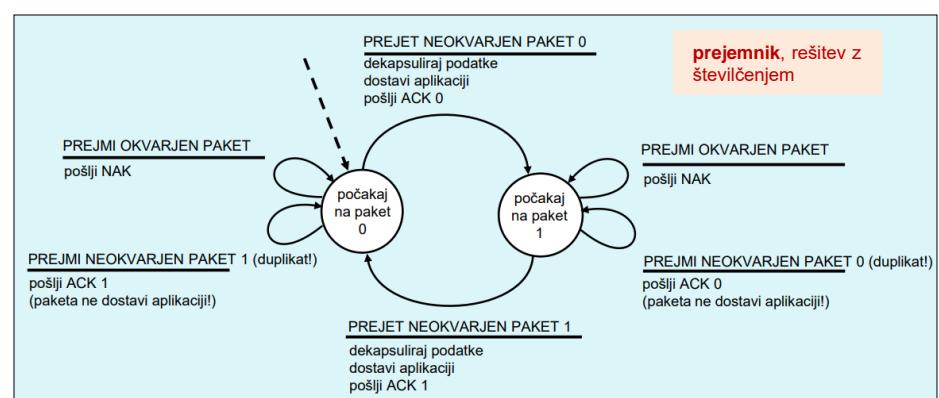
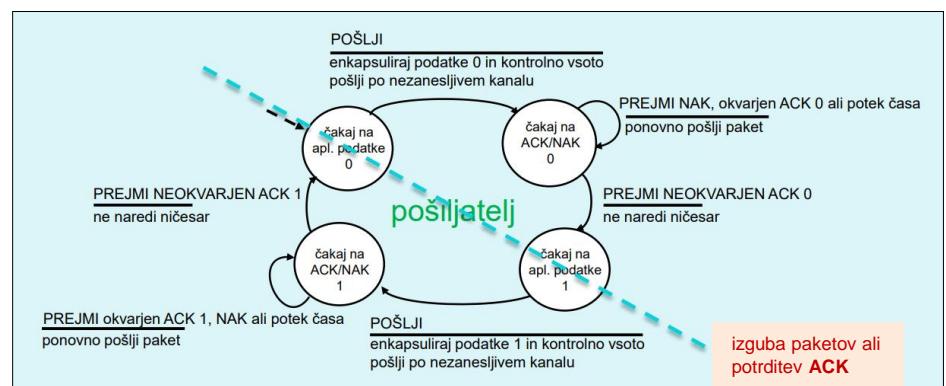
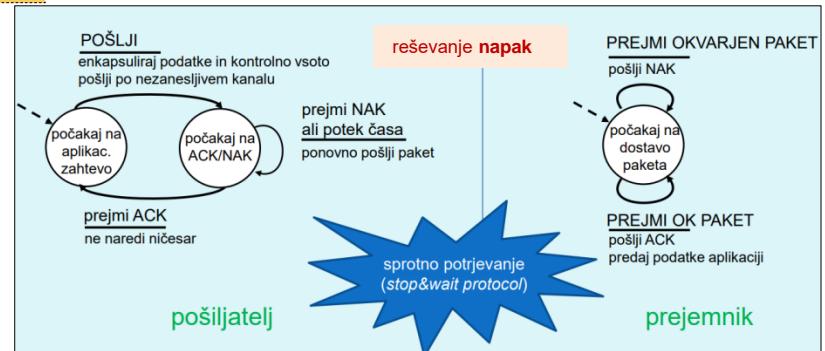
- posredno potrjevanje samo **ACK**
- uporabimo **ACK** v katerega vključimo **številko** segmenta ki ga **poterujemo**
- **neposredno** potrjevanje: **ACK** in **NAK**
- **posredno** potrjevanje: samo **ACK**
- prejemnik odgovori z **ACK** za zadnji še **uspešno prejeti segment**
- če pošiljalatelj prejme **ACK** za isti segment **dvakrat** segment ki je sledil ni bil sprejet pravilno

5. odprava neučinkovitosti

- to je **stop and wait** protokol
- pošiljalatelj mora pred oddajo naslednjega paketa **čakati** na potrditev prvega paketa
- implementacija **vzporednega** oz. **cevovodnega** oz. **tekočega** pošiljanja večja **izkoriščenost** kanala
- potrebujemo večji **razpon** števil za številčenje paketov
- **pomnilnik** za shranjevanje paketov

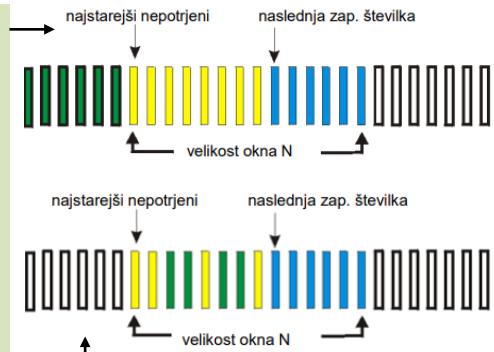
1. ponavljanje
N nepotrjenih

2. ponavljanje
izbranih



ponavljanje N nepotrjenih paketov:

- pošiljalatelj hrani **okno** največ dovoljenih **nepotrjenih** paketov
- to je **protokol z drsečim oknom**
- ko prejemnik pošlje **ACK (n)** s tem potrdi vse pakete do vključno z n
- časovna kontrola za **najstarejši paket**
- ko časovna kontrola **poteče** pošljemo vse nepotrjene pakete v oknu **ponovno**
- tako lahko dobimo **podvojene** pakete
- pakete prepoznamo po **zaporedni številki** in jih **zavrhemo**
- prejemnik lahko prejme pakete v napačnem **vrstnem redu**
- take pakete spet **zavrhemo**



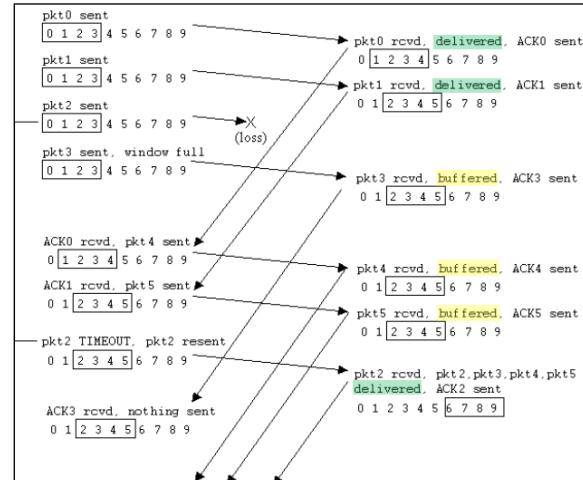
ŽE POTRJENI
POSLANI, ampak še NEPOTRJENI

kandidati za pošiljanje
pošiljanje ni dovoljeno

ponavljanje izbranih paketov:

- vsak prejet paket potrjen posamezno
- prejemnik **shranjuje** pakete v **napačnem vrstnem redu** in jih sortira preden jih pošlje aplikaciji
- pošiljalatelj ponovno pošlje samo pakete od katerih ni dobil **ACK**
- pošiljalatelj hrani **štoparico** posebej za vsak nepotrjen paket

možne so vse oblike potrjevanja



TCP SEGMENT

URG: urgentni podatki (označi pošiljalitelj)

ACK: zastavica, ki pove, da gre za (potrditev)

dolžina glave v 32-bit besedah

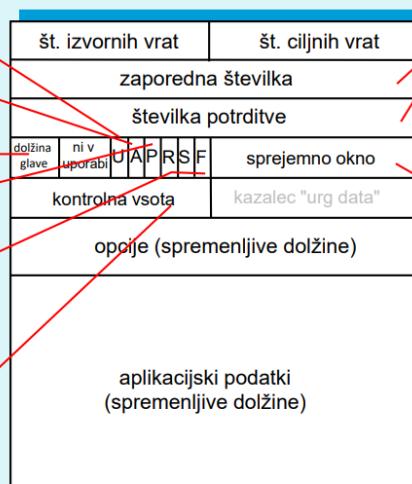
PSH: takoj predaj aplikaciji

RST, SYN, FIN: za vzpostavitev in rušenje povezave

Internetna kontrolna vsota

SEQ: zaporedna številka je številka prvega bajta v segmentu

32 bitov



ACK: številka naslednjega pričakovanega bajta

štejeta število bajtov podatkov (ne število segmentov!)

za kontrolo pretoka: število bajtov, ki jih prejemnik lahko sprejme

	SPRTOVNO	TEKOČE pošiljanje
NEPOSREDNO	✓	✓
POSREDNO	✓	✓

ŠTEVILČENJE SEGMENTOV:

- **client in server** vzpostavita zvezo
- povezava je nato **dvosmerna**
- pošiljalatelj lahko v enem segmentu **istočasno pošlje nove podatke** in **potrditev** prejšnjega segmenta

LASTNOSTI TCP:

- med dvema točkama **point to point**
- **povezavni protokol:** vzpostavitev in rušenje zveze
- **dvosmerni** promet pri povezavi
- zanesljiv **urejen** tok podatkov
- **kontrola pretoka:** pošiljalatelj ne preobremeniti prejemnika
- **kontrola zasičenja:** pošiljalatelj ne preobremeniti omrežja
- uporablja **tekoče pošiljanje**
- velikost okna se določa **avtomatsko** glede na kontroli



uporabnik pošlje 6 Byte podatkov ("lalala")



B potrdi sprejem podatkov in **pošlje** svoje podatke **skupaj s potrditvijo** prejetega (pošiljanje na "štuporamo", *piggy-backing*)

A potrdi sprejem "hopsasa"

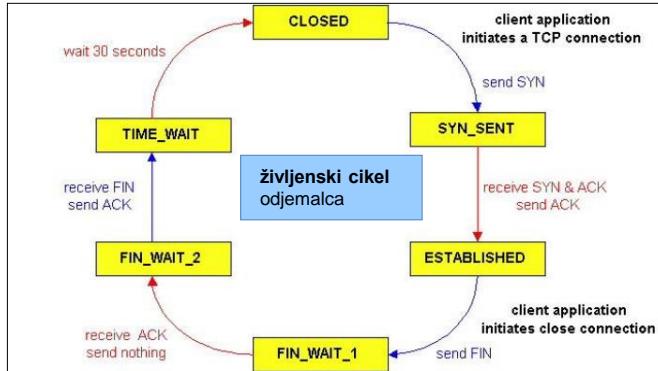
Seq=42, ACK=79, data = 'lalala'
Seq=79, ACK=48, data = 'hopsasa'
Seq=48, ACK=86

VZPOSTAVLJANJE POVEZAVE:

- prejemnik in pošiljalatelj izvedeta **handshake**
- izmenjata naključno določene **začetne pričakovane zaporedne številke**
- izmenjata **velikosti medpomnilnikov** za kontrolo pretoka

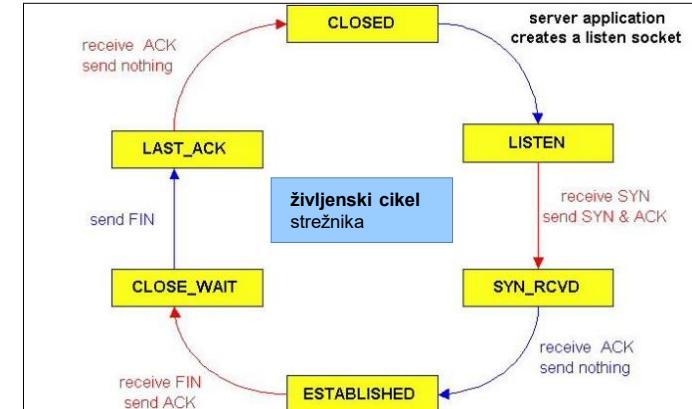
three way handshake

- odjemalec pošlje segment z **SYN** flag to je začetna številka
- pri **SYN** ni **podatkov**
- strežnik vrne segment **SYN ACK** z začetno številko
- rezervira **medpomnilnik**
- odjemalec vrne **ACK** ki ima lahko že **podatke**



RUŠENJE POVEZAVE:

- odjemalec pošlje TCP FIN strežniku
- strežnik potrdi z ACK
- zapre povezavo in pošlje FIN
- počaka časovni interval
- po potrebi ponovno pošlje ACK
- strežnik sprejme ACK



napad SYN FLOOD: napadalec pošlje strežniku veliko paketov za vzpostavitev zveze TCP SYN zato strežnik vsakič rezervira del pomnilnika

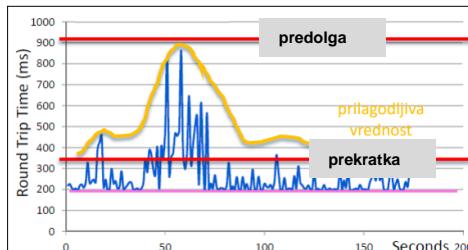
pomnilnik ostane zaseden zaradi napol napol odprtih zvez, ker napadalec ne zaključi rokovanja. Strežniku zmanjka prostora in pride do odpovedi sistema

način potrjevanja ?

- kombinacija obeh načinov
- podoben ponavljanju N nepotrjenih
- štoparica za najstarejši nepotreni segment
- poteku časovne kontrole ne pošlje vseh segmentov v oknu ampak samo najstarejšega

nastavitev časovne kontrole:

- štoparica je potrebna za zanesljivo dostavo paketov
- interval mora biti daljši od časa vrnitve
- temu rečemo RTT oziroma round trip time
- to je pot od pošiljalnika do prejemnika in nazaj
- prekratek: preveč ponovnega pošiljanja
- predolg: predolgo čakamo na reakcijo
- avtomatsko opravimo RTT meritve
- potrebujemo prilagodljivo vrednost časovne kontrole saj je RTT lahko nestabilen zaradi različnih poti in preobremenjenosti usmerjevalnikov



HITRO PONOVNO POŠILJANJE:

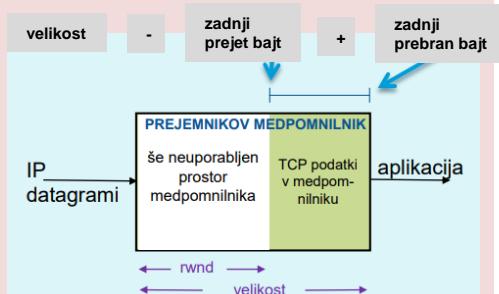
- po navadi ponovno pošiljanje izvedemo po preteklu časovne kontrole
- včasih je časovni interval predolg
- lahko skrajšamo če prejme za nek paket 3 podvojene potrditve

Dogodek pri prejemniku	Odziv prejemnika
Sprejem segmenta s pričakovano številko, vsi prejšnji že potrjeni.	Počakaj na naslednji segment max 500 ms. Če ta pride v tem intervalu, izvedi zakasnjeni potrditev obeh (delayed ACK). Če ne pride v tem intervalu, potrdi samo prejetega.
Isto kot zgoraj, a potrditev za prejšnji segment še ni bila poslana.	Takoj pošlji kumulativno potrditev za oba segmenta brez izvajanja zakasnjenje potrditve.
Sprejem segmenta s previsoko številko (zaznamo vrzel)	Takoj potrdi zadnji še sprejeti segment (pošlji duplikat ACK).
Sprejem segmenta z najnižjo številko iz vrzeli (polnjenje vrzeli)	Takoj potrdi segment.

posebnosti pri potrjevanju TCP

KONTROLA PRETKA:

- ne smemo pošiljati hitreje kot prejemnik bere da ni prekoračitve medpomnilnika
- rwnd: neuporabljen prostor



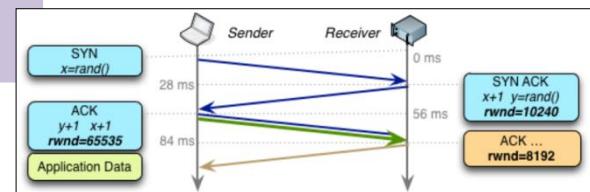
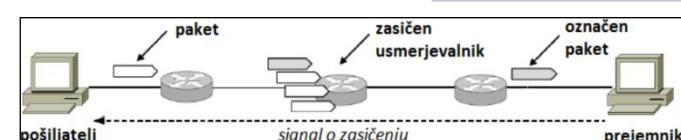
- prejemnik sporoča **velikost** razpoložljivega prostora v **glavi** vsakega segmenta
- pošiljalatelj ustrezno **omeji** število paketov za katere še ni prejel potrditve
- **rwnd** oziroma **receive window**

KONTROLA ZASIČENJA:

- **zasičenje**: stanje omrežja ko veliko virov skupaj prehitro pošilja preveč podatkov
- **izgubimo** segmente zaradi **prekoračitve medpomnilnika** v usmerjevalnikih
- velike **zakasnitve** zaradi čakalne vrste v usmerjevalnikih
- nadzorujemo na 2 načina

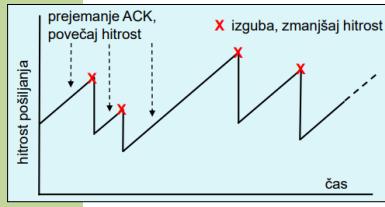
z uporabo omrežnih storitev

- usmerjevalniki obvestijo pošiljalnika
- obvestilo o zasičenju ECN
- usmerjevalnik nastavi ustrezni bit in sporoči sprejemljivo hitrost oddajanja



TCP KONTROLA ZASIČENJA

- vsak pošiljaljelj si sproti nastavlja hitrost
- na podlagi **opazovanja reakcij** v omrežju
- če prejme potrditev ni zasičenja in **poveča** hitrost
- če se segment izgubi je zaradi zasičenja **zmanjša** hitrost
- ima **žagasto obliko** hitrosti pošiljanja
- uporabljamo cwnd ali congestion window
- TCP pošilja s hitrostjo **minimalno** od cwnd in rwnd
- cwnd lahko povečujemo eksponentno ali linearno
- če se paket izgubi zmanjšamo cwnd na 1



SLOW START

- ob **vzpostaviti** povezave je cwnd na 1
- hitrost povečujemo **eksponentno**
- za vsak ACK torej povečamo za 1
- ob vseh prejetih ACK torej dvakratno
- ko pride do prve izgube se ustavi
- zapomni si **prag** ki je polovica cwnd preden pride do zasičenja
- cwnd se nato nastavi na 1
- spet izvajamo **slow start** do praga !

zgodovina: TCP Tahoe je osnovna verzija ima samo **počasni začetek** in **izogibanje zasičenju**

TCP Reno ima 4 kolesa in havbo. ne lol sorry dobri fazo **hitre obnove**

TCP Vegas ima dodano raznavanje situacij ki vodijo v zasičenje in **linearno zmanjšuje hitrost ob zasičenju**

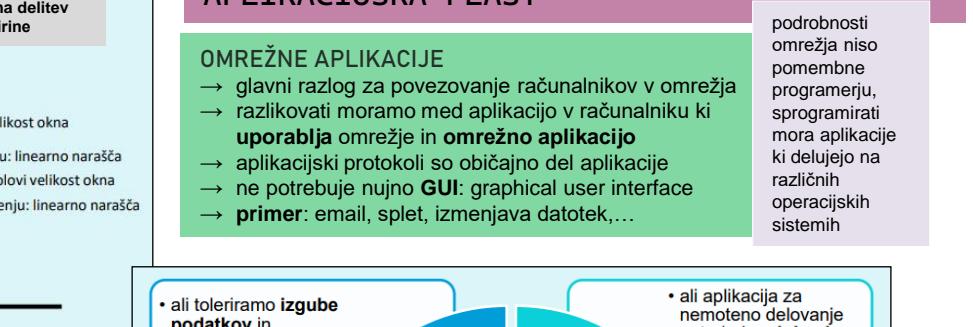
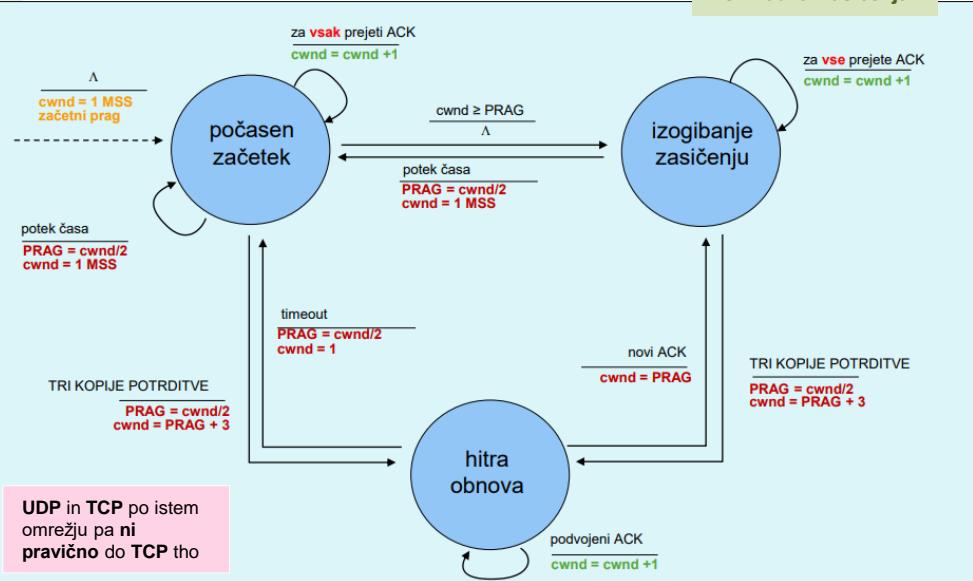
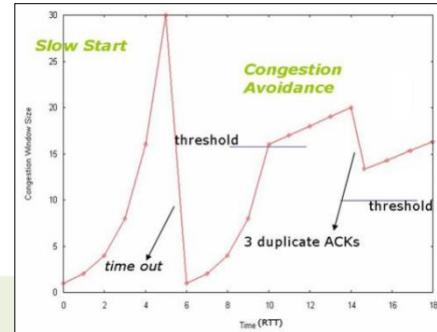
pravičnost: vsaka od N TCP sej po isti povezavi s **kapaciteto C** naj bi dobila delež prenosa **C / N**

več TCP pošiljaljelj v praksi **pravično** deli pasovno širino
mehanizem nadzora zasičenja konvergira v **sredinsko točko** grafa

IZOGIBANJE ZASIČENJU

- ko čez prag začnemo povečati **linearno**
- tako se pragu približujemo **bolj počasi**
- če poteče **časovna kontrola** gre cwnd na 1 in od začetka
- če dobimo **trikrat** podvojeni ACK na faze **hitre obnove**
- nastavimo prag in cwnd gre na **prag plus 3**

končni avtomat za TCP nadzor zasičenja



arhitekture aplikacij:

- **strežnik in odjemalec** : strežnik **ves čas** dostopen odjemalcu priključen le **občasno**
- **peer to peer** : končni sistemi **direktno** komunicirajo med seboj, člani lahko nedostopni
- **mešana arhitektura** : na primer Skype

zahteve aplikacij po storitvah transporta

Aplikacija	primeri aplikacijskih protokolov	dovoljena izguba	potrebna minimalna časovna širina	časovna občutljivost	običajen transportni protokol
prenos datotek	FTP	ne	ne (elastična)	ne	TCP
e-pošta	SMTP, POP3, IMAP	ne	ne (elastična)	ne	TCP
splet	HTTP	ne	ne (elastična)	ne	TCP
oddaljen dostop	telnet	da	~10 kb/s	da	TCP
multimedija (realni čas)	RTP, HTTP	da	10 kb/s – 5 Mb/s	da	TCP/UDP
multimedija (shranjena)	RTP, HTTP	da	10 kb/s – 5 Mb/s	da	TCP/UDP
interaktivne igre	HTTP, lastniški	da / ne (?)	~10 kb/s	da	TCP/UDP
IP telefonija	SIP, RTP, lastniški	da	100 kb/s – 3 Mb/s	da	TCP/UDP

izbira transportnega protokola in zanesljivost dostave

HTTP, FTP

zaporedja podatkov spremenljive dolžine, komunikacija po principu zahteva/odgovor

TCP

RTP, Skype

nezanesljiv prenos (tokovi v realnem času) ali kratka sporočila

UDP

kaj narediti, če potrebujemo zanesljivost (DNS, DHCP) ?
UDP

protokol HTTP

metode pri HTTP zahtevi

QUICK HISTORY LESSON :

- pojavi se okoli 1990
- revolucionarna aplikacija
- dostop do vsebin na **zahtevo**
- vsak lahko **oglašuje**
- iskanja, povezave, grafika, vmesniki, multimedija

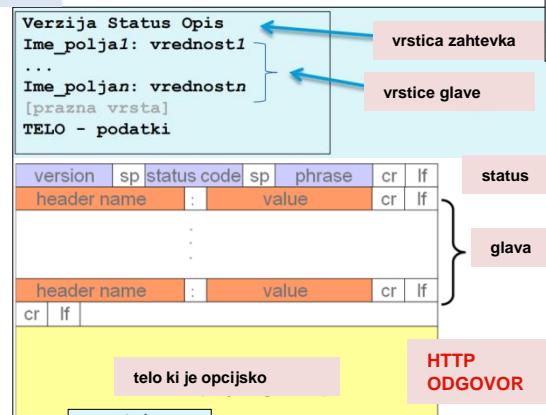
specifikaciji: RFC 1945 (HTTP 1.0)
in RFC 2616 (HTTP 1.1)

DELOVANJE :

- odjemalec naslovi **TCP zahtevo**
- zahteva na **vrrata 80** strežnika
- strežnik vrne **HTTP odgovor**
- **TCP** poskrbi za potrditev, ponovitve, vrstni red
- protokol **brez stanj**
- imamo 2 tipa sporočil torej **request** in **response**

VRSTE HTTP POVEZAV

- **nevztrajne** : za vsak prenašani objekt se vzpostavi **nova TCP povezava**
- **vztrajne** : strežnik uporabi isto povezavo za pošiljanje več objektov in pusti povezavo odprto
- **vztrajne z cevovodom** : tekoče pošiljanje več zahtev naenkrat **brez čakanja** na prejem prejšnjih



GET zahteva objekta

POST zahteva objekta + poslane vrednosti (obrazci)

- obrazec lahko uporabi tudi metodo GET, vrednosti parametrov pa pošilja kot podaljšan naslov (<http://...../search/rs.aspx?lang=slv&id=31318>)

HEAD zahteve, na katero strežnik odgovori z odgovorom brez zahtevanega objekta (uporabno za razroščevanje)

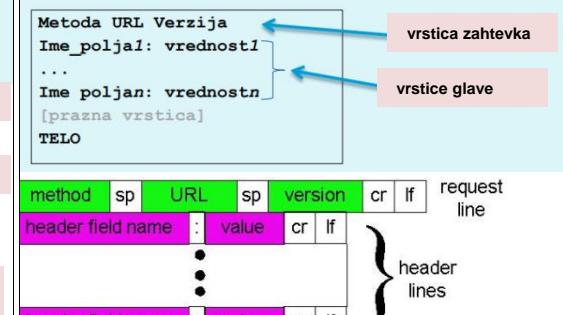
PUT (HTTP 1.1) – nalaganje na strežnik (upload)

DELETE (HTTP 1.1) – brisanje s strežnika

TRACE razroščevanje (echo-odmev zahtevka, podobno PING)

CONNECT povezava preko medstrežnika

OPTIONS povpraševanje o možnih opcijah pri zahtevku



HTTP ZAHTEVA

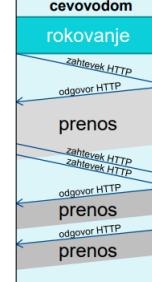
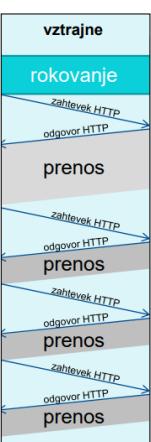
nevztrajne : 2 RTT / objekt
vztrajne : 1 RTT / objekt

PİŞKOTKI

- HTTP ne pomni stanja povezav zato med seboj **ne razlikuje** odjemalcev in ne pomni **zgodovine**
- nad plastjo HTTP se ustvari **sejna plast** s stanji
- problem je **varovanje zasebnosti**
- zato uporabimo piškotke
- **identifikator** ki ga strežnik dodeli uporabniku in ga uporabnik **lokalno shrani**

elementi:

- datoteka z ID piškota shranjena na računalniku
- strežniki hrani **evidenco** o uporabnikih izdanih piškotov
- ID piškota dodamo v **glavo** zahteve in odgovora



MEDSTREŽNIK : PROXY

- **web cache**, proxy strežnik, navadno pri ISP
- odgovarja na **zahteve** namesto strežnikov, ima svoje kopije spletnih strani
- jo zahteva od pravega strežnika če kopije **nima**
- odjemalec mora biti ustrezno konfiguriran

zakaj uporabljamo medstrežnike
? imajo vlogo **odjemalca** in strežnika, omogoča **hitrejši odgovor** odjemalcu, **manj prometa** na dostopni povezavi do javnega omrežja

HTTP Status Codes

For great REST services the correct usage of the correct HTTP status code in a response is vital.

1xx – Informational	2xx – Successful	3xx – Redirection	4xx – Client Error	5xx – Server Error
This class of status code indicates a provisional response, consisting only of the Status-Line and optional headers, and is terminated by an empty line 100 – Continue 101 – Switching Protocols 102 – Processing	This class of status code indicates that the client's request was successfully received, understood, and accepted. 200 – OK 201 – Created 202 – Accepted 203 – Non-Authoritative Information 204 – No Content 205 – Reset Content 206 – Partial Content 207 – Multi-Status	This class of status code indicates that further action needs to be taken by the user agent in order to fulfill the request. 300 – Multiple Choices 301 – Moved Permanently 302 – Found 303 – See Other 304 – Not Modified 305 – Use Proxy 307 – Temporary Redirect	The 4xx class of status code is intended for cases in which the client seems to have erred.	Response status codes beginning with the digit "5" indicate cases in which the server is aware that it has erred or is incapable of performing the request. 500 – Internal Server Error 501 – Not Implemented 502 – Bad Gateway 503 – Service Unavailable 504 – Gateway Timeout 505 – HTTP Version Not Supported 506 – Variant Also Negotiates 507 – Insufficient Storage 510 – Not Extended

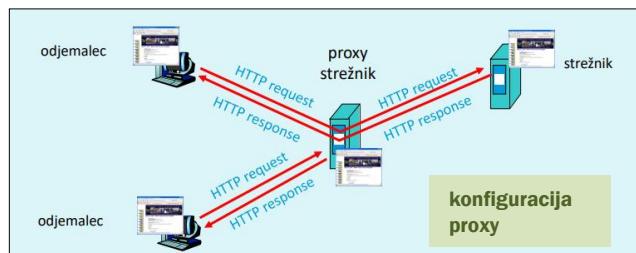
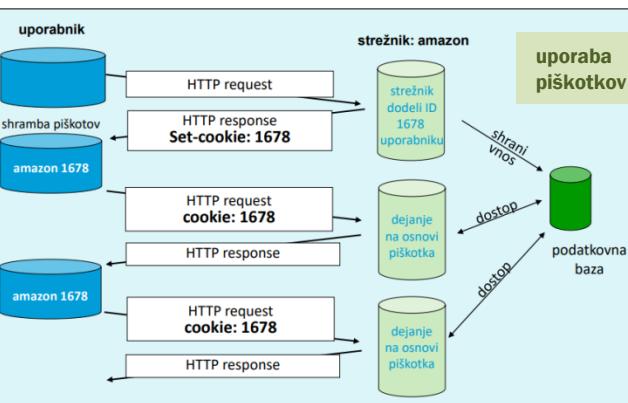
Examples of using HTTP Status Codes in REST

- 201 – When doing a POST to create a new resource it is best to return 201 and not 200.
- 204 – When deleting a resources it is best to return 204, which indicates it succeeded but there is no body to return.
- 301 – If a 301 is returned the client should update any cached URI's to point to the new URI.
- 302 – This is often used for temporary redirect's, however 303 and 307 are better choices.
- 409 – This provides a great way to deal with conflicts caused by multiple updates.
- 501 – This implies that the feature will be implemented in the future.

Key	Description
Black	HTTP version 1.0
Blue	HTTP version 1.1
Aqua	Extension RFC 2295
Green	Extension RFC 2518
Yellow	Extension RFC 2774
Orange	Extension RFC 2817
Purple	Extension RFC 3648
Red	Extension RFC 4918

Special Cases

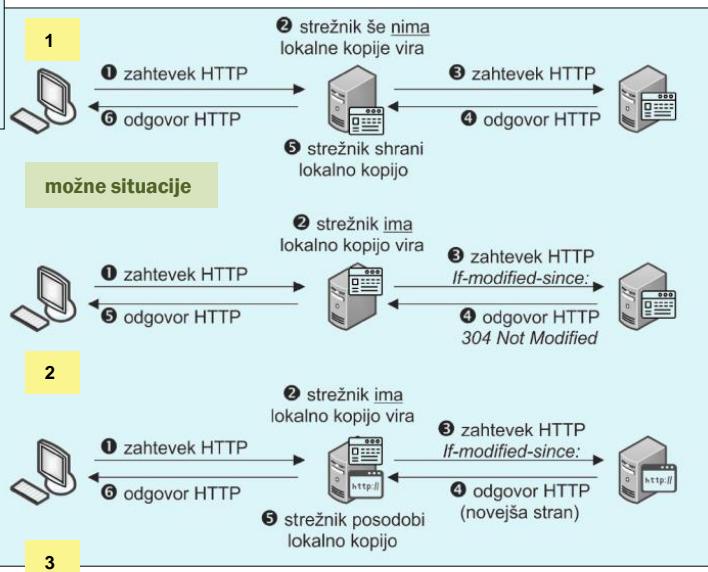
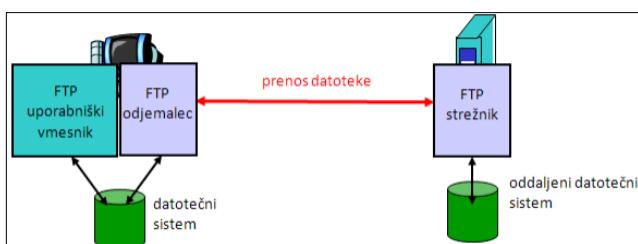
- 306 – This status code is no longer used. It used to be for switch proxy.
- 418 – This status code from RFC 2324. However RFC 2324 was submitted as an April Fools' Joke. The message is *I am a teapot*.



protokol FTP

BASICS: File Transfer Protocol

- prenos datotek z oddaljenega računalnika k uporabniku
- protokol hrani stanje povezave
- strežnik ve kdo je odjemalec



DELOVANJE :

- 2 ločeni TCP povezavi na **FTP strežnik**
- vrata **TCP 21** so **kontrolna povezava**
- to so ukazi za prenos datotek, uporabniško ime in geslo, menjava map...
- vrata **TCP 20** so za **prenos podatkov**
- na zahtevo odjemalca strežnik odpre povezavo po kateri prenaša podatke
- težava med povezavo strežnik odjemalec za **TCP 20** če je vmes **požarni zid** ali **NAT**
- zato vzpostavitev podatkovne povezave **od odjemalca k strežniku**
- temu rečemo **pasivni način**

```

C:\WINDOWS\system32\cmd.exe
220 ProFTPD 1.3.1 Server <ProFTPD> [64.120.98.33]
USER anonymous
331 Anonymous login ok, send your complete email address as your password
PASS blogger@webdigicloud.co.uk
230 Anonymous access granted, restrictions apply
HELP
214-The following commands are recognized (*=>'s unimplemented):
CWD  XCWD  CDUP   XCUP   SMNT*  QUIT   PORT   PASV
EPRT  EPSV  ALLO*  RNFR  RNTO  DELE  MDIM   RMD
XRMF  MKD   XMKD  PWD   XPWD  SIZE   SYST   HELP
NOOP  FEAT   OPTS  AUTH*  CCC*  CONF*  ENC*  MIC*
PBSZ* PROT*  TYPE STRU  MODE  REIR  STOR  STOU
APPE  REST  ABOR  USER  PASS  ACCT*  REIN*  LIST
NLST  STAT  SITE
214 Direct comments to root@datatracker.ietf.org
QUIT
221 Goodbye.

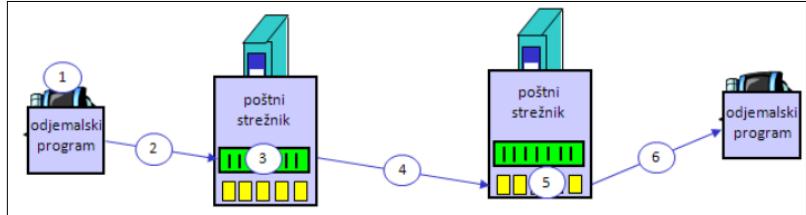
Connection to host lost.

```

sporočila FTP protokola:
nadzorna povezava uporablja **7 bitne ASCII ukaze**

ELEKTRONSKA POŠTA

- hitro, poceni in enostavna dostava
- priponke, povezave in HTML
- **poštni strežniki** hranijo **poštne predale**
- strežniki imajo **izhodno vrsto** sporočil
- odjemalski programi so tiskovni ali grafični
- protokol za prenos sporočil je **SMTP**



PROTOKOL SMTP: Simple Mail Transfer Protocol

- star več kot **30 let** ki uporablja TCP vrata **25**
- ukazi in telo sporočila morata biti kodirana z **7 bitnim ASCII**
- **binarne priponke** je potrebno prekodirati v **ASCII**
- na prejemni strani jih zakodiramo nazaj v **binarno obliko**
- sporočilo je sestavljeno iz **glave, prazne vrstice in telesa**
- **glava** sporočila hrani **podatke** o sporočilu

1. uporabnik uporabi **odjemalca** da **sestavi** sporočilo
2. odjemalec **pošlje** sporočilo strežniku
3. strežnik shrani sporočilo v **izhodno vrsto**
4. strežnik vzpostavi **TCP** povezavo s poštnim strežnikom prejemnika in **prenese** sporočilo
5. prejemnikov strežnik shrani sporočilo v pravi **poštni predel**
6. prejemnik ima odjemalski program za **prenos** in **branje** sporočila

SMTP	HTTP
PUSH princip	PULL princip
7 bitni ASCII nabor	lahko binarno kodiranje
več objektov lahko v enem sporočilu	vsak objekt se pošilja v svojem sporočilu
uporablja vztrajne povezave	uporablja vztrajne ali neveztrajne povezave

POP: Post Office Protocol

- je **preprost**
- ima omejeno funkcionalnost
- uporablja TCP vrata **110**
- **3 faze:** avtorizacija, prenos sporočil in oznak, posodabljanje
- slabosti: **lokально urejanje pošte, ne hrani stanja** med sejami

IMAP: Internet Mail Access Protocol

- kompleksen in zahtevnejši
- ima večjo funkcionalnost
- uporabnik lahko **določi mape** na strežniku
- možen prenos **le delov** sporočil
- večja **obremenitev strežnika**

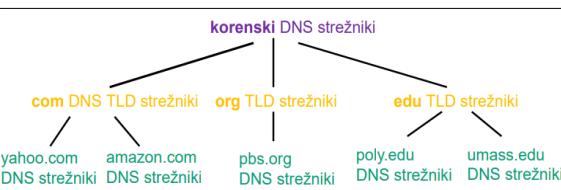
HTTP:

- brskalnik dostop od **kjerkoli**
- **brezplačni** ponudniki
- mape kot pri **IMAP**
- skripte na **HTTP** strežniku komunicirajo s poštnim strežnikom

imenjska storitev: DNS

NAMEN:

- naprave identificiramo z **imenom** in IP številko
- **DNS preslika** imena v IP naslove
- transparentnost: **več imen** gre lahko v **isti IP**
- porazdeljevanje bremena: **eno ime v več IP**
- porazdeljena **hierarhična** podatkovna zbirka
- protokol za poizvedovanje po zbirki



hierarhična organizacija:

- če bi imeli en **centralni strežnik** bi nastali problemi
- **enotna točka odpovedi**
- težko vzdrževanje
- rešitev ni **skalabilna** ?
- zato naredimo hierarhična organizacijo imen strežnikov in povpraševanje
- **13 korenskih strežnikov** od A do M
- **TDL:** top level domain strežniki
- **generične domene:** 7 prvotnih in okoli 1500 dodatnih
- prvotne: com, edu, gov, mil, org, net, biz
- **domena za države:** 255
- **avtoritativni strežniki:** organizacija z javnimi računalniki

primer: uni-lj

primer dostopa do www.ime.com:

- povpraša **korenski** strežnik po naslovu TLD strežnika za **domeno .com**
- povpraša **.com TLD DNS** strežnik po naslovu avtoritativnega strežnika **ime.com**
- povpraša strežnik podjetja **ime.com** o IP naslovu za **www.ime.com**

lokalni strežniki: posredniki do DNS hierarhije, običajno poizvedbe napotimo k njemu, ponavadi ga nudi ISP)

ITERATIVNA POIZVEDBA:

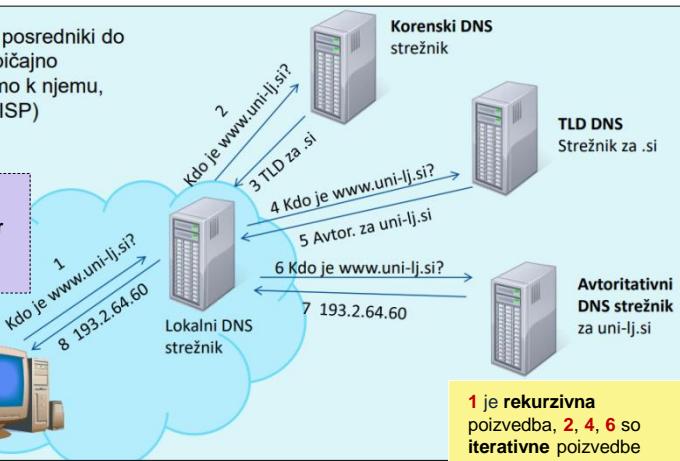
- strežnik lahko vrne **končni odgovor**
- strežnik lahko vrne **naziv strežnika** za naslednje poizvedovanje

ZAPIS DNS:

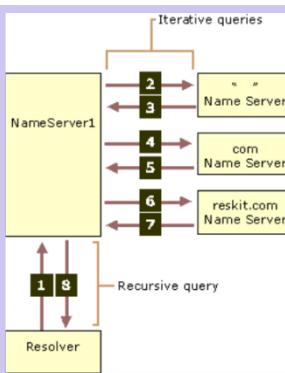
Name in Value
Type: 4 tipi
TTL: čas veljavnosti zapisa

REKURZIVNA POIZVEDBA:

- strežnik poiče **preslikavo imena** in vrne odgovor
- **razbremenimo** končne cliente komunikacije in povpraševanja
- možnost **centralnega predponjenja** v lokalnem strežniku



1 je **rekurzivna** poizvedba, 2, 4, 6 so **iterativne** poizvedbe



DNS PREDPOMNJEVANJE: caching

- DNS strežnik si lahko **zapomni** prejete **odgovore**
- dosežemo **hitrejši odziv**
- dosežemo **manj prometa** v omrežju to je pomembno ker **DNS** povzroča del čakanja pri **HTTP** zahtevkih
- zapomni si lahko tudi naslove **TLD** strežnikov, tako razbremenimo korenski strežnik
- možna tarča **napadov**

kaj nam pove Type v zapisu?

- A: address **Name** je ime računalnika **Value** pa IP številka, ti zapis so shranjeni v avtoritativnih strežnikih za svoje gostitelje
- AAAA: isto kakor A vendar imamo IPv6
- NS: name server, **Name** je ime domene in **Value** ime avtoritativnega DNS strežnika, ti zapis običajno v TLD strežnikih za iskanje avtoritativnega strežnika neke domene
- CNAME: canonical name, **Name** je nek **alias** in **Value** je pravo kanonično ime
- MX: mail exchange, **Name** je alias poštnega strežnika in **Value** je pravo ime poštnega strežnika

protokol DNS uporablja UDP vrata 53, strežnik ne hrani stanja **povezav** skrbi za ponovno pošiljanje, polje ima 16 bitov za ID ki povezuje **zahteve** in **odgovore**, **DNSSEC** je razširitev ki zagotavlja večjo varnost

novi DNS vnesi: podjetje registrira domeno pri **registrarju** posreduje mu **imena** in IP številke svojih avtoritativnih strežnikov, nato mu registrar dopolni **bazo TLD strežnikov**. v avtoritativni strežnik vnesemo zapis tipa A za spletni strežnik, MX zapis za **poštni strežnik domene**

identifikator	zastavice
število poizvedb	število zapisov v odgovoru
število avtoritativnih zapisov v odgovoru	število dodatnih zapisov v odgovoru
poizvedbe (spremenljivo število)	
odgovori (spremenljivo število zapisov)	
avtoritativni odgovori (spremenljivo število zapisov)	
dodatni zapisi (spremenljivo število zapisov)	

12 bajtov

parametri poizvedbe:
URL, tip, razred

deli odgovorov:
URL, tip, razred,
TTL, dolžina, odgovor

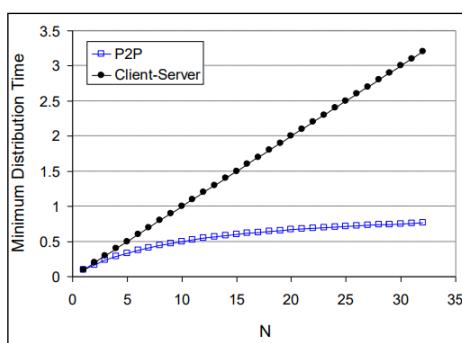
dopolnitev baze TLD strežnikov za domeno www.mojafirma.com

(mojafirma.com, dns1.mojafirma.com, NS)
(dns1.mojafirma.com, 212.212.212.1, A)

P 2 P S T O R I T V E

ARHITEKTURA 2P2

- **ni strežnika** ki bi bil nenehno **prižgan**
- izmenjava podatkov med poljubnima **končnima sistemoma**
- odjemalci sodelujejo **po potrebi** pri priključku menjajo IP naslov
- problemi so **varnost** in **iniciativnost** pri sodelovanju ter **NAT**
- sistem **Klient** in **strežnik**: čas prenosa **linearno** narašča s številom odjemalcev
- sistem **2P2**: čas prenosa je krajši ker poteka tudi med klienti
- zato večja **skalabilnost** sistema



BitTorrent:

- sosede vpraša po **razpoložljivih koščkih datotek**
- zahteva najprej tiste manjajoče ki so med sosedov **najbolj redki**
- opazujemo hitrost prejemanja od sosedov zagotavlja **pravičnost**
- pošiljamo jim koščke s **sorazmerno** visoko hitrostjo
- odjemalec lahko po prejemu datotekе ostane ali odide, sebično tho, guys nimamo usi za netflix
- **aplikacijska sporočila** imajo kontrolna bita za status odjemalca

STATUS:

statusa **choked** in **interested** 😊 na začetku je stanje vsake povezave pri odjemalcu **choked** pretok podatkov se izvaja, kadar je eden **zainteresiran**, drugi pa **ni zamašen** namen zastavice za zamašitev: **nadzor zasičenja** preko različnih **TCP** povezav

Skype:

- lastniški aplikacijski protokol torej so **podrobnosti neznane**
- nekatere pridobimo z **obratnimi inžiniringom**
- strežnik za prijavo preverja podatke o uporabnikih
- nadzorna vozlišča **supernodes**

SUPERNODES:

hranijo preslikave uporabniškega imena v IP **naslov** in skrbijo za **povezovanje** med uporabniki

PREMOSTITVENA VOZLJČA

- NAT povzroča težave v P2P arhitekturah
- odjemalci ne morejo **direktno** kontaktirati odjemalca za prehodom NAT

rešitev:

- odjemalca A in B vzpostavita zvezo preko svojih nadzornih vozlišč **NA** in **NB** 1 2 3
- NA in NB izbereta tretje **premostitveno** oz. **relay** nadzorno vozlišče **NR**
- A in B vzpostavita sejo s tem vozliščem
- ker sta odjemalca prva vzpostavila zvezo lahko prejemata **dohodni promet** od premostitvenega vozlišča 4 5
- to vozlišče poskuša zagotoviti **neposredno** povezavo med odjemalcema 6

PREDSTAVITVENA PLAST

→ **predstavitev podatkov:** različni sistemi predstavljajo **binarne tipe** na različen način, uporaba sintakse **ASN.1** zmanjša število vseh možnih preslikav

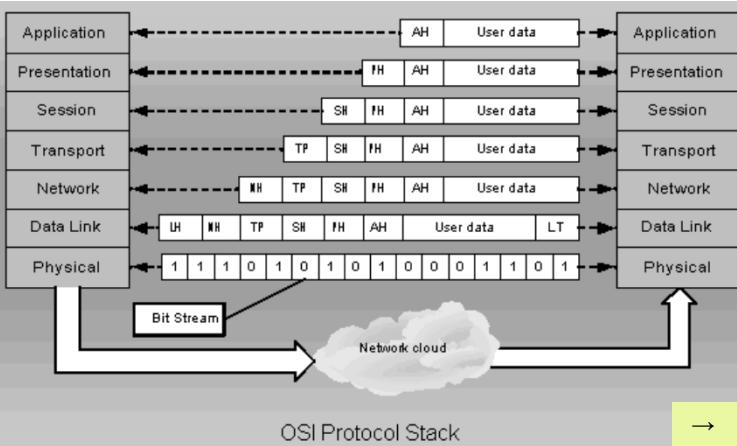
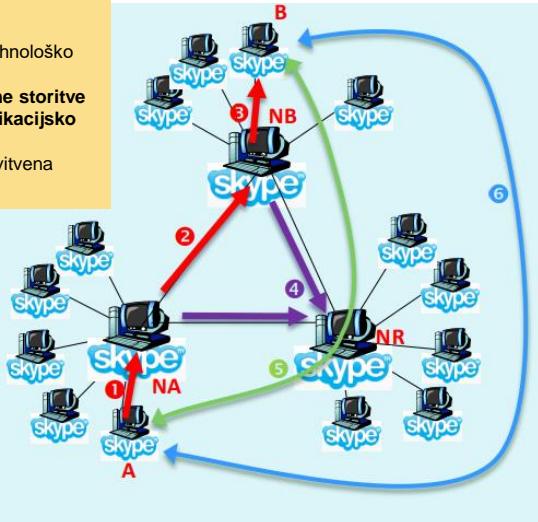
APLIKACIJSKE STORITVE:

- delovati morajo v skladu s tehnoško infrastrukuro
- aplikacija potrebuje **podporne storitve**
- v **OSI modelu** ležijo med **aplikacijsko plasto** in **omrežjem**
- podporne storitve:** predstavitev plas in sejna plast

→ **predstavitev alfanumeričnih znakov:** znaki so predstavljeni s številkami po **kodnem sistemu**, potrebno zagotoviti, da se prenašajo pravi znaki

→ **stiskanje podatkov:** omogoča zmanjšanje velikosti podatkov in s tem **pohitritev** prenosa

→ **zaščita podatkov:** varnostni mehanizem, ki omogoča vpeljavo zaupnosti v komunikacijo, **šifriranje**



SEJNA PLAST

→ **naloge:** vzpostavljanje, rušenje, vzdrževanje sej med aplikacijama oziroma nasprotno **dialog** med aplikacijama,
odgovornost za **obnovitvene točke** oziroma **checkpoints** seje in obnovo oziroma **recovery** če seja ne uspe

sinhronizacija podatkov iz različnih tokov in virov

KRIPTOGRAFIJA

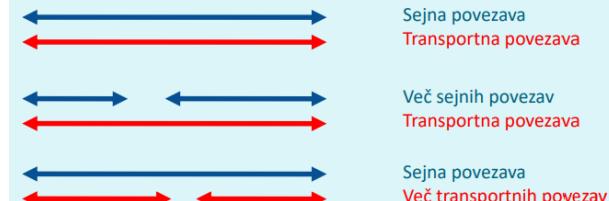
OMREŽNA VARNOST:

- zaupnost:** samo pošiljalatelj in prejemnik naj bi lahko brala sporočilo
- integriteta:** želimo da sporočilo med prenosom ni bilo spremenjeno
- identifikacija in avtentikacija:** pošiljalatelj in prejemnik želita preveriti in potrditi medsebojni identiteti
- varnost v praksi je **operativna varnost**
- to so naprave: **požarni zidovi**, sistemi za zaznavanje udonov...
- varnost je potrebna na **vseh** plasteh

→ **protokoli:**

- H.245: Call Control Protocol for Multimedia Communication
- RTCP: Real-time Transport Control Protocol
- SCP: Session Control Protocol

možni odnosi med **transportno** in **sejno** plastom



omrežna varnost na različnih plasteh:

- fizična:** šifriranje povezave
- omrežna:** filtriranje paketov, opazovanje prometa
- transportna:** šifriranje povezav med procesoma
- aplikacijska:** avtentikacija na podlagi identitete

terminologija kriptografije: čistopis iz ene strani gre skozi nek **enkripciji** algoritom z enkripcionskim **ključem** dobimo **kriptogram** nato gre čez **dekripciji** algoritom z dekripcionskim ključem in dobimo čistopis

prislушкиvanje:
prestrezanje sporočil, to je pasivni napad

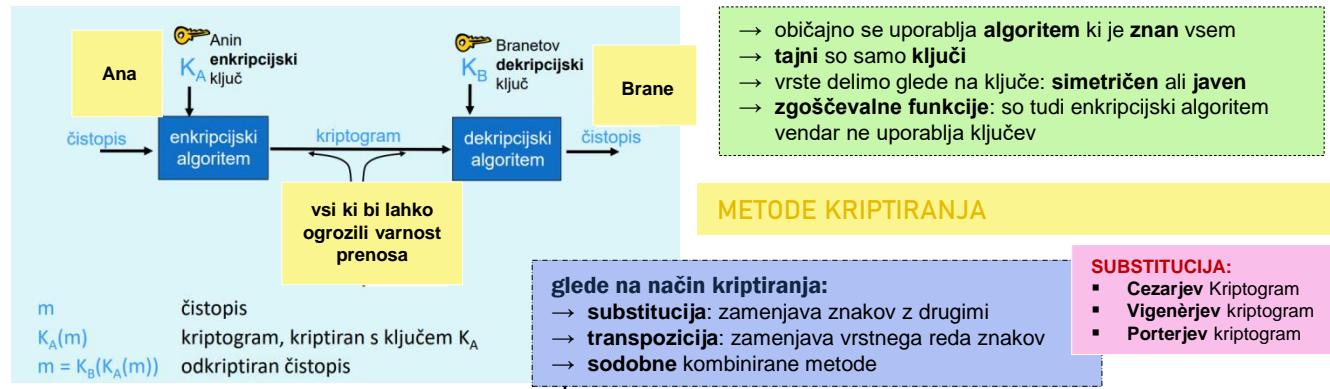
preprečevanje zanikanja komunikacije

kraja identitete:
napadalec lahko ponaredi izvirni naslov v paketu

spreminjanje sporočil:
to je aktivni napad

ugrabitev seje: hijacking, kjer napadalec odstrani pošiljalatelja in prevzame njegovo vlogo

omogočanje avtorizacije:
katere aktivnosti lahko uporabnik izvaja v sistemu



SIMETRIČNA KRIPTOGRAFIJA

CEZAR

- klasična metoda
 - kriptogram: **substitucija** z zamikom za **k črk**
 - **kluč** je število k
 - imamo **25 možnih** klučev
 - kriptogram razbijemo v največ **25 poskusih**

VIGENÈR

- večbesedno kriptiranje
 - **Viegenerjeva matrika:** vse Cezarjeve abecede
 - **ključ** je niz D črk in vsaki pripada ena vrstica
 - z abecedo n-te črke kriptiramo: **n-to** črko in jim pristevamo D torej n + D-to črko itd.
 - statistika jezika in semantika neuporabni

računalniške komunikacije	juni	javni izpit in raziskovalno-raziskovalno
dpa dejorazenpriekonomiki	Septembra	pabospetvsepost
arem		

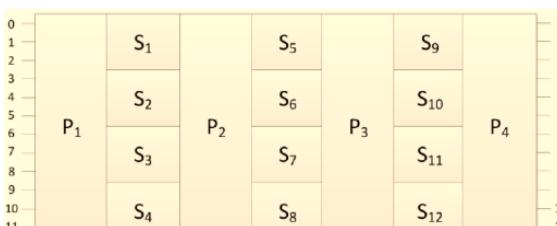
primer kjer je geslo računalniške komunikacije
in sporočilo: junija vsi izpiti na žalost odpadejo,
razen pri ekonomiki, septembra pa bo
spet vse po starem. D je 24

TRANSPOZICIJA

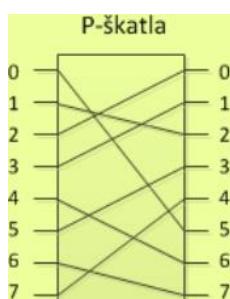
- spremenimo zaporedje znakov ali delov besedila
 - uporabimo ključ, oštevilčimo črke po abecedi
 - zapišemo **stolpce** glede na **oštevilčenje** črk

čistopis	criptogram
000	101
001	010
010	000
011	001
100	110
101	011
110	111
111	100

možno je
kombiniranje škatel v
preslikovalno kaskado
za poenostavitev
logike



primer
če je k 3



- običajno se uporablja **algoritem** ki je **znan** vsem
- **tajni** so samo **ključi**
- vrste delimo glede na ključe: **simetričen** ali **javen**
- **zgoščevalne funkcije**: so tudi enkripcijski algoritem vendar ne uporablja ključev

METODE KRIPTIRANJA

SUBSTITUCIJA:

- Cezarjev Kriptogram
 - Vigenèrjev kriptogram
 - Porterijev kriptogram

glede na način kriptiranja:

- **substitucija**: zamenjava znakov z drugimi
 - **transpozicija**: zamenjava vrstnega reda znakov
 - **sodobne kombinirane metode**

glede na velikost sporočila:

- **znamkovna**
 - **bločna**: zakriptiramo zaporedja znakov

KRIPTOANALIZA:

- razbijanje kriptogramov
 - na podlagi **posameznega besedila**, ki vemo da bo v sporočilu npr. beseda login
 - zato zakriptiramo **le vsebino** ne pa cele komunikacije
 - na podlagi **statistike jezika**, kjer potrebujemo daljše besedilo
 - če **poznamo vsebino lažje** saj iščemo pričakovane **korene** besed

P O R T E R

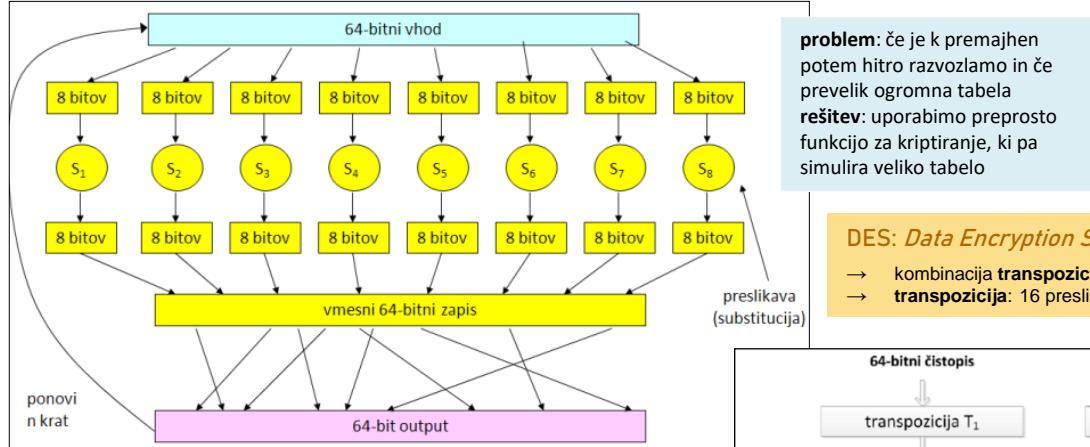
- kriptiramo po **dva znaka** naenkrat
 - **vrstica** je prvi znak **stolpec** drugi

primer geslo je **kača**

SODOBNA SIMETRIČNA KRIPTOGRAFIJA

- **bločna**: sporočilo m kriptiramo tako, da obdelamo posamezne bloke **k bitov**
 - preslikava med bloki sporočila in kriptograma je **bijektivna** in tako **enolična**

- permutacijska škatla: na primer key (23157046)
- substitucijska škatla: dekoder-permutacija-koder



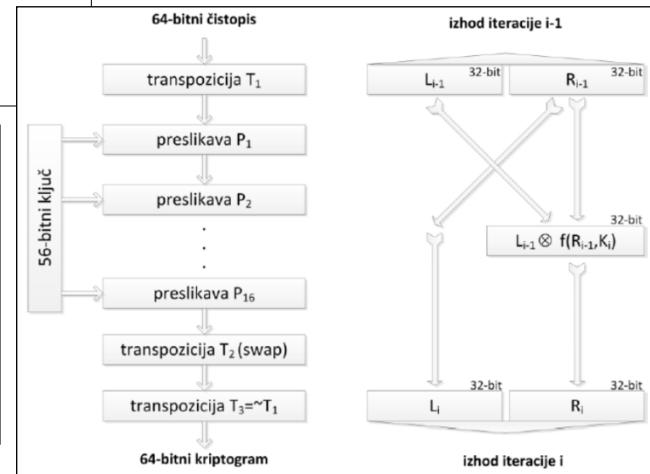
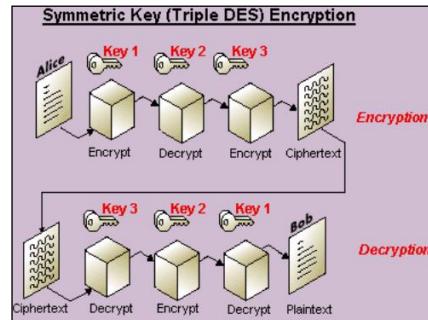
problem: če je k premajhen potem hitro razvzlamamo in če prevelik ogromna tabela
rešitev: uporabimo preprosto funkcijo za šifriranje, ki pa simulira veliko tabelo

DES: Data Encryption Standard

- kombinacija **transpozičijskih** in **substitucijskih** metod
- **transpozicija:** 16 preslikav SWAP transpozicija

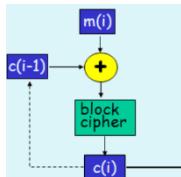
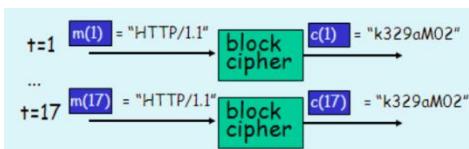
3DES: Trojni DES

- trikrat šifriranje
- trikrat počasneži varnjenje za brute force kрат $2^{2 \cdot 56}$
- dodatni dve šifriranji z 56 bitnim ključem
- združljivost z DES če je $K_2 = K_3$



AES: Rijndael algoritom

- najbolj učinkovita in varna metoda
- bloki po 128 bitov
- ključ je 128, 192 in 256 bitov
- dober računalnik 10^9 let za razbijanje
- **AES operacije:** zamikanje vrstic, zamenjave stolpcev, izpeljave ključev



pošiljatelj: šifrirja vsako sporočilo v kriptogram
 $c(i) = K_S(m(i) \otimes c(i-1))$

prejemnik: odšifrirja
 $m(i) = K_S^{-1}(c(i)) \otimes c(i-1)$

PRIMER

$$\bullet m = 010010010, IV = c(0) = 001, k = 3$$

• pošiljatelj:

- $c(1) = K_S(m(1) \text{ XOR } c(0)) = K_S(010 \text{ XOR } 001) = K_S(011) = 100$
- $c(2) = K_S(m(2) \text{ XOR } c(1)) = K_S(010 \text{ XOR } 100) = K_S(110) = 000$
- $c(3) = K_S(m(3) \text{ XOR } c(2)) = K_S(010 \text{ XOR } 000) = K_S(010) = 101$

m	$K_S(m)$
000	110
001	111
010	101
011	100
100	011
101	010
110	000
111	001

ASIMETRIČNA KRIPTOGRAFIJA

kako deluje asimetrična kriptografija?

- **enkripcijiški E** in **dekripcijiški D** key sta lahko različna
- **E** je lahko javen **D** pa mora biti **tajen**
- $D(E(m)) = m$ in iz **m**, **E** in **E(m)** je **nemogoče** ugotoviti **D**

razbijanje bločnih kriptosistemov z grobo silo 2013

najbolj znan tak je **RSA** algoritem ki deluje na podlagi praštevilskog razcepja

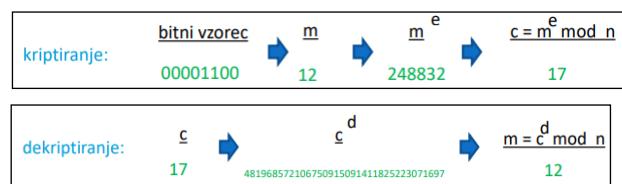
RSA: Rivest, Shamir, Adleman

- počasen: uporabimo za izmenjavo simetričnega ključa
- izberemo dve vsaj 1024 bitov veliki **praštevili** p in q
- $n = p \cdot q$ in $z = (p - 1) \cdot (q - 1)$
- izberemo tak e da nima skupnih deliteljev z z
- izberemo d tako da $e \cdot d = 1 \pmod z$
- definiramo javni ključ $E = (n, e)$
- definiramo zasebni ključ $D = (n, d)$
- varen ker težko najdemmo razcep n

primer:

- $p = 7$ in $q = 5$
 dobimo $n = 35$
 dobimo $z = 24$
 izberemo $e = 5$
 izberemo $d = 29$

šifriranje: $m \rightarrow c = m^e \text{ mod } n$ **šifriranje**
 $c \rightarrow m = c^d \text{ mod } n$ **dekriptiranje**



PRINCIPI VARNE KOMUNIKACIJE

AVTENTIKACIJA

AVTENTIKACIJA UDELEŽENCEV:

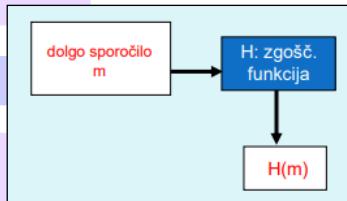
- udeleženci lahko **preverijo** podatke
 - da je pošiljalatelj dejansko oseba za katero se **predstavlja**
 - da je prejemnik zagotovo prava oseba in ne **prisluškovalec**

INTEGRITETA

- udeleženci lahko preverijo da sporočilo ni spremenjeno
 - da je sporočilo res od **zdaj**
 - uporabljamo **zgoščevalne kriptografske funkcije**
 - so preprosto izračunljive
 - iz $H(m)$ ne moremo ugotoviti **m**
 - **težko** najdemo m in n da $H(m) = H(n)$
 - zgleda kot **naključen** niz
 - primer je internet checksum
 - **MD5** in **SHA-1**

2. NAČIN:

- **pošiljatelj B** z zgoščevalno funkcijo H podpiše le **zgoščeno** vrednost
 - z og **nekriptiranim** sporočilom pošlje **kriptirano** zgoščeno vrednost



ZGOŠČENA VREDNOST SPOROCILA:

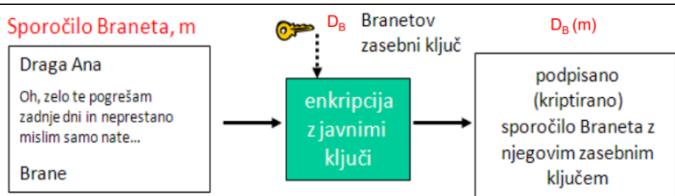
- s sporočilom **m** pošljemo tudi **H(m)**
 - prejemnik preveri ali se **ujemata**
 - za preverjanje avtentifikacije
uporabimo še **avtentifikacijski ključ s**
 - pošljemo **m** in **H(m + s) = MAC**
 - problem distribucija tega ključa

DIGITALNI PODPIS:

- način za zamenjavo osebnega podpisa ki enolično potrjuje identiteto
 - avtentikacija **MAC** ni enolična
 - uporabimo kriptografijo z **javnimi ključi**

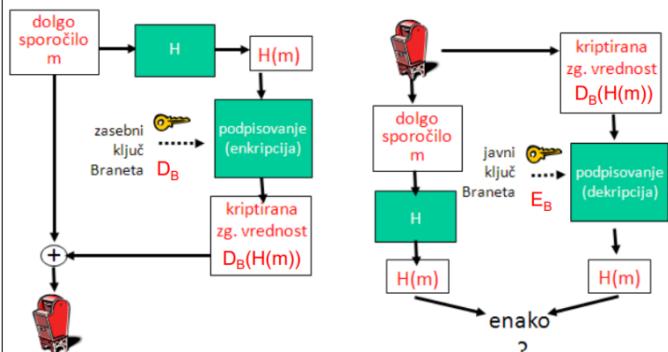
1. NAČIN

- **pošiljatelj B** lahko izračuna $D_B(m)$
 - to lahko predstavlja **podpisani** dokument
 - časovno zahtevno pri dolgih sporocilih



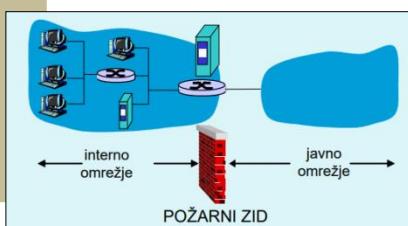
Brano pošlo podpisano sporočilo:

Ana preveri podpis in integriteto digitalno podpisane sporočila:



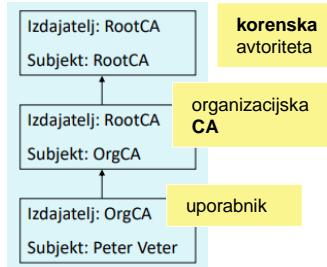
OPERATIVNA VARNOST

- **administrator** omrežja del uporabnike na good guys in bad guys
 - **good guy:** uporabnik legitimno uporablja vire omrežja in pripada organizaciji
 - **bad guy:** vsi ostali, dostope moramo skrbno nadzorovati – bili eyelash
 - omrežje običajno samo **eno točko vstopa**
 - na njej kontroliramo dostope



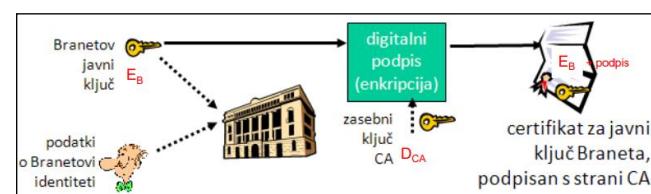
CERTIFIKACIJSKA AGENCIJA:

- vdiralec lahko podpiše sporočilo s **svojim zasebnim ključem** in se pretvarja, da je to ključ nekoga drugega
 - **certifikacijske agencije**: preverjajo povezavo med **javnim ključem** in identiteto osebe
 - agencija shrani povezavo med ključem in identitetom v **certifikat** ki ga agencija **podpiše s svojim zasebnim** ključem
 - **veriga zaupanja certifikatom**: z njim preverimo veljavnost certifikata, korenske avtoritete so znane



CERTIFIKAT

- ime **izdajatelja**
 - ime **osebe**
 - **naslov**
 - ime **domene**
 - **javni ključ** osebe
 - **digitalni podpis**: podpis z zasebnim ključem izdajatelja
 - uveljavljen standard za zapis certifikatov je **X.509**



FIREWALL

- izolira interna omrežja od velikega javnega omrežja
- določenim paketom dovoli prehod druge blokira
- filtrira ves promet
- prepriča samo doposten promet glede na zahteve
- je imun na napade

Kakšne vrste filtriranj poznamo?

- izolirano filtriranje paketov: pretežno filtriranje na podlagi podatkov v glavi, izvorni in ponorni naslovi ter vrata
- filtriranje paketov v kontekstu: nadzoruje vzpostavljenost povezave
- aplikacijski prehodi: filtriranje z vpogledom v podatke aplikacijske plasti

primer: onemogočimo ves promet razen WWW navzven in DNS v obe smeri

IZOLIRANO FILTRIRANJE

- filtriranje običajno izvaja že usmerjevalnik, ki meji na javno omrežje
- odloča se glede na vsebinsko paketov
- IP izvornega oz. ponornega naslova
- številke IP protokola: TCP, UDP, ICMP...
- TCP / UDP izvornih in ciljnih vrat
- tip sporočila pri ICMP
- zastavice TCP: SYN in ACK bit
- imamo access control list ACL
- tabela pravil od vrha proti dnu
- zapisani so v paru pogoj in akcija

FILTRIRANJE V KONTEKSTU

- izolirano filtriranje dovoli vstop nesmiselnim paketom čeprav notranji odjemalec ni vzpostavil povezave
- vodi evidentno o vsaki vzpostavljeni TCP povezavi
- zabeleži vzpostavitev povezave SYN in njen konec FIN
- tako se odloči ali paketi smiselnii
- po preteklu določenega časa povezava kot neveljavna
- uporablja podoben dostopovni seznam ACL
- ta določa kdaj je treba kontrolirati veljavnost povezave

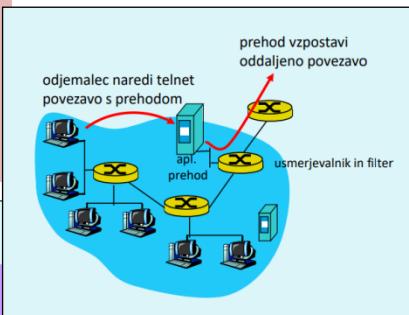
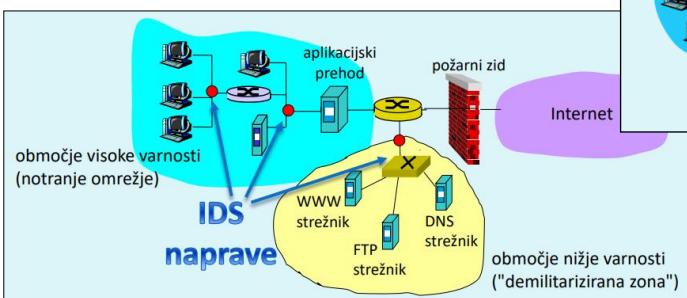
izvorni naslov	ciljni naslov	protokol	izvorna vrata	ciljna vrata	zastavica	akcija
222.22.16	izven 222.22.16	TCP	> 1023	80	any	dovoli
izven 222.22.16	222.22.16	TCP	80	> 1023	ACK	dovoli
222.22.16	izven 222.22.16	UDP	> 1023	53	---	dovoli
izven 222.22.16	222.22.16	UDP	53	> 1023	----	dovoli
all	all	all	all	all	all	zavrsi

APLIKACIJSKI PREHODI

- omogočajo dodatno filtriranje glede na izbiro uporabnikov
- omogočajo filtriranje na podlagi podatkov na aplikacijskem nivoju
- če uporabniki potrebujejo več aplikacij potrebuje vsaka svoj aplikacijski prehod
- klient je potreben nastaviti da se znajo povezati s prehodom

IDS IN IPS

- IDS dodatna naprava za poglobljeno analizo paketov
- na podlagi vstopa sumljivih paketov v omrežje lahko naprava prepreči njihov vstop ali razpošlje obvestila
- posieme sporočilo o potencialno škodljivem prometu
- IPS ukrepa pri pojavitvi sumljivega prometa
- primer: Cisco, CheckPoint, Snort IDS
- primerjava s shranjenimi vzorci napadov: signatures
- opazovanje netipičnega prometa: anomaly based



ZAZNAVANJE Z VZORCI

- vzorci lahko hranijo izvorni ali ponorni IP
- lahko hranijo protokol ali zaporedje bitov v podatkih paketa
- lahko so vezani na serijo paketov
- odvisno od baze znanih vzorcev
- slabo zaznavanje še nevidenih napadov
- možni lažni alarmi
- zahtevno procesiranje

Pogosti napadi na omrežne sisteme:

1. prislушкиvanje in ponarejanje sporočil
2. matematični napadi na kriptografske algoritme in ključe
3. ugibanje gesel: brute force, napad s slovarjem
4. virusi, črvi, trojanci
5. izkoriscenje šibkosti v programske opremi
6. socialni inženiring
7. pregled vrat: napadalec testira, kateri strežniki so delujoči in katere storitve so na voljo, tako pridobiva informacije o sistemu
8. brskanje po smeteh: informacije o sistemu
9. rojstnodnevni napad: je napad na zgoščevalne funkcije
10. back door: napadalec dostopi do sistema preko druge poti
11. ponarejanje IP naslovov
12. prestreganje komunikacije
13. ponovitev komunikacije
14. ugrabitve TCP sej: napadalec prekine komunikacijo med uporabnikoma in se vrne v mesto enega od njiju

ZAZNAVANJE NETIPICNEGA PROMETA

- sistem opazuje običajen promet in izračuna statistike vezane nanj
- sistem reagira na statistično neobičajen promet
- možno zaznavanje še nevidenih napadov
- težko ločevanje med normalnim in nenavadnim prometom

16. napadi s fragmentacijo: z razbijanjem paketa na fragmente razdelimo glavo paketa da jih požarni zid ne more filtrirati
17. odpoved delovanja sistema: obremenjuje omrežne vire tako, da se nehajo odzivati zahtevam regularnih uporabnikov

Napadi DoS:

- prekoracitev medpomnilnika: procesu pošljemo več podatkov, kot jih lahko sprejme
- SYN napad: napadalec pošlje veliko število zahtev za vzpostavitev povezave in se na odgovor sistema ne odzove
- Teardrop: spremeni podatke o št. in dolžini fragmentov v IP paketu
- Smurf: posredni broadcast za preobremenitev sistema
- uporaba botov