

## ISO-OSI medel

<i>Datagr.</i>	ISO-OSI	TCP-IP	<i>prot.</i>
podatki	aplikacijska	aplikacijska	HTTP, DNS, POP3, SSH
podatki	predstavitvena		
podatki	sejna		
seg-menti	prenosna	transportna	TCP, UDP
paketi	omrežna	internetna	IP, ICMP
okvirji	povezavna	povezovalna	MAC, PPP
biti	fizična		

## Fizična plast

Naloge:

- kodiranje bitov** z neko fizikalno veličino (*signalom*) za prenos po mediju (*baker, optika, radijski, IR, ...*)
- prenos posameznih bitov** v analogni ali digitalni obliki
- prenos celotnega signala** (zaporedja bitov po mediju)
- pretvorba signala** v obliko, ki je primerna za prenos po mediju

## Modulacija

- amplitudna* (1: velika amplituda, 0: majhna ampituda)
- frekvenčna* (1: visoka frekvenca, 0: nizka frekvenca)
- fazna* (1: zakodiramo z fazo 0°, 0: z fazo 180°)
- kvadratna*
  - kombinacija amplitudne in fazne modulacije
  - 4 fazni koti 0°, 90°, 180° 270°
  - 2 nivoja amplitude
  - 8 kombinacij faze in amplitude → 3 bite
  - lahko imamo tudi več faznih kotov in amplitud in naenkrat zakodiramo do 6 bitov

## Povezavna plast

Lahko (ne pa njuno) opravlja naloge: okvirajanje datagramov, zaznavanje in odpravljanje napak, dostop do medija, zagotavljanje zanesljive dostave, kontrola pretoka.

### Okvirjanje datagramov

Okvir je *enota* na povezavni plasti. Določa začetek in konec prenesenih podatkov.

### Zaznavanje in odpravljanje napak

dodamo bite za preverjanje pravilnosti (EDC - Error Detection Code)

- soda/liha pariteta** vseh 1 skupaj s pariteto je sodo/liho (*omogoča le zaznavanje lihega števila napak*)
- dvo dimenzionalna pariteta** dodamo praitetni bit za vsako vrstico in vsak stolpec + kumulativni/meta paritetni bit je pariteta paritetnih bitiv (*omogoča zaznavanje in odpravljanje enojnih napak in zaznavanje dvojnih napak*)
- Hammingova koda** Paritetni biti *p* imajo indekse 2<sup>*i*</sup> in jih postavimo na mesta, ki jih določajo indeksi. Vmes vstavimo bite podatkov *d*. Vsak paritetni bit *p<sub>i</sub>* pokriva *i* bitov od svoje pozicije naprej, nato preskoči *i* bitiov in tako naprej. Število 1, ki jih paritetni bit pokriva mora biti sodo.

	1.	2.	3.	4.	5.	6.	7.
	<i>p</i> <sub>1</sub>	<i>p</i> <sub>2</sub>	<i>d</i> <sub>1</sub>	<i>p</i> <sub>4</sub>	<i>d</i> <sub>2</sub>	<i>d</i> <sub>3</sub>	<i>d</i> <sub>4</sub>
<i>p</i> <sub>1</sub>	✓		✓		✓		✓
<i>p</i> <sub>2</sub>		✓	✓			✓	✓
<i>p</i> <sub>4</sub>				✓	✓	✓	✓

Ko prejmemo sporočio, preverimo paritetne bite in definiramo:

$$p_i' = \begin{cases} 1 & \text{paritetni bit } p_i \text{ napačen} \\ 0 & \text{sicer} \end{cases}$$

Nato izračunamo sindrom:

$$[\dots p_4' p_2' p_1']_{(2)} = \# \text{ bita, ki ga je treba obrniti}$$

- Cyclic Redundancy Check - CRC** z *r* paritetnimi biti lahko popravi do *r* + 1 napak
- Internet checksum** uporablja se na transportni plasti

### Dostop do skupinskega medija

Imamo dve vrsti povezav: **dvotočkovna** (le en pošiljatelj in prejemnik - PPP, DHLC) in **oddajna** (več vozlišč komunicira naenkrat - Ethernet, Wireless LAN).

Kanal lahko delimo na več načinov:

#### Delitev kanala

Time Division Multiple Access TDMA, Frequency Devision Multiple Access FDMA *Ni kolizij*

#### Sočasni/naključni dostop (kolizijski)

- ALOHA** Če pride do kolizije, se paket po naključnem času ponovno pošlje.
- razsekana ALOHA** Čas je razsekan na časovne intervale. V vsakem intervalu se lahko pošlje en paket. Če pride do kolizije, se v nasldnjem intervalu pošlje ponovno z verjetnostjo *p*.
- Carrier Sense Multiple Access - CSMA** Poslušaj ali že kdo oddaja preden začeneš oddajati sam.
- Carrier Sense Multiple Accerss Collision Detection - CSMA/CD** Če zaznaš kolizijo nehaj oddajati in pošlji motilni signal.

### Izmenični dostop

- Rezervacija s centralnim vozliščim (pooling):** centralno vozlišče zaporedno dodeluje medij posameznim vozliščem (*zakasnitev, ena točka odpovedi*)
- Rezervacija z žetonom, ki kroži (token)** (*zakasnitev, ena točka odpovedi - žeton*)

#### Dostop do brezžičnega medija

Protokol 802.11 (WiFi) za dostop do medija uporablja **Carrier Sense Multiple Accerss Collision Avoidance - CSMA-CA**.

Uporablja signala:

- Request To Send - RTS**
- Clear To Send - CTS**

Terminal, ki želi pošiljati najprej pošlje RTS in čaka na CTS. Ko terminal prejme CTS, ne bo oddajal toliko časa kot traja, da se prenese en paket.

### Protokoli na povezavni plasti

#### Ethernet

Okvir ethernet:

<i>pream-bula</i>	<i>ponorni nalosv</i>	<i>izvorni naslov</i>	<i>tip</i>	<i>podatki</i>	<i>CRC</i>
8B	6B	6B	2B	1500B	4B

Ethernet ponuja:

- nepovezavna storitev
- nezanesljiva storitev (*kontrola pravilnosti se izvaja s CRC a brez popravljanja, potrjevanje in ponovno pošiljanje se ne uporabljata*)
- uporablja CSMA/CD (*pri vsaki koliziji se čakanje eksponentno poveča* 0 – 1, 0 – 3, 0 – 7, 0 – 15, ...)

#### Point to Point Protocol - PPP

Povezavni protokol; imamo le enega pošiljatelja in prejemnika.

Okvir PPP:

<i>zastavica za začetek</i>	<i>address</i>	<i>control</i>	<i>protokol</i>	<i>podatki</i>	<i>CRC</i>	<i>zastavica za konec</i>
1B	1B	1B	1B/2B	≤1500B	2B/4B	1B

Poja “address” in “control” se ne uporabljata.

Pred niz 01111110 (zastavica) urivamo *ubežno kodo* 011111101.

### Naslavljanje naprav na povezavni plasti

Vsaka naprava ima fizični naslov (Media Access Control - MAC) iz 48 bitov.

Naslov FF-FF-FF-FF-FF-FF je **broadcast** (prejemniki so vsi).

### Address Resolution Protocol - ARP

Vsaka naprava ima ARP tabelo [IP | MAC | TTL]

Če naprava želi ugotoviti MAC druge naprave, pošlje **ARP poizvedbo** (*Moj max je y, kdo ima IP x?*) na broadcast naslov. Naparava, ki ima ta IP odgovori z **ARP odgovorom** (*Jaz imam IP x in oj MAC je z.*). Nato si obe napravi shranita MAC naslove v ARP tabelo.

### Stikalo

Stikalo uporablja **stikalno tabelo** [MAC | vrata | TTL], da se odloči, na katera vrata poslati okvir. Ko prejme kak okvir, si zapomni lokacijo pošiljatelja in okvir:

- poplavi** na vsa vrata razen na tista iz katerih je prišel okvir (*če ne ve, na katerih vratih je prejemnik*)
- posreduje** na izbrana vrata (*če ve, na katerih vratih je prejemnik*)
- filtriria** / zavrže okvir (*če je namenjen istim vratom iz katerih je prišel*)

## Omrežna plast

*Lahko omogoča:*

- zagotovljena dostava paketov
- dostava v zagotovljenem času
- dostava v pravem vrstnem redu
- zagotovljena spodnja meja pasovne širine
- največja dovoljena varianca zakasnitve (jitter)
- varno komunikacijo (zaupnost integriteto podatkov, avtentikacijo)

### Storitve interneta

		<i>zagotovljene storitve</i>				
Omrežje	Model	pas. širina	brez izgub	vr. red	čas	obv. o zamaš.
<i>Internet</i>	best effort	×	×	×	×	×
<i>ATM</i>	CBR constant bit rate	const	✓	✓	✓	ni izguba
<i>ATM</i>	ABR available bit rate	min	×	✓	×	✓

## Usmerjevalnik

Naprava, ki deluje na omrežni plasti in skrbi za transport datagrama po *jedru omrežja*. Povezuje ralične medije in portokole.

Funkcije usmerjevalnika:

- Posredovanje paketov**/forwarding: prenos na pravi izhodni vmesnik glede na destinacijo paketa in **posredovalno tabelo**.
- Usmerjanje**/routing: določanje poti paketov od izvora do cilja. Usmerjevalniki s poočjo usmerjevalnih protokolov najdejo najcenejšo pot.

## Povezavna omrežja / navidiezni vodi

Faze pri vzpostavitvi navideznega voda: *vzpostavitev, tok podatkov, rušenje*

Usmerjevalniki usmerjajo pakete glede na *št. voda* v paketu in posredovalno tabelo [ vhodni vmesnik | vhodna št. voda | izhodni vmesnik | izhodna št. voda].

## Nepovezavna / datagramska omrežja

Ni faze vpostavljanja povezave. Paket lahko do cilja pride po različnih poteh.

Usmerjevalniki usmerjajo glede na ciljni (IP) naslov in usmerjevalno tabelo [predpona naslova | vmesnik].

## Internet Protocol - IP

## Paket IPv4

Dolžina glave brez “možnosti” je 20B.

4b	4b	4b	4b	4b	4b	4b	4b
Verzija	Dolžina glave	Tip storitve		Skupna dolžina			
Identifikacija				Zastav.	Zamik fragmenta		
TTL		Protokol		Kontrolna vsota			
Izvorni IP naslov 32b							
Ponorni IP naslov 32b							
Možnosti						Zpolnjevanje	
Podatki							

## Fragmentacija

IP datagram se mora enkapsulirati v omejene okvirje (MTU) povezavne plasti. Zato se IP paket razbije na več manjših. Da fragmente lahko sestavimo nazaj se uporabljajo polja v glavi:

- ID**: pri vseh fragmentih enak
- Zastavica MF**: 0, če je to zadnji fragment; 1, sicer

- Offset**: enota za odmik je 8 Byte

### Podomrežja

Razred	Začetek naslova	Maska
A	0 xxxxxxx	255.0.0.0
B	10 xxxxxx	255.255.0.0
C	110 xxxxx	255.255.255.0
D	1110 xxxx	–
E	11110 xxx	–

### Posebni naslovi

- Privatni (24 bitni) 10.0.0.0/8
- Privatni (20 bitni) 172.16.0.0/12
- Privatni (16 bitni) 192.168.0.0/16
- Povratna zanka (localhost) 127.0.0.0/8
- Link-local 169.254.0.0/16

### Posebni naslovi v lokalnem omrežju

- Naslov omrežja: *1. naslov v omrežju (same ničle)*
- Hosts: *vsi naslovi vmes*
- Broadcast: *zadnji naslov v omrežju (same enke)*

## Dinaminčno dodeljevanje naslovov - DHCP

Ko se naprava priključi v omrežje, pošlje **DHCP discover**, DHCP strežnik(i) odgovori(jo) z **DHCP offer**, naprava izbere IP iz ponudbe in pošlje **DHCP request**, končno strežnik potrdi z **DHCP ACK**.

## Internet Control Message Protocol - ICMP

Sporočilo je enkapsulirano v paketu IP. Uporablja se za sporočanje napake, nedosegljivosti, protokola, vrat, ...

**Traceroute** pošlje vrsto paketov z TTL=1,2,3,... vsakič ko paketu poteče TTL router pošlje nazaj ICMP sporočilo.

## Network Address Translation - NAT

Lokalne naslove slika v globalne in obratno.

To doseže z NAT preslikovalno tabelo [globalni naslov, port | zasebni naslov, port].

Vsak zahtevek ima izvorni in ponorni port/vrata.

## IPv6

Naslovi so razdeljeni (:) na 8 blokov po 16 bitov. Zaporedje blokov samih ničel lahko krajšamo z “::”,

Dolžina glave je vedno 40B.

4b	4b	4b	4b	4b	4b	4b	4b
Verzija	Traffic class		Flow label				
Payload len				Next header		Hop limit	
Izvorni IP naslov 128b							
Ponorni IP naslov 128b							
Podatki							

Novo polje “flow label” (vrsta toka) za zagotavljanje kakovosti storitev za posebne tokove podatkov.

## Prehodni mehanizmi

- Dvojni silkad**: Vse naprave imajo IPv4 in IPv6. Če je na poti kakšno IPv4 vozlišče, se bo proromet vmes pretvarjal v IPv4.
- Tuneliranje**: Če je na poti kakpno IPv4 vozlišče se IPv6 paket enkapsulira v IPv4.

## Usmerjanje

Usmerjevalni algoritmi so lahko:

- centralizirani** (*imajo podatke o celem omrežju*) ali **decentralizirani** (*imajo podatke le o sosednjih vozliščih*)
- prilagodljivi** (*lahko prilagajajo cene povezav glede na zasičenost*) ali **naprilagodljivi**

### Decentralizirano usmerjanje

Vsako vozlišče pozna ceno povezave do svojih sosedov. Hrani **vektor razdalj**, **vektorje razdalj svojih sosedov** in **posredovalno tabelo** [cilj | cena | via sosed].

Ko vozlišče prejme vkektor razdalj svojega sosed, izračuna nov vektor razdalj *če lahko do nekega vozlišča pride ceneje prek nekega sosed*a, *posodobi posredovalno tabelo*.

Usmerjevalniki so povezani v **avtonomne sisteme - AS** v katerih uporabljajo isti **INTRA-AS** usmerjevalni protokol.

- Routing Information Protocol - RIP** usmerjanje z vektrojem razdalj (na 30s, 180s+ → prekinjena povezava), cena je število skokov, max cena je 15M
- Open Shortest Path First - OSPF** obvestila se s poplavljanjem posredujejo po celem sistemu, vsak potem preračuna najkrajše poti
- Interior Gateway Routing Protocol - IGRP** Cisco-va izboljšava RIP; cena je odvisna od pasovne širine, zakasnitve, obremenitve, MTU in zanesljivosti.

Za povezovanje AS mad seboj se uporablja **INTER-AS** usmerjevalni protokol, ki pa morabiti enak v celem omrežju.

- Border Gateway Protocol - BGP** omogoča, da omrežja oglašujejo svojo prisotnost drugim omrežjem

## Transportna plast

Naloge transportne plasti so:

- povezovanje dveh oddaljenih *procesov*
- multipleksiranje/demultipleksiranje komunikacije med procesi
- zanesljiv prenos podatkov
- kontrola pretoka in zasičenja

## Vrata - port

FTP-data: 20, FTP-cmd: 21, SSH: 22, Telnet: 23, SMTP: 25, DNS: 53, HTTP: 80, POP3: 110, IMAP: 143, IRC: 194, HTTPS: 443

## User Datagram Protocol: UDP

Nepovezavna storitev, nudi le best-effort: ni zagotavljanja vrstnega reda, ni nadzora zamašitev, ...

Dolžina glave je le 8B.

16b	16b
Izborna vrata	Ponorna vrata
Dolžina (skupaj z glavo) v B	Internetna kontrolna vsota

## Internetna kontrolna vsota

*Pošiljatelj*: pošiljatelj sešteje 16 bitne besede in shrani eniški komplement = kontrolna vsota
*Prejemnik*: sešteje 16 bitne besede skupaj s kontrolno vsoto → dobiti mora same enice

## Potrjevanje

**sprotno potrjevanje**: po vsakem pošiljanju počakaj na potrditev

**tekoče pošiljanje**: večji razpon za številčenje paketov, shranjevanje paketov na obeh straneh

- ponavljanje N potrjenih (*go-back-N*)
  - pošiljatelj hrani okno največ dovoljenih nepotrjenih paketov
  - ACK(*n*) potrdi vse pakete do vključno *n*
  - časovna kontrola za najstarejši paket, ko poteče pošlji vse nepotrjene pakete na novo
- ponavljanje izbranih (*selective repeat*)
  - prejemnik shranjuje sporočila in jih sortira pred dostavo
  - pošiljatelj ponovno pošlje le tiste pakete za katere ni prejel potrdila
  - vsak paket ima svojo časovno kontrolo

**neposredno**: uporaba ACK in NCK **posredno**: samo ACK, namesto NCK se ponovi ACK zadnjega segmenta

## Transfer Control Protocol: TCP

Povezavna storitev, full duplex, zanesljiv, kontrola pretoka, kontrola zasičenja, *tekoče pošiljanje in neposredno potrjevanje*

16b	16b
<i>Izvorna vrata</i>	<i>Ponorna vrata</i>
<i>Dolžina (skupaj z glavo) v B</i>	<i>Internetna kontrolna vsota</i>
<i>zaporedna št. (B)</i>	
<i>št. potrditve (B)</i>	
<i>Dolžina glave</i>	<i>0</i>
<i>zastavice</i>	<i>sprejemno okno</i>
<i>Internetna kontrolna vsota</i>	<i>karalec urg. data</i>
<i>opcije (spremenljiva dolžina)</i>	

Dolžina glave (4b) je podana v številu 32-bitnih besede

Sprejemno okno (16b) št. bajtov, ki jih sprejemnik lahko sprejme

Zastavice (9b):

- URG\*: urgentni podatki
- ACK: potrditev povezave
- PSH: takoj predaj aplikaciji
- RST, SYN, FIN: vzpostavljanje in rušenje povezave

### Vzpostavitev povezave - trojno rokovanje

- Odjemalec pošlje segment z zastavico SYN (sporoči začetno številko segmenta, ni podatkov)
- Strežnik vrne segment SYN ACK (rezervira medpomnilnik, odgovori z začetno številko svojega segmenta)
- Odjemalec vrne ACK, lahko že s podatki ("štuporama")

### Rušenje povezave

- odjemalec pošlje segment TCP FIN strežniku
- strežnik potrdi z ACK, zapre povezavo, pošlje FIN
- odjemalec prejme strežnikov FIN, potrdi ga z ACK (počaka časovni interval, da po potrebi ponovno pošlje ACK, če se ta izgubi)
- strežnik sprejme ACK, končano

### Nastavitev časovne kontrole

Na potrditev moramo čakati vsaj RTT (Round Trip Time).

$$\begin{aligned} \text{OcenRTT}[i] &= (1 - \alpha)\text{OcenRTT}[i - 1] + \alpha\text{IzmerRTT}[i] \\ \text{DevRTT}[i] &= (1 - \beta)\text{DevRTT}[i - 1] + \beta|\text{IzmerRTT}[i] - \text{OcenRTT}[i]| \\ \text{ČakalniInterval}[i] &= \text{OcenRTT}[i] + 4\text{DevRTT}[i] \end{aligned}$$

### Način potrjevanja

TCP uporablja podobno strategijo kot *ponavljanje N nepotrjenih*. Pošiljatelj ima časovno kontrolo le za najstarejši nepotrjen segment, vendar ob poteku časovne kontrole ne pošlje vseh segmentov v oknu, temveč le najstarejši nepotrjen segment.

RFC2018 vpeljuje potrjevanje le izbranih paketov.

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

### Hitro ponovno pošiljanje (fast retransmit)

Ponovno pošiljanje se običajno izvede po preteku časovne kontrole. Če pa pošiljatelj prejme 3 potrditve za že potrjen paket, takoj izvede ponovno pošiljanje.

### Kontrola pretoka

Usklajevanje hitrosti pošiljanja med pošiljateljem in prejemnikom.

Prejemnik sporoča razpoložljiv prostor v pomnilniku v polju 'sprejemno okno':

$$\text{rwnd} = \text{velikost} - [\text{lastByteRcvd} - \text{lastByteRead}]$$

### Oznake

rwnd	...	recieve window
cwnd	...	congestion window
MSS	...	maximum segment size
RTT	...	round trip time
ssthresh	...	slow start threshold

### Nadzor zasičenja

min(rwnd, cwnd) določa največ nepotrjenih B preden moramo ustaviti pošiljanje.

**TCP Tahoe:** osnovna verzija; 2 fazi: *počasen začetek* in *izogibanje zasičenju*; po izgubi paketa cwnd = 1 MSS.

**TCP Reno:** 3 faze: *počasen začetek*, *izogibanje zasičenju* in *hitra obnova*; če dobiš 3 kopije ACK že potrjenih podatkov, cwnd = cwnd/2 + 3 MSS.

**TCP Vegas:** dodano zaznavanje situacij, ki vodijo v zasičenje in *linearno zmanjševanje hitrosti* ob zasičenju.

Na začetku: cwnd = 1MSS; faza počasnega začetka; ssthresh = ∞.

- Faza počasnega začetka:**  
Za vsak prejet ACK: cwnd += min(N, MSS), kjer je N št. B, ki jih je potrdil ta ACK.  
*Alt. rešitev:* Za vsak prejet ACK: cwnd += MSS.  
*Na ta način se skozi čas cwnd povečuje eksponentno.*

- Če je cwnd ≥ ssthresh, preidemo v fazo izogibanja zasičenju.
- Če prejmemo 3 ACK, že potrjenega segmenta, gremo v *fazo hitre obnove* cwnd = ssthresh + 3MSS, ssthresh = cwnd / 2

- Faza izogibanja zasičenju:**  
Števemo B, ki jih potrdijo ACK. Ko to število preseže cwnd, resetiramo števec in cwnd += MSS.  
*Alt. rešitev:* Za vsak ACK, ki potrdi nove podatke povečamo cwnd+= MSS · MSS/cwnd  
*Na ta način se skozi čas cwnd povečuje linearno.*

- Če poteče časovna pontrola za najstarejši segment, gremo v *fazo počasnega začetka* in cwnd = MSS, ssthresh = cwnd / 2.
- Če prejmemo 3 ACK, že potrjenega segmenta, gremo v *fazo hitre obnove* cwnd = ssthresh + 3MSS, ssthresh = cwnd / 2

- Faza hitre obnove:**  
Za vsak podvojeni ACK: cwnd += MSS.
- Če dobimo ACK, ki potrjuje nove podatke, gremo v *fazo izogibanja zasičenju* in cwnd += ssthresh.
- Če preteče nek timeout, gremo v *fazo počasnega začetka* in cwnd = MSS, ssthresh = cwnd / 2.

TCP konvergira k pravični delitvi pasovne širine med uporabniki.

## Aplikacijska plast

## HTTP

### Glava

Metoda URL Verzija  
Ime polja 1: vrednost 1  
...  
Ime polja n: vrednost n  
*[prazna vrstica]*  
TELO

### Metode

GET	zahteva objekta
POST	zahteva objekta + poslane vrednosti (obrazci)
HEAD	zahteva, na katero strežnik odgovori z odgovorom brez zahtevanega objekta (uporabno za razhroščevanje)
PUT	(HTTP 1.1) – nalaganje na strežnik (upload)
DELETE	(HTTP 1.1) – brisanje s strežnika
TRACE	razhroščevanje (echo-odmev zahtevka, podobno PING)
CONNECT	povezava preko medstrežnika
OPTIONS	povpraševanje o možnih opcijah pri zahtevku

### Statusi

- 1xx: informativne kode**
- 2xx: uspešno**  
200: OK
- 3xx: preusmeritev**  
301: Moved Permanently – prestavljen dokument
- 4xx: napake pri odjemalcu**  
400: Bad Request – sintaksa  
404: Not Found – ni dokumenta
- 5xx: napake na strežniku**  
500: Internal Server Error  
505: HTTP Version Not Supported

### Vrste HTTP povezav

- nevztrajne (nonpersistent):** za vsak prenašani objekt (stran, sliko) se vzpostavi nova TCP povezava  
2 RTT/objekt (rokovanje + prenos)
- vztrajne (persistent):** strežnik uporabi isto povezavo za pošiljanje več objektov, strežnik pusti povezavo odprto  
1 RTT/objekt (prenos)
- vztrajne s cevovodom (persistent, pipelined):** tekoče pošiljanje več zahtev naenkrat, brezčakanja na prejem prejšnjih

### Medstrežnik

Pogojni zahtevek: If-modified-since: <datum>.

Strežnik pošlje nov stran ali pa HTTP/1.1 304 Not Modified