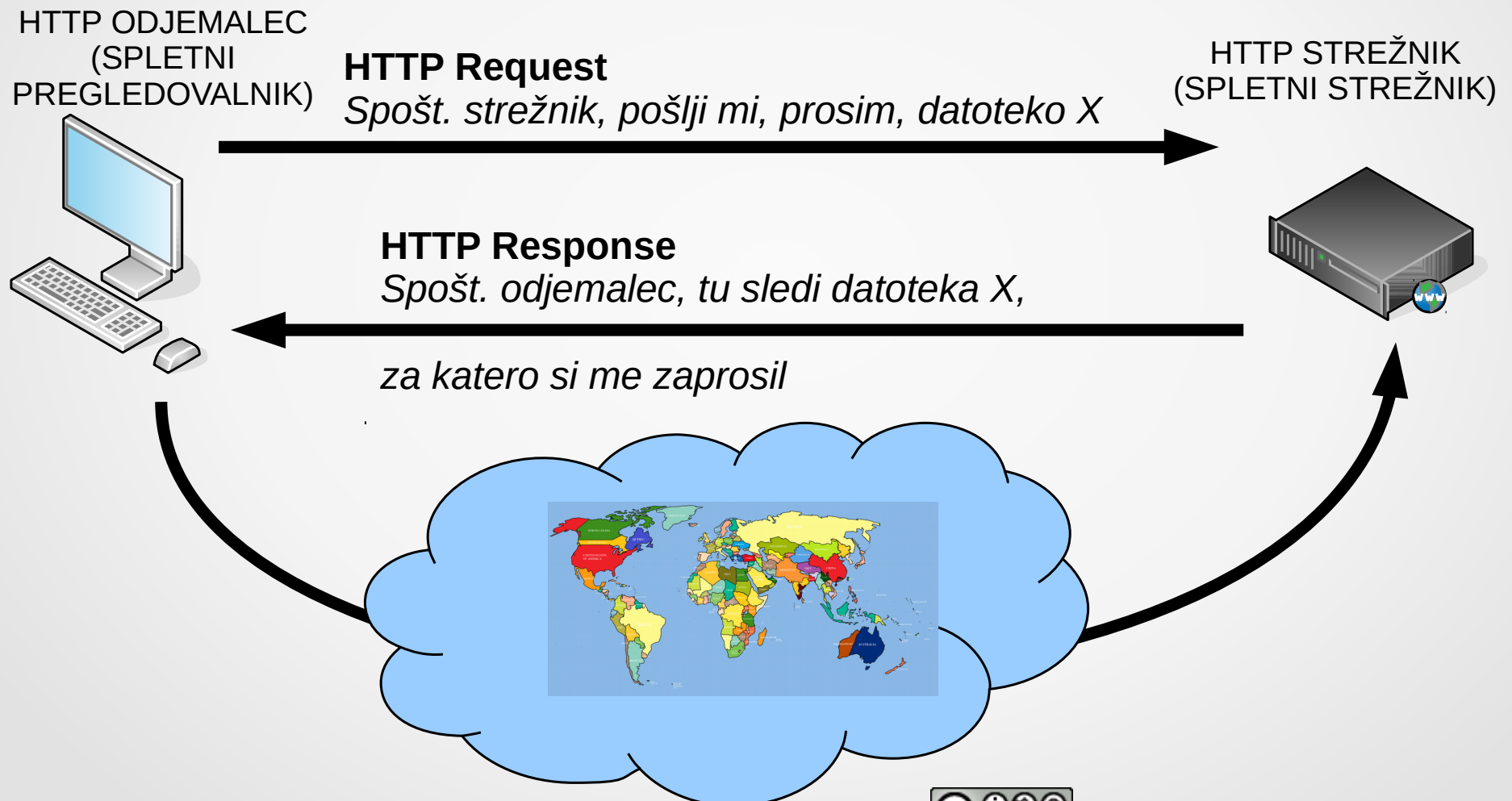


Svetovni Splet - World Wide Web

Izrazoslovje

- Spletni pregledovalnik / brskalnik = browser
- Splet = Web (samostalnik)
- spletni = web (pridevnik)
- Svetovni splet = World Wide Web

WWW - World Wide Web



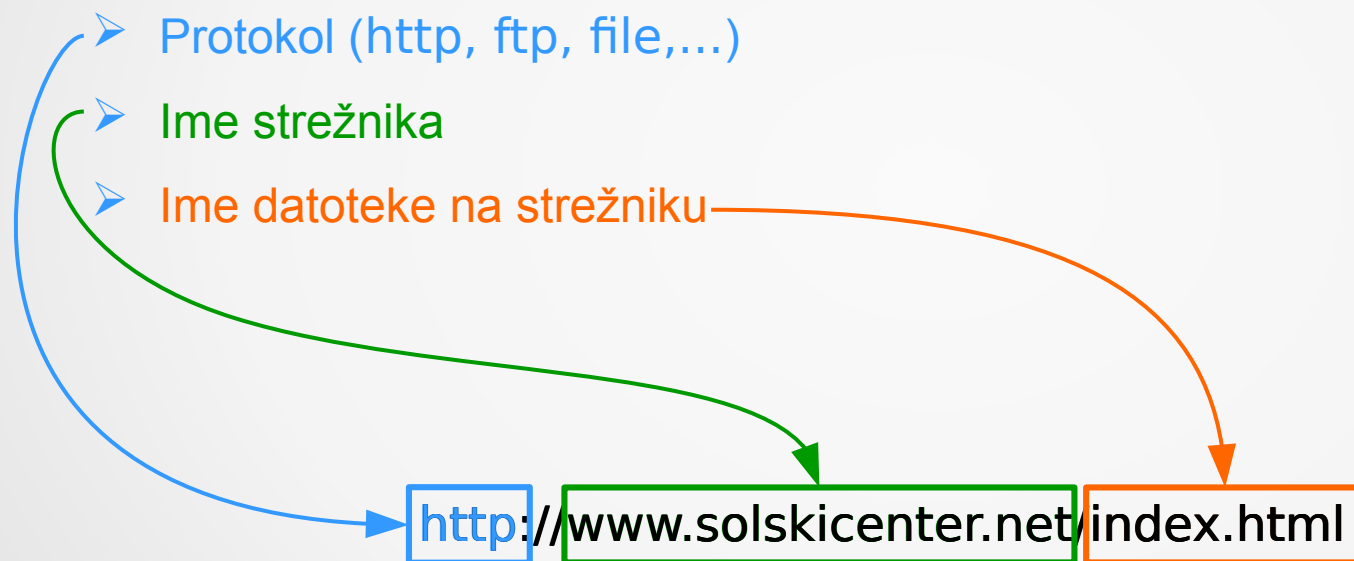
Author: Paolo Posillipo



URI

Uniform Resource Identifier (URI) - enolični identifikator vira:

Enolično določa (identificira) vir (naslov, besedilo, datoteko, storitev, naslov elektronske pošte, itd.) v svetovnem spletu:



URI, URL in URN (I)

- **Uniform Resource Identifier (URI) – enolični identifikator vira**
 - Niz znakov, ki enolično določajo (identificirajo) **vir** v svetovnem spletu. Ta vir je lahko naslov, besedilo, datoteka, storitev, naslov elektronske pošte, itd.
- **Uniform Resource locator (URL) – enolični krajevnik vira**
 - Niz znakov, ki enolično določajo (identificirajo) **naslov** vira (besedila, slike, video posnetka) v svetovnem spletu.
- **Uniform Resource Name (URN) – enolično ime vira**
 - Niz znakov, ki enolično določajo (identificirajo) **ime** vira znotraj istega imenskega prostora, tj. znotraj neke množice imen, ki so si med seboj različna.

URI, URL in URN (II)

- **Razlikovanje: PAZI!!!!**
- URN je URI, ki identificira vir, vendar, z razliko od URL-ja, ne dovoljuje identifikacije naslova samega vira. Na primer ISBN koda, ki enolično identificira knjigo, a nam ničesar pove, kje se ta knjiga nahaja.
-
- URL je URI. Tudi URN je URI.-
- URL je URI, ki ga imenujemo *spletni naslov*.
-
- Če uporabimo prisposodbo, je URN ime osebe (njeno istovetnost, identiteto), URL pa je naslov hiše, kjer se tista oseba nahaja, oz. način, kako osebo najti.
-
- **V pogovornem jeziku se namesto URI pogosto uporablja besedo URL (Uniform Resource Locator), ki naj bi bila sopomenka za URI. Tehnično pa to ni pravilno.**

WWW – Statične in dinamične strani

- Statična spletna mesta: strani so shranjene na strežniku v taki obliki, kot si jih bo uporabnik ogledal
- Dinamična spletna mesta: informacije se pogosto posodablajo in spreminjajo vsakič, ko je neka spletna stran zahtevana

WWW – Statične strani

HTTP ODJEMALEC
(SPLETNI
PREGLEDOVALNIK)



HTTP Request

1) *Spošt. strežnik, pošlji mi, prosim, datoteko X*



HTTP STREŽNIK
(SPLETNI STREŽNIK)



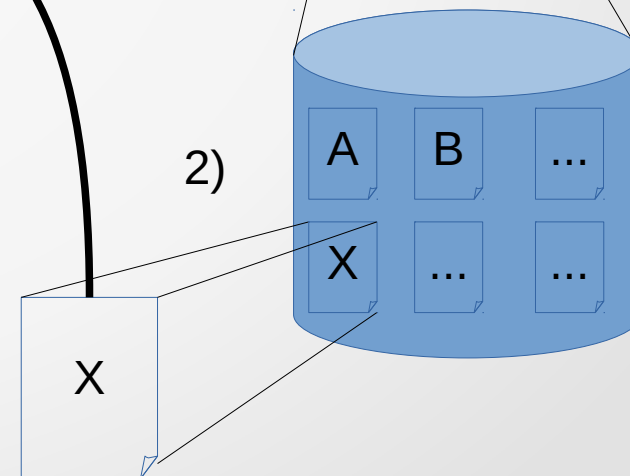
HTTP Response

*Spošt. odjemalec, tu sledi datoteka X,
za katero si me zaprosil*



3)

2)



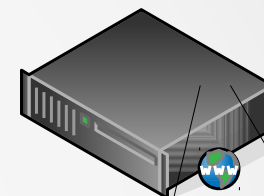
WWW – Dinamične strani

HTTP ODJEMALEC
(SPLETNI
PREGLEDOVALNIK)



HTTP Request
1) *Spošt. strežnik, sledijo parametri za sestavo strani X*

HTTP STREŽNIK
(SPLETNI STREŽNIK)



2)

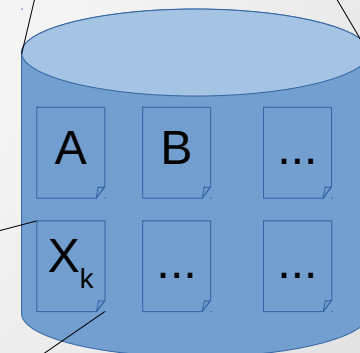
parametri

HTTP Response
4) *Spošt. odjemalec, tu sledi stran X, ki sem jo sestavil na podlagi tvojih parametrov*



3)

koda



HTTP - Uvod

- Skoraj vse, kar vidimo v brskalniku, se prenaša po spletu preko HTTP protokola. Npr., ko odpremo določeno spletno stran, brskalnik pošlje tudi nad 40 HTTP zahtevki in sprejme HTTP odgovore na vsakega od teh.
- HTTP (**H**yper**T**ext **T**ransfer **P**rotocol - Protokol za prenos hiperbesedil) je mrežni protokol, ki se uporablja za pošiljanje vseh virov po spletu in je prvotno namenjen objavljanju in prejemanju HTML strani, z značilno arhitekturo odjemalec – strežnik (client – server).
- Viri na spletu so vse datoteke in drugi podatki kot so HTM datoteke, slike, rezultati poizvedb ali drugo.
- Razvoj HTTP protokola upravlja WWW Consortium - W3C (konzorcij za svetovni splet <http://www.w3.org/>)

HTTP – Delovanje (I)

- HTTP temelji na mehanizmu zahtevkov/odgovor (client/server – odjemalec/strežnik): odjemalec pošlje strežniku zahtevek in strežnik mu odgovori. Navadno soupada odjemalec z *brskalnikom* (ki mu pravimo tudi *preglednik*), spletni strežnik pa s spletnim mestom (točneje, z aplikacijo, ki se izvaja na strežniku-računalniku). Potemtakem, obstajata dve vrsti HTTP sporočil: zahtevki in odgovori.
- HTTP se od drugih protokolov, kot je npr. FTP razlikuje po tem, da se *povezava zaključi, ko se z odgovorom zadosti zahtevku* (ali zaporedju zahtevkov, ki so nanj vezani). Zaradi takšnega delovanja, je HTTP protokol idealen za Svetovni Splet, saj večkrat spletne strani vsebujejo povezave (linke) na strani, ki so na drugih strežnikih in s tem mehanizmov se zmanjša število aktivnih povezav: dejansko je število omejeno na tiste povezave, ki so nujno potrebne in se tako poveča učinkovitost (manjša obremenitev) tako odjemalca kot strežnika.

HTTP – Delovanje (II)

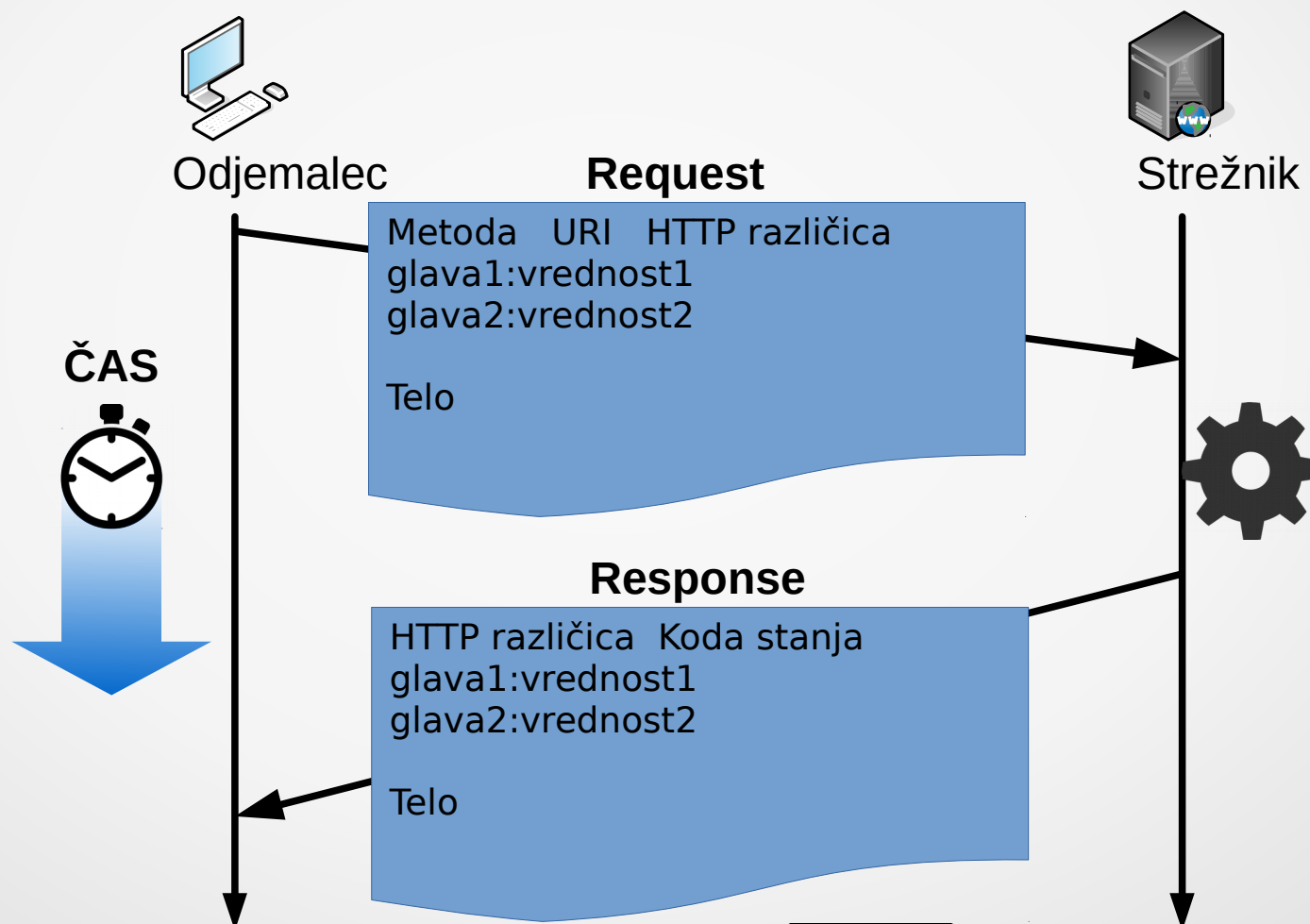
- Navadno odjemalec začne zahtevek tako, da vzpostavi TCP povezavo na izbranih vratih (ali priklupu) na oddaljenem gostitelju (privzeta je številka 80). HTTP strežnik, ki na tem priklupu pričakuje, da bo odjemalec poslal svoj zahtevek na primer

GET / HTTP/1.1

- (ta zahtevek prosi za privzeto spletno stran na strežniku), čemur sledi sporočilo, ki vsebuje vrsto obveznih in neobveznih podatkov za informiranje strežnika (podatek o gostitelju *Host* je na primer obvezen), čemur lahko sledi neobvezno polje poljubnih podatkov.
- Strežnik se nato odzove s sporočilom odgovora (HTTP Response), ki običajno vsebuje vir (vsebino), za katero je zahteval odjemalec.

HTTP – Delovanje (IIa)

- Vse HTTP povezave potekajo v obliki, kot je prikazano v sledeči sliki:



HTTP – Delovanje (III)

- HTTP je protokol brez stanja (*stateless*), v katerem odjemalec pošilja zahteve, strežnik na zahtevek odgovori in se povezava zaključi. Izraz *stateless* pomeni prav to, da **nobeden od udeležениh računalnikov ne ohranja informacij o določeni povezavi**: ko strežnik izpolni vse zahteve odjemalca, se povezava zaključi. Odjemalec začne povezavo kot sledi:
- [Še preden odjemalec vzpostavi HTTP povezavo, mora poznati strežnikov IP naslov, ki ga pridobi s pomočjo DNS-a. Šele nato vzpostavi TCP povezavo in:]
- 1) Najprej odjemalec pošlje strežniku *sporočilo zahtevka* na primernih vratih (priklopu): za protokol HTTP je privzeto število vrat **80**.

Omrežna Vrata

- (Omrežna) Vrata (port) v računalništvu pomenijo vmesnik, skozi katerega pošiljamo in prejemamo podatke.
- Vrata ima vsak računalnik v omrežju, ki lahko uporablja za komunikacijo protokola UDP in TCP. Razpon števila vrat je med 0 in 65535, kjer so vrata 0 rezervirana za lokalni računalnik, razpon vrat od 1 do 1023 pa imenujemo privilegirana vrata, ki jih v GNU/Linux sistemih in v vseh sistemih UNIX in podobnih sistemih lahko uporablja samo superuporabnik, navadni uporabniki, pa lahko uporabljajo razpon vrat od 1024 do 65535. To območje vrat pa imenujemo nepriviligirano območje vrat. Območja je predpisala IANA (Internet Assigned Numbers Authority - <http://www.iana.org/>). Na privilegiranem območju tako navadno poslušajo (pričakujejo vzpostavitev povezave) strežniki, kot so: FTP, TELNET, SSH, POP3, SMTP, HTTP, ...
- Tako na primer odjemalec na našem računalniku lahko uporabi za povezavo vrata iz nepriviligiranega področja in se poveže na strežnik na oddaljenem gostitelju, ki uporablja vrata iz privilegiranega območja. To je precej pomembno pri gradnji požarnega zidu, saj lahko bolj natančno omejimo posamezne povezave.
- Ker je TCP povezovalni protokol, se najprej vzpostavi povezava med odjemalcem in strežnikom. Pri povezavi je določen odjemalčev naslov IP in vrata, ter strežnikov naslov IP in vrata na katerih posluša strežnik. Naslov IP povezan z določenimi vrati tvori *vtičnico (socket)* in par odjemalčeve in strežnikove vtičnice tvori povezavo TCP, ki je edinstveno določena. Glava (header) paketa TCP vsebuje izvorni naslov IP in vrata, ciljni naslov IP in vrata, zaporedna številka paketa, številka potrditve in kontrolne zastavice. Npr.:



HTTP - Sporočilo zahtevka

- Sporočilo zahtevka je sestavljeno iz:
 - *vrstice zahtevka (request line)*;
 - *glav (headers)*: dodatne informacije
 - *teles (body)*: vsebina sporočila: lahko je datoteka, spletna stran ali drug vir
- Telo od ostalih dveh delov ločuje prazna vrstica, kot v sledeči shemi:

<vrstica zahtevka>

Glava1: vrednost1

Glava2: vrednost2

[Druge glave...]

<----[prazna vrstica]

<morebitno telo sporočila je tukaj: lahko gre za parametre; lahko je dolgo več vrstic, ali pa več bytov (binarnih) podatkov>

HTTP - Sporočilo zahtevka - Vrstica

- Prvi del zahtevka je vrstica zahtevka (request line), oz. del sporočila, v katerem imamo nekaj osnovnih informacij o zahtevku:
 - ukaz (ali metoda), kateremu sledita
 - identifikator vira (URI) in
 - različica HTTP protokola.
- Npr.: GET /index.html HTTP/1.0
- V zgornjem primeru smo uporabili metodo GET, da bi zahtevali za datoteko /index.html in smo izjavili, da uporabljamo različico 1.0 HTTP protokola. Čeprav se v HTTP zahtevkih najpogosteje uporablja metoda GET, HTTP podpira še druge metode, ki - dejansko - točneje opišejo namen povezave. Mi bomo obravnavali izključno metodi GET in POST, o katerih bomo podali nekaj primerov in obrazložili njihovo uporabo.

HTTP - Vrstica zahtevka

- Prvi del zahtevka je vrstica zahtevka (request line), oz. del sporočila, v katerem imamo nekaj osnovnih informacij o zahtevku:
 - ukaz (ali metoda), kateremu sledita
 - identifikator vira (URI) in
 - različica HTTP protokola.

GET /tutorials/other/top-20-mysql-best-practices/ HTTP/1.1

HTTP - Vrstica zahtevka - Metode

- Pri različici 1.1 imamo sledeče metode:
 - **GET** – najpogostejše uporabljena metoda za pridobitev vsebine datoteke.
 - **POST** - podobno kot **GET**, samo, da vsebuje telo zahtevka podatke, ki jih želimo sporočiti strežniku.
 - **HEAD** - enako kot **GET**, le da na to strežnik odgovori le z glavo datoteke. Uporabno za pridobivanje meta-oznak.
 - **PUT** - se uporablja za pošiljanje datotek na ciljni spletni strežnik.
 - **DELETE** - redko uporabljen; zbriše dotično datoteko z strežnika.
 - **TRACE** - vrne nazaj zahtevek, zato, da lahko odjemalec ugotovi, ali so vmes kaki dodatni strežniki, ki spreminjajo zahteve.
 - **OPTIONS** - vrne seznam HTTP zahtevkov, ki jih strežnik podpira. Zahtevek se uporablja takrat, ko želimo ugotoviti, ali spletni strežnik deluje.
 - **CONNECT** - se uporablja pri spletnih posrednikih za vzpostavitev SSL tunela (povezave).

HTTP - Vrstica zahtevka - Različica

- Prva različica protokola **HTTP/0.9**, sega proti koncu letih 80 prejšnjega stoletja in je, skupaj z jezikom HTML in z URL naslovi, jedro *World Wide Web (WWW) global information initiative*, ki jo je razvil Tim Berners-Lee na CERN-u v Ženevi za porazdelitev informacij med različnimi skupnostmi fizikov visokih energij. HTTP/0.9 Podpira samo GET ukaz in se ni nikoli uveljavil.
- Prvo dejansko uporabno različico protokola, **HTTP/1.0**, je uresničil sam Berners-Lee leta 1991 in jo predlagal kot RFC 1945 organizaciji IETF leta 1996. Ta različica je še vedno uporabljena, posebno s strani spletnih posrednikov.
- Z razširitvijo grafičnega brskalnika NCSA Mosaic, se je WWW močno razvil in kmalu so prišle na dan šibkosti različice HTTP/1.0, zlasti nezmožnost gostovanja številnih spletnih mest na istem strežniku (virtual host) in nezadostna zaščita varnostnih mehanizmov. Protokol so zaradi tega razširili z različico **HTTP/1.1**, ki so jo predložili kot *RFC 2068* leta 1997 in jo nadalje posodobili leta 1999. Poleg tega, omogoča HTTP/1.1 *trajne povezave* (znane tudi kot keep-alive povezave) z več kot enim zahtevkom z odgovorom na povezavo, za povečanje odzivnosti in učinkovitosti.
- **Internet Engineering Task Force (IETF)** je organizacija, ki standardizira Internetne protokole: ocenjuje kandidate za potencialne standarde in vodi proces potrditve standardov.
- **Request For Comment (RFC)** je zbirka dokumentov, katerih glavni namen je, da priskrbijo informacije ali opišejo delo v teku v zvezi z določenim standardom. Kdorkoli lahko pošlje predlog standarda (draft) RFC v oceno spletni skupnosti.

HTTP - Vrstica zahtevka – Glava (I)

- Po vrstici zahtevka, imamo v sporočilu neobvezne dele, ki jim pravimo glave (*headers*): to so vrstice, ki vsebujejo dodatne informacije o konfiguraciji odjemalca in o želenih lastnostih vira. Vsaka informacija je podana v svoji vrstici, in sicer v obliki:
- ime glave:vrednost
- Npr. odjemalec (brskalnik) lahko pošlje svoje ime in kodo različice ali želene lastnosti datoteke:

```
GET /tutorials/other/top-20-mysql-best-practices/ HTTP/1.1
Host: net.tutsplus.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.5)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: PHPSESSID=r2t5uvjq435r4q7ib3vtdjq120
Cache-Control: no-cache
```

HTTP - Vrstica zahtevka – Glava (II)

- Najpogostejši glavi sta:
 - **Host (Gostitelj):** ime strežnika, na katerega se nanaša URL. Je obvezna v zahtevkih, ki so v skladu z različico HTTP/1.1, ker omogoča uporabo virtualnih gostiteljev (virtual host), ki temeljijo na imenih.
 - **User-Agent:** identificira odjemalca: pregledovalnik, proizvajalec, različica, itd.

GET /tutorials/other/top-20-mysql-best-practices/ HTTP/1.1

Host: net.tutsplus.com

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.5)

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: en-us,en;q=0.5

Accept-Encoding: gzip,deflate

Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7

Keep-Alive: 300

Connection: keep-alive

Cookie: PHPSESSID=r2t5uvjq435r4q7ib3vtdjq120

Cache-Control: no-cache

HTTP - Sporočilo odgovora

- Sporočilo odgovora je sestavljeno iz:
 - *vrstice stanja (status line)*;
 - *glave (header)*;
 - *prazne vrstice*;
 - *telesa(body)* - vsebina odgovora: lahko je datoteka, spletna stran ali drug vir
- Podobno, kot pri sporočilu zahtevka, telo od ostalih dveh delov ločuje prazna vrstica, kot v sledeči shemi:

<vrstica stanja>

Glava1: vrednost1

Glava2: vrednost2

[Druge glave...]

<----[prazna vrstica]

<morebitno telo sporočila je tukaj: gre za vsebovano datoteko ali za podatke poizvedbe; lahko je dolgo več vrstic, ali pa več bytov (binarnih) podatkov>

HTTP - Sporočilo zahtevka - Vrstica Stanja

- Prvi del odgovora je vrstica stanja (status line), oz. del sporočila, v katerem imamo nekaj osnovnih informacij o odgovoru:
 - različica HTTP protokola in
 - *koda stanja (status code)*.

Npr.: HTTP/1.1 200 OK

- V zgornjem primeru odgovori strežnik odjemalcu, da je zahtev ek bil uspešen
- Vrstica stanja vsebuje kodo, ki predstavlja stanje vira, za katerega je zaprosil odjemalec. Prva šte vka kode predstavlja enega od petih razredov (vrst) odgovora (**xx** pomeni, da nadomestimo z dvema števkama):
 - **1xx** Informational - Informativne kode
 - **2xx** Success - Uspeh
 - **3xx** Redirection - Preusmeritev
 - **4xx** Client Error - Napaka odjemalca
 - **5xx** Server Error - Napaka strežnika

HTTP - Vrstica Stanja 1xx in 2xx

- **1xx Informational - Informativne kode:** Zahtevek prejet, nadaljuj obdelavo.
- **100 Continue:** Strežnik je prejel glavo (header) zahtevka in odjemalec mora poslati telo zahtevka (navadno s POST metodo).
- **2xx Success – Uspeh:** Strežnik je uspešno prejel, razumel in sprejel zahtevek, zahtevek je pravilen.
- **200 OK:** Standardni odgovor za uspešne HTTP zahtevke.
- **202 Accepted:** Zahtevek za obdelavo je bil sprejet, a obdelava se ni še zaključila.
- **204 No Content:** Strežnik je zadostil zahtevku, a v odgovoru ni nikakršnega telesa.

HTTP - Vrstica Stanja 3xx

- **3xx Redirection – Preusmeritev:** Odjemalec mora izvesti dodatna dejanja za zadostitev zahtevku.
- **301 Moved Permanently:** Ta in nadaljnje zahteve bo treba preusmeriti na drug URI (ki ga določamo v glavi Location).
- **302 Found:** Zahtevani vir se začasno nahaja na različnem URI-ju. Je najpogostejše uporabljena koda.
- **303 See Other** (od HTTP/1.1 dalje): Odgovor zahtevku najdeš na drugem URI-ju z metodo GET.
- **304 Not Modified:** Če odjemalec pošlje strežniku zahtevek za določen vir in zahtevek za isto stran ponovno pošlje v zelo kratkem času, mu strežnikov odgovor s to kodo sporoči, da se od zadnjega zahtevka vir ni spremenil.



HTTP - Vrstica Stanja 4xx

- **4xx Client Error - Napaka odjemalca:** Zahtevek ni pravilen ali strežnik ga ne more sprejeti.
- **400 Bad Request:** Strežnik ne zadovolji zahtevka zaradi sintaktičnih nepravilnosti.
- **401 Unauthorized:** Podobno kot 403/Forbidden, namenjen za uporabo, ko je avtentikacija možna, a ni uspela.
- **403 Forbidden:** Zahteveke je legitimen, a strežnik ga noče zadovoljiti. V nasprotju s kodo 401 Unauthorized, avtentikacija nima nikakršnega učinka.
- **404 Not Found:** Zahtevani vir ni bil najden, a bo v prihodnje mogoče na voljo.
- **405 Method Not Allowed:** Zahtevek vsebuje z nedovoljeno metodo. Npr. to se dogaja, kadar odjemalec napačno uporablja metodo GET za posredovanje podatki, za katere je dovoljena le metoda POST.
- **408 Request Timeout:** Rok za pošiljanje zahtevka se je iztekel in strežnik je zaključil povezavo.
- **410 Gone:** Označi, da vir ni več na voljo in ne bo na voljo niti v prihodnosti.

HTTP - Vrstica Stanja 5xx

- **5xx Server Error - Napaka strežnika:** Strežniku ni uspelo zadovoljiti zahtevku, čeprav je ta pravilen.
- **500 Internal Server Error:** Splošne napake, brez podrobnosti.
- **501 Not Implemented:** Strežnik ne more zadovoljiti metodi zahtevka.
- **503 Service Unavailable:** Nedosegljiva storitev; strežnik začasano ne more nuditi storitve. Gre le za začasno stanje.
- **505 HTTP Version Not Supported:** Strežnik ne podpira HTTP različice zahtevka.

HTTP - Sporočilo odgovora – Glava (I)

- Podobno kot pri vrstici zahtevka, imamo v sporočilu odgovora dele, ki jim pravimo glave (*headers*): to so vrstice, ki vsebujejo dodatne informacije o konfiguraciji strežnika in o lastnostih vira. Vsaka informacija je podana v svoji vrstici, in sicer v obliki:
- ime glave:vrednost
- Npr. strežnik lahko pošlje lastnosti datoteke:

HTTP/1.1 200 OK

Date: Mon, 23 May 2005 22:38:34 GMT

Server: Apache/1.3.27 (Unix) (Red-Hat/Linux)

Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT

Etag: »3f80f-1b6-3e1cb03b«

Accept-Ranges: bytes

Content-Length: 438

Connection: close

Content-Type: text/html; charset=UTF-8

Trajnost HTTP povezave

- V HTTP/0.9 in HTTP/1.0, odjemalec pošlje zahtevek strežniku in strežnik nanj odgovori, zatem se povezava zapre. HTTP/1.1 pa je podprta tudi *trajna povezava*. To omogoča odjemalcu, da pošlje dodatne zahteve, ne da bi bilo treba ponovno vzpostavljati povezavo. To tudi omogoča pošiljanje več zahtevkom pred sprejemom odgovora.
- Tovrstno metodo včasih imenujemo »pipelining« ali »keep alive«. Nekateri odjemalci in nekateri strežniki, ki trdijo da uporabljajo HTTP/1.0 lahko podpirajo tudi trajnost povezave in strežniki, ki podpirajo HTTP/1.1, imajo lahko izklopljeno podporo za le-to. Če strežnik v svojem odgovoru poda glavo »Connection: close«, to pomeni, da bo povezavo zaprl takoj po odgovoru na prvi zahtevek.
- Tako HTTP strežniki kot odjemalci lahko zaprejo povezavo kadarkoli iz kakršnega koli razloga. To naredi HTTP protokol idealen za spletno okolje, kjer se spletne strani enega strežnika pogosto povezujejo na spletne strani z drugih strežnikov.
- Zapiranje HTTP/1.1 povezave je lahko precej daljše opravilo (od 200 milisekund do nekaj sekund) kot zapiranje HTTP/1.0 povezave, ker prva navadno rabi trajajoče zapiranje (da se primerno obdelajo še tavajoči paketi še ne odgovorjenih zahtevkov) medtem, ko druga se lahko zapre takoj, ko je poln odgovor poslan.

Domain Name System (DNS)

Izrazoslovje (I)

- vozlišče = node (en) = nodo (it)
- gostitelj = host (en/it)
- strežnik = server (en/it)
- odjemalec = client (en/it)
- vrata = port number (en) = porta (it)
- zapis = record (en/it)

Definicije

- (spletno) ime = (spletni) naslov - zaporedje znakovnih nizov (črk in števil), ki jih ločuje pika (".");
- strežnik - računalnik ali program v izvajanju (proces) na njem, ki nudi neko storitev odjemalcu (pri DNS bomo uporabljali prvi pomen);
- odjemalec - računalnik ali program v izvajanju (proces) na njem, ki se poslužuje neke storitve strežnika (pri DNS bomo uporabljali prvi pomen)
- domena (SSKJ)
 - 1. področje ustvarjanja, delovanja;
 - 2. *zgod.* državna ali vladarska posest, zlasti v starem in srednjem veku.

Naslovi in imena (I)

- dostop do Internetnega gostitelja preko **IP naslova** je (za ljudi) zahtevno delo
- zlasti, ker si je treba vedno zapomniti zaporedje števil (ki jih ločuje "."), in za ljudi to ni "naravno"
- zato so si izmislili **(spletna) imena**

www.microsoft.com
mail.google.com
ftp.cs.cornell.edu

*zaporedje nizov,
ki jih ločuje "."*

načeloma velja, da

- vsak gostitelj **ima** en IP naslov
- vsak gostitelj **lahko ima** ime
- gostitelja lahko identificiramo
 - bodisi preko IP naslova
 - bodisi preko njegovega imena

Naslovi in imena – DNS (I)

- **DNS (Domain Name System) – Sistem Domenskih Imen**
 - infrastruktura, ki upravlja imena
 - protokol za sporazumevanje s strežniki infrastrukture
- **DNS je hierarhično strukturiran sistem za razreševanje imen (pretvarjanje imen v IP naslov). Domenski prostor lahko prikažemo kot drevesno strukturo, ki se deli na domene (področja delovanja).**

enostaven primer...

- **imenski strežnik, NS strežnik (name server, NS server) hrani vse dvojice (vrstice)**
- **< Ime, IP naslov > v tabeli**

Naslovi in imena (III)

- vsak gostitelj pozna IP naslov imenskega strežnika
- odjemalec pošlje strežniku **zahtevo (request)**, v kateri je ime (spletnega mesta)
- nato strežnik odgovori z **odgovorom (response)**, ki vsebuje IP naslov

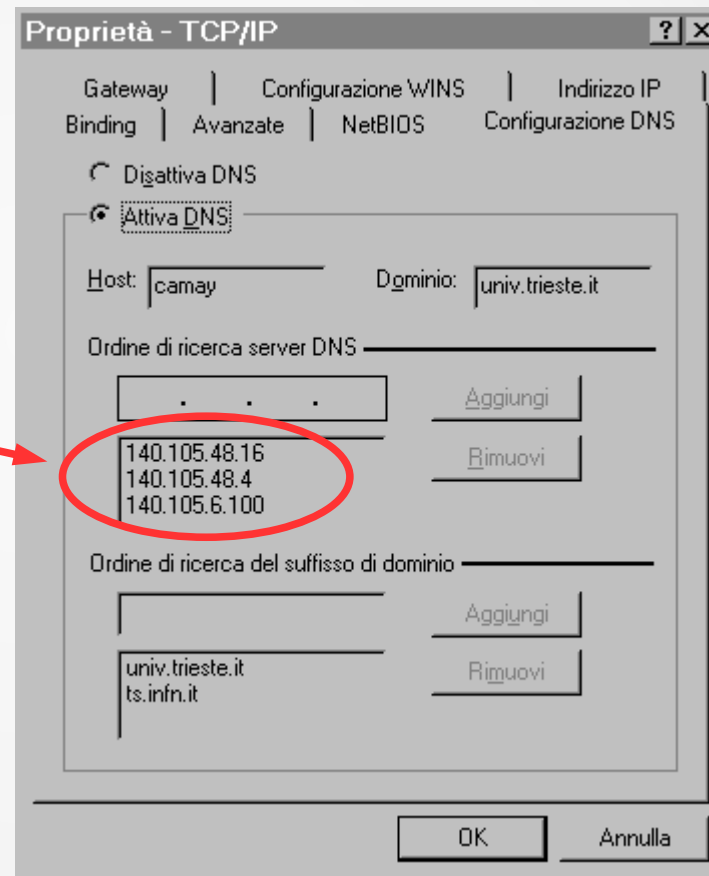
IME	NASLOV
www.microsoft.com	131.114.9.252
mail.google.com	192.113.21.27
ftp.cs.cornell.edu	115.116.123.11
...	...



Naslovi in imena (IV)

- vsak gostitelj mora poznati IP naslov imenskega strežnika

- IP naslov
- (vrata 53)



Naslovi in imena (V)

- Ime lahko identificira **gostitelja** ali **neko drugo stvar** (bomo videli kasneje)
- shema tabele je < IME, VRSTA (ZAPISA), VREDNOST > ,
- oz. v ang < NAME, TYPE, VALUE >
- doslej smo videli le vrsto A (Address - Naslov), ki označi spletne naslove

IME	VRSTA	VREDNOST
mail.google.com	A	192.113.21.27
ftp.cs.cornell.edu	A	115.116.123.11
...
...

ime gostitelja



IP naslov

Resource Record

- Podatki v DNS strežniku so organizirani v tako imenovanih **zapisih virov** - **Resource Record (RR)**, to so vrstice tabele.
- request:
 - Ime (Name)
 - Vrsta (Type)
- response:
 - Zapis (RR)



OPOMBA

- gostitelj **lahko nima** imena
- gostitelj **lahko ima** ime
- gostitelj lahko ima **dve ali več imen** (več zapisov vrste A z različnimi Imeni, a z isto Vrednostjo)
- na splošno, ima vozlišče **strežnik** (najmanj) eno ime

(zelo) poseben primer:

- Ime je vezano na več gostiteljev (več RR vrste A z istim Imenom in različnimi vrednostmi)

Type CNAME

- vrsta CNAME od DNS strežnika zahteva t.i. *kanonično ime* oz. *alias* (drugo ime) nekega kanoničnega imena:
 - pravo, **kanonično, ime** je v stolpcu VALUE,
 - **alias** pa v stolpcu NAME

NAME	TYPE	VALUE
mail.google.com	CNAME	googlemail.l.google.com
...
...

Type MX

- vrsta MX od DNS strežnika zahteva ime strežnika elektronske pošte:

NAME	TYPE	VALUE
google.com	MX	alt4.aspmx.l.google.com
hotmail.com	MX	mx3.hotmail.com
...

ime
“domene e-pošte”



ime strežnika (gostitelja) e-pošte,
ki pripada tisti “domeni e-pošte”

sporočila, naslovljena naslovu
je treba posredovati gostitelju

ivan.novak@gmail.com
alt4.aspmx.l.google.com

sporočila, naslovljena naslovu
je treba posredovati gostitelju

ivan.novak@hotmail.com
mx3.hotmail.com

Opomba

- Iz **samega imena** ni mogoče določiti, ali je tisto ime vezano na gostitelja, ali pa na nekaj drugega (bomo videli kasneje)
- Vsako domensko ime ima eno ali več zapisnih virov, kateri so različnih vrst, za shranjevanje različnih vrst podatkov.

Struktura DNS

- DNS je urejen v hierarhični drevesni strukturi, ki jo tvorijo vozlišča.
- Ta predstavljajo računalnike in njihova imena:
 - na vrhu imamo **korensko domeno** (root domain), ki jo imenujemo s praznim znakom “”, v besedilu pa jo zapišemo s piko “.”
 - *končna vozlišča*, **listi**, pa predstavljajo večinoma računalnike uporabnikov Interneta
 - nekatera vozlišča pa so **strežniki domenskih imen**: le-ti hranijo v svojem pomnilniku segment (tj. del) celotne strukture (tabele) DNS
- ponekod najdemo izraze kot
 - “domenska imena”
 - “domena”

} področje delovanja (imenskega strežnika)

URI, URL in URN (I)

- **Uniform Resource Identifier (URI) – enolični identifikator vira**
 - Niz znakov, ki enolično določajo (identificirajo) **vir** v svetovnem spletu. Ta vir je lahko naslov, besedilo, datoteka, storitev, naslov elektronske pošte, itd.
- **Uniform Resource locator (URL) – enolični krajevnik vira**
 - Niz znakov, ki enolično določajo (identificirajo) **naslov** vira (besedila, slike, video posnetka) v svetovnem spletu.
- **Uniform Resource Name (URN) – enolično ime vira**
 - Niz znakov, ki enolično določajo (identificirajo) **ime** vira znotraj istega imenskega prostora, tj. znotraj neke množice imen, ki so si med seboj različna.

URI, URL in URN (II)

- **Razlikovanje: PAZI!!!!**
- URN je URI, ki identificira vir, vendar, z razliko od URL-ja, ne dovoljuje identifikacije naslova samega vira. Na primer ISBN koda, ki enolično identificira knjigo, a nam ničesar pove, kje se ta knjiga nahaja.
- URL je URI. Tudi URN je URI-
- URL je URI, ki ga imenujemo *spletni naslov*.
- Če uporabimo prisposodbo, je URN ime osebe (njena istovetnost, identiteta), URL pa je naslov hiše, kjer se tista oseba nahaja, oz. način, kako osebo najti.
- **V pogovornem jeziku se namesto URI pogosto uporablja besedo URL (Uniform Resource Locator), ki naj bi bila sopomenka za URI. Tehnično pa to ni pravilno.**

Struktura DNS – Domene

- **Domena** je enolično ime za določen in zelo omejen del Interneta, npr. za spletno stran. Uporabniki vidijo domene v sledeči obliki:
 - **www.example.com**
- Kot bistveni del **URL**-ja, predstavlja domena **lokacijo** nekega vira znotraj DNS-ja.
- **Pretvorbo domen (imen) v IP naslove** opravljajo **imenski strežniki (Name Servers - NS)**. Gre za posebne namenske spletne strežnike, ki se ukvarjajo s takoimenovanim razreševanjem imen v IP naslove.
- Ta storitev deluje podobno kot navadna telefonska storitev: uporabnik vnese domeno **www.example.com** v naslovno vrstico brskalnika in ta pošlje zahtevek imenskemu strežniku, ki je za to domeno odgovoren.
- V imenskem strežniku se izvaja asociacija spletno ime - IP naslov, nato strežnik našemu brskalniku odgovori z IP naslovom odgovarjajočega imena.

Struktura domene (I)

- Celotno ime domene imenujemo **popolnoma kvalificirano domensko ime - Fully Qualified Domain Name (FQDN)**. FQDN označi točno pozicijo ciljnega računalnika v drevesni hierarhiji DNS-ja in je sestavljeno iz dveh delov: ime računalnika (host name, ime gostitelja) in ime domene. Sledeči primer kaže (izmišljeno) FQDN strežnika elektronske pošte:

mailserver.example.com

- V tem primeru:
 - **mailserver** predstavlja **ime gostitelja**,
 - **example.com** pa je **ime domene**, pod katero se nahaja uporabljen računalnik.
- Za ime gostitelja spletnih storitev običajno uporabljamo niz **www**:
 - **www.example.com**

Struktura domene (II)

- Treba je pripomniti, da se, nasprotno od IP naslova, FQDN običajno vedno zaključi s piko; to se dogaja zaradi hierarhične strukture DNS-ja, na podlagi katere, domene razdelimo začnši od najvišje stopnje, ti. **korenske oznake (root-label)**, iz desne proti levi:

domena 3. stopnje	domena 2. stopnje	domena 1. stopnje	korenska oznaka
www	example	com	(prazno)

- Iz desne proti levi imamo:
 - **korensko oznako – prazno**,
 - **domeno 1. stopnje (TLD - Top Level Domain) – com**,
 - **domeno 2. stopnje (SLD - Second Level Domain) – example** in
 - **domeno 3. stopnje - www**.
- Potemtakem, vsebuje FQND imena www.example.com štiri elemente. Načeloma lahko imajo domene druge **poddomene** pod domeno 3. stopnje.

Struktura domene – Korenska oznaka

- Prva stopnja korena DNS drevesa se imenuje **korenska oznaka (root label)** ali **ničelna oznaka (null label)**. Korensko oznako FQDN-ja opredelimo kot prazno in se običajno izpušča v aplikacijah na Internetu. Nasprotno, vrstice imenskih strežnikov, ti. Zapisi Virov (RR - Resource Records) morajo vedno imeti popoln FQDN s piko na koncu po domeni prve stopnje:

mailserver.example.com.

Struktura domene – Korenski strežniki (I)

- **Korenski strežnik (root nameserver ali root server)** je DNS strežnik, ki odgovarja zahtevkom za razreševanje imen v zvezi z imenskim prostorom korenske domene (root domain). Njihova naloga je ta, da zahteve, v zvezi z vsako vrhovno domeno (TLD), preusmerijo imenskim strežnikom tiste TLD.
- Korenski strežniki vsebujejo informacijo, ki predstavlja korensko cono, ki je svetovni seznam vrhnjih domen. Korenska cona vključuje omenjene vrhnje domene.
- Za vsako od vrhnjih domen, vsebuje korenska cona IP naslove imenskih strežnikov, ki upravljajo vsebine vrhnjih domen in korenski strežniki odgovarjajo s temi naslovi, ko sprejmejo zahtevek glede neke vrhnje domene.
- Na svetu imamo več kot 1300 korenskih strežnikov, ki so dosegljivi preko 13 IP naslovov - enega za vsako mednarodno organizacijo, ki jih upravlja, razen za Verisign, ki upravlja dva korenska strežnika.

Struktura domene – Korenski strežniki (II)

- Zakaj imamo le 13 DNS Korenskih Imenskih Strežnikov?
- Ko je nastala ta infrastruktura, so izbrali število 13 kot kompromis med zanesljivostjo omrežja in performansi, poleg tega je število 13 vezano na omejitev protokola Internet Protocol (IP), različica 4 (IPv4).
- Čeprav obstaja le 13 DNS korenskih strežnikov (v resnici gre za 13 naslovov) za IPv4, vsako od teh imen ne predstavlja v resnici en sam računalnik, pač pa gručo, ki jo sestavlja več računalnikov. Ta strategija omogoča zanesljivost DNS brez negativnih učinkov na performance. Poleg tega, onemogočajo, da npr. množični napadi tipa DDoS - Distributed Denial of Service - okrnejo sistem.
- Seznam korenskih strežnikov:
 - <https://root-servers.org/>

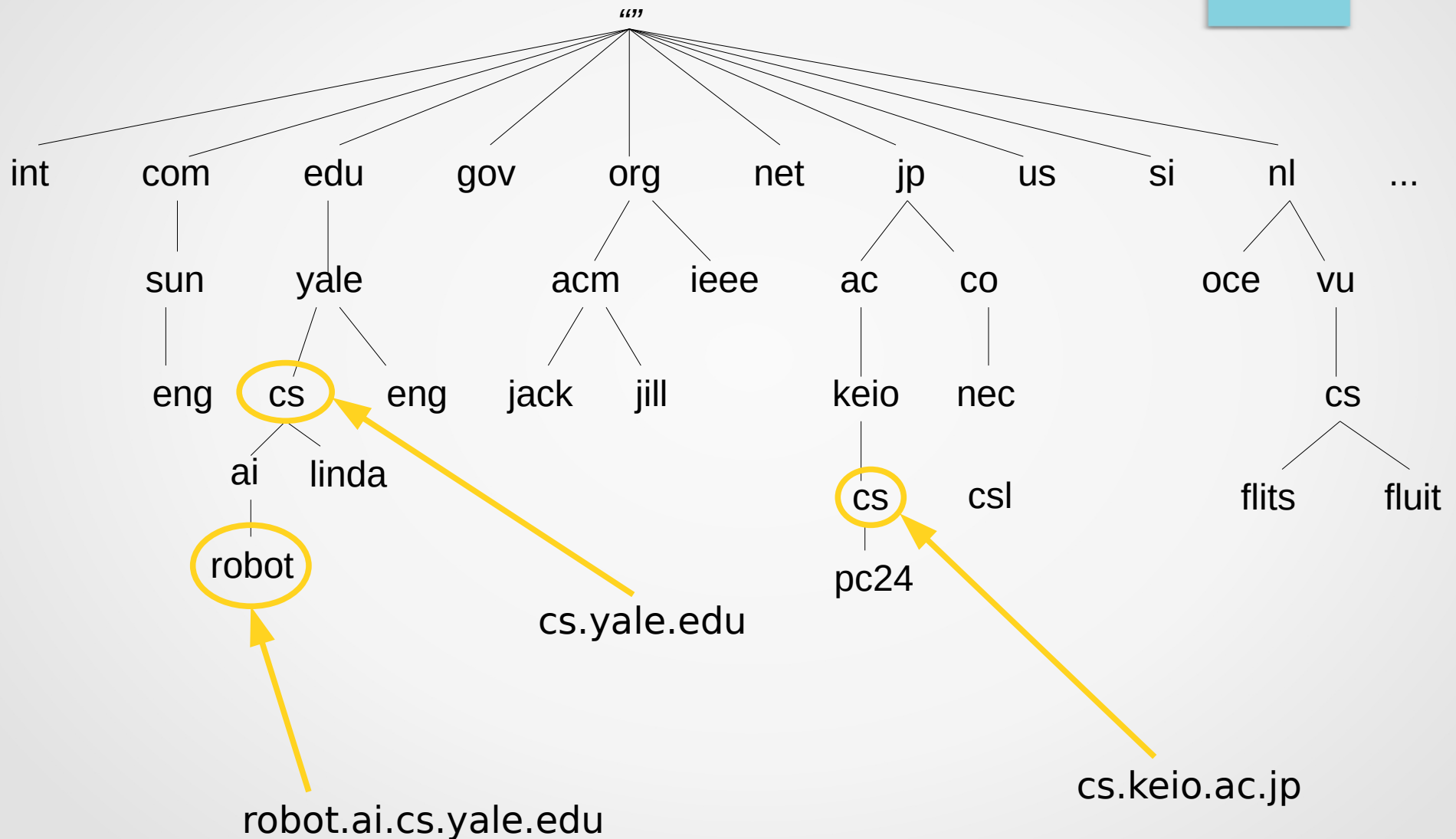
Struktura domene – Vrhnje domene (TLD) (I)

- Ker glavno - tj. korensko - domeno (root domain) opredelimo kot prazno, predstavljajo domene prve stopnje **najvišjo stopnjo razreševanja imen**.
- **Domene prve stopnje ali vrhnje domene** (ang. **Top-Level Domains - TLD**) upravlja ti. **Network Information Center (NIC)**. Med nalogami NIC-a imamo upravljanje imenskih strežnikov in dodelitev domen druge stopnje. **IANA (Internet Assigned Numbers Authority)**, oddelek **ICANN**-a (ameriška ustanova, ki se ukvarja z upravljanjem internetnega omrežja) ločuje dve skupini domen prve stopnje:
 - **generične vrhnje domene (generic top-level domain - gTLD)** in
 - **nacionalne vrhnje domene (country code top-level domain - ccTLD)** za države, kot npr. .de .at, .si, itd.

Struktura domene – Vrhnje domene (TLD) (II)

- Med generične sodijo domenske končnice splošne narave, kot so **.com**, **.net**, **.biz**, **.org**, itd. Običajno gre za tričrkovne vrhnje domene.
- Nacionalne pa so uporabljajo dvočrkovno kodo, določeno s standardom **ISO 3166**, in predstavljajo vrhno domeno posamezne države.
- Nekaterne nacionalne domene lahko registrirajo le rezidenti držav, druge pa so za registracijo na voljo vsem. Tudi pri generičnih velja, da jih večinoma lahko registrira kdorkoli, omejitve obstajajo le pri nekaterih.
- Cena registracije vrhnje domene je velikokrat odvisna od same domene.
- Seznam vrhnjih domen:
 - https://en.wikipedia.org/wiki/List_of_Internet_top-level_domains

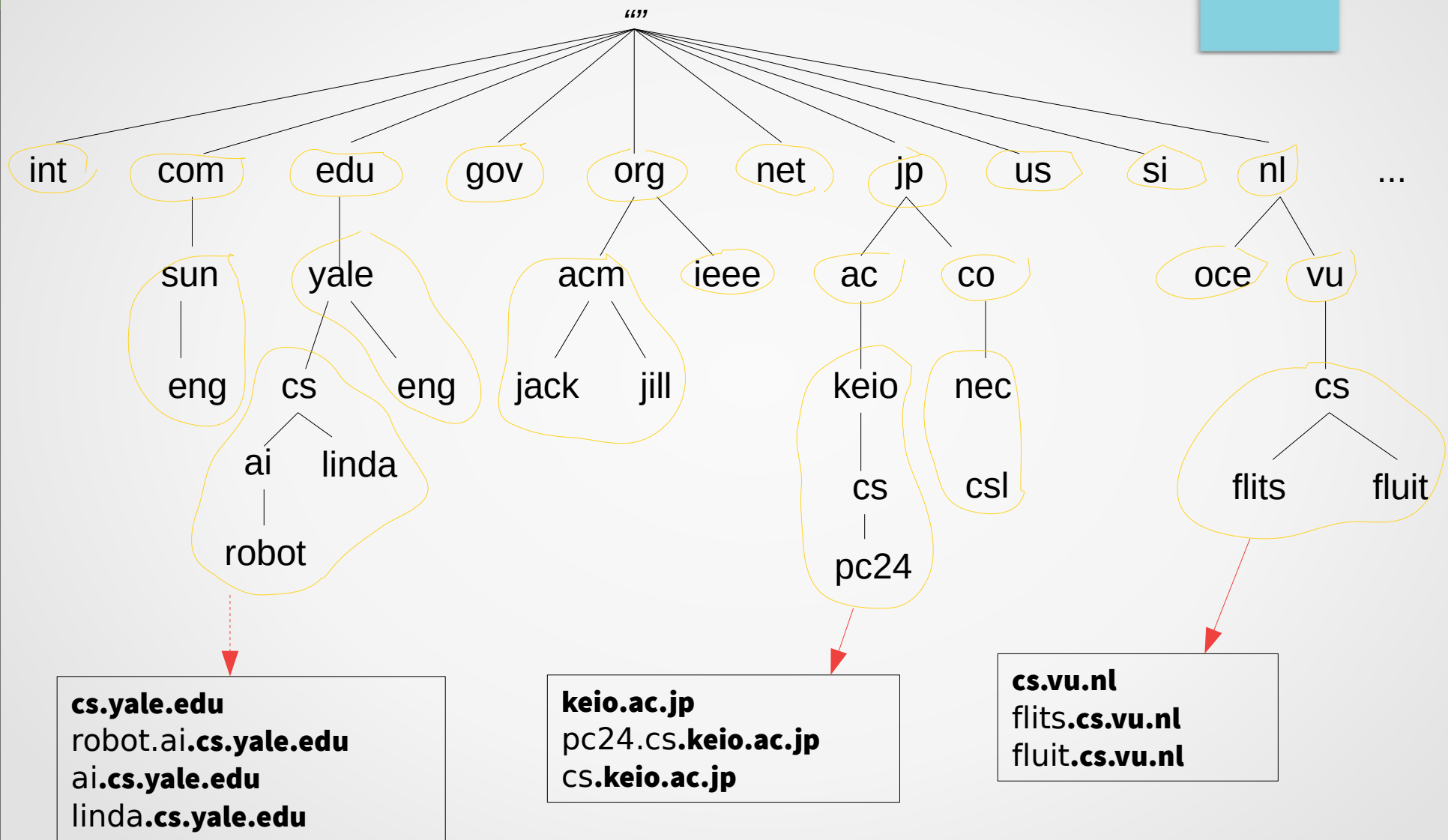
Struktura DNS



Struktura DNS - Cone (I)

- drevesno strukturo razdelimo na podstrukture (poddrevesa) ali **cone**
 - imena v isti coni imajo isto *pripono*
 - skupno pripono imenujemo **ime cone**

Struktura DNS - Cone (I-a)



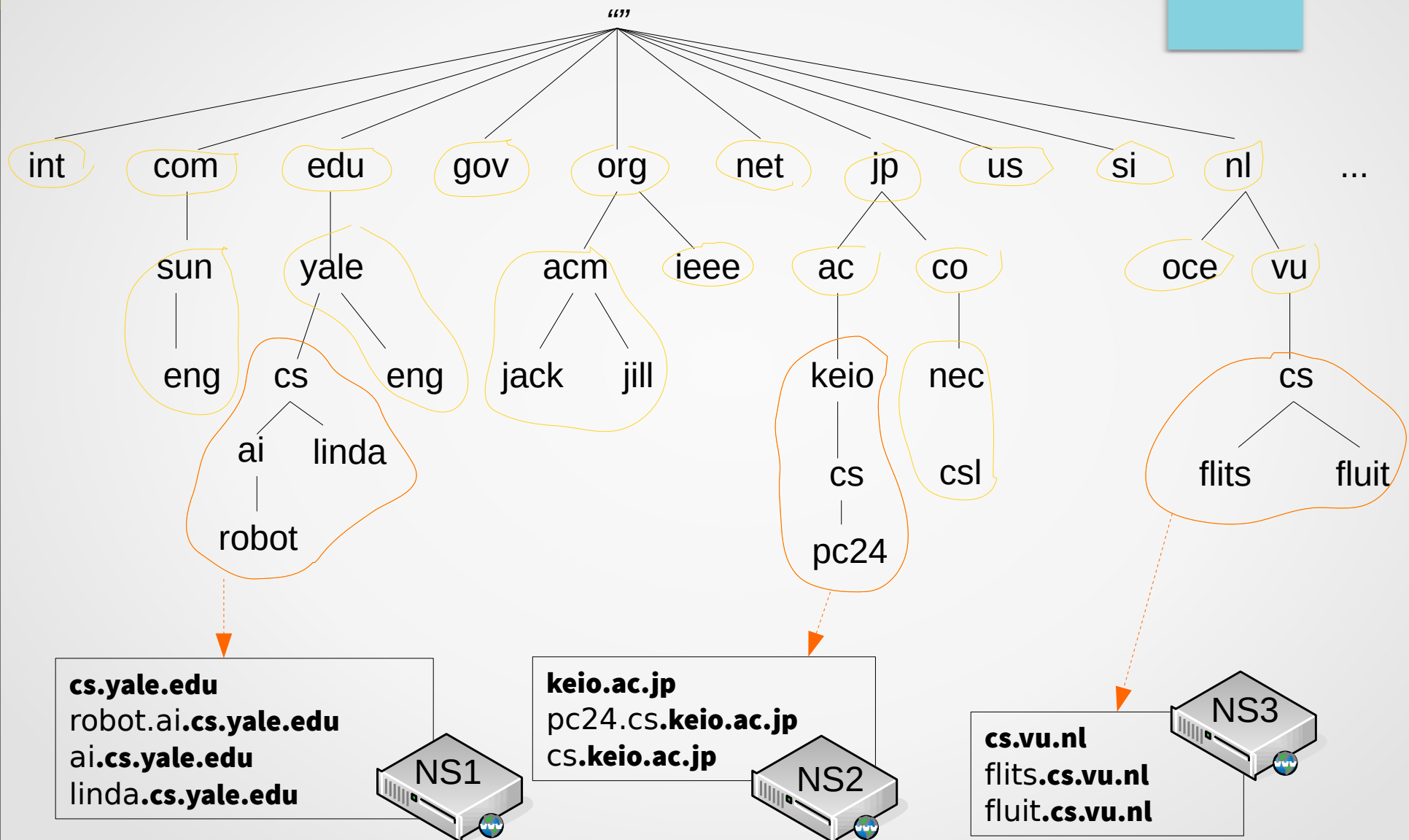
Struktura DNS - Cone (II)

- iz samega imena **ni mogoče** določiti:
 - kateri coni pripada
 - ali gre za ime cone
- primer: robot.ai.cs.yale.edu : to ime lahko pripada kateri koli izmed sledečih con:
 - edu
 - yale.edu
 - cs.yale.edu
 - ai.cs.yale.edu
 - robot.ai.cs.yale.edu
 - gre za ime cone

Struktura DNS – Cone (III)

- vsaka cona ima **lastnega** imenskega strežnika
- vsak imenski strežnik hrani **izključno** RR svoje cone (tabela je razdeljena po vrsticah)

Struktura DNS - Cone (III-a)



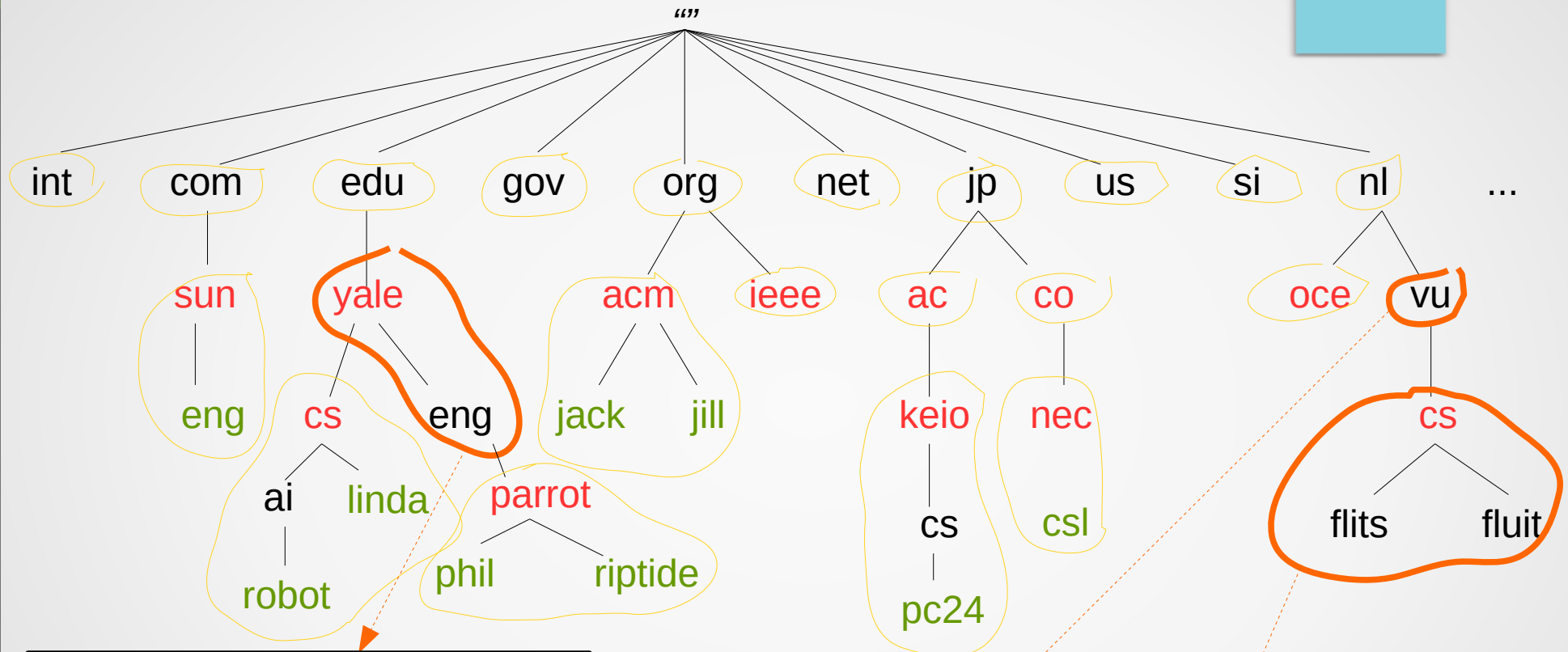
Author: Paolo Posillipo



Struktura DNS - Cone (IV)

- Vsak imensi strežnik mora poznati:
 - Ime lastne cone
 - **Imena** vseh **nadrejenih** in vseh **podrejenih** con
 - **IP naslove** imenskih strežnikov vseh nadrejenih in vseh podrejenih con

Struktura DNS - Cone (IV-a)



Jaz sem **yale.edu**

GOR: edu IP-ns

DOL: **cs.yale.edu** IP-ns

DOL: parrot.yale.edu IP-ns

Jaz sem **vu.nl**

GOR: nl IP-ns

DOL: cs.vu.nl IP-ns

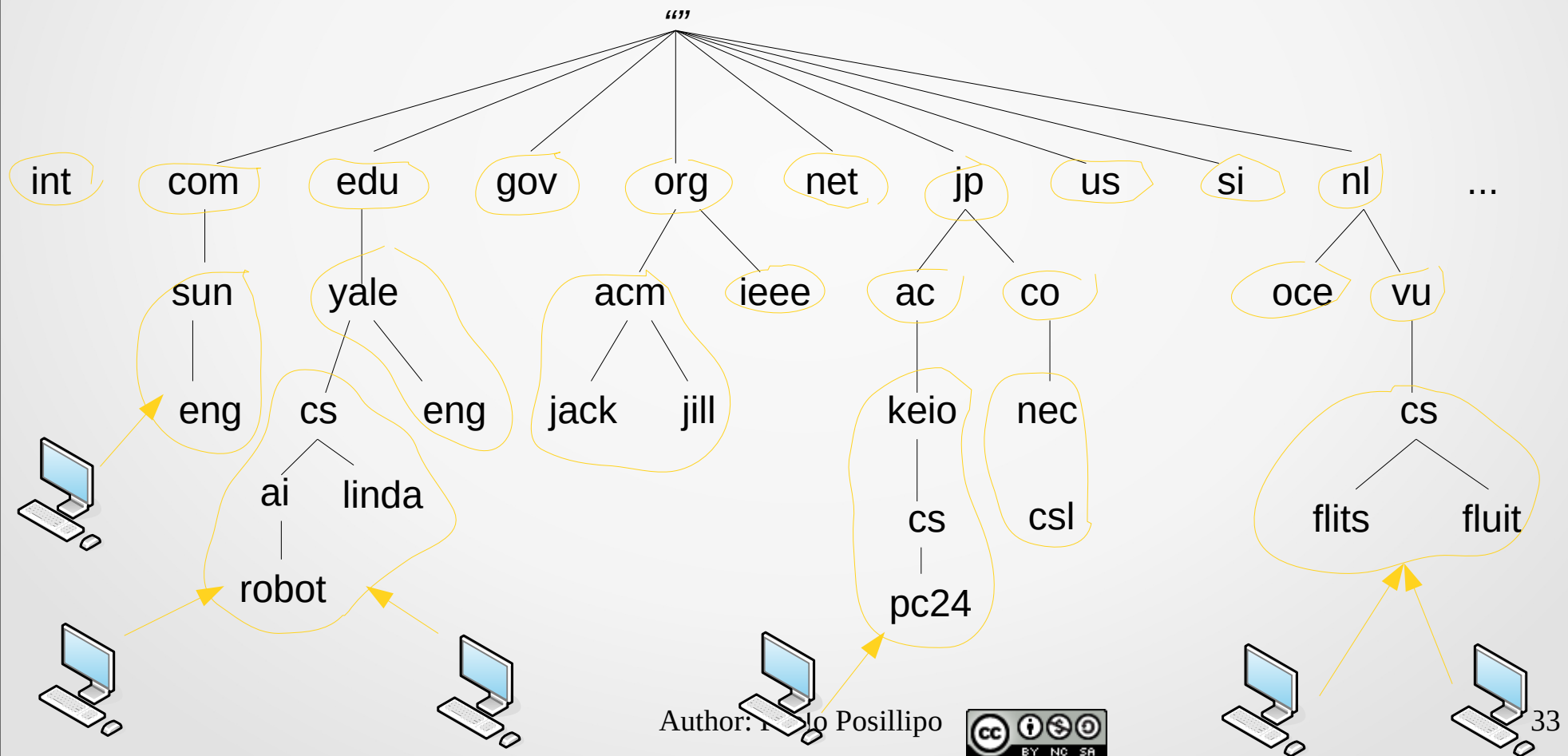
Jaz sem **cs.vu.nl**

GOR: vu.nl IP-ns

DOL: none

Struktura DNS - Cone (V)

- Za razreševanje imen, se odjemalec lahko obrne na imenski strežnik katere koli cone
- Običajno je tako nastavljen, da se obrne na najbližji imenski strežnik

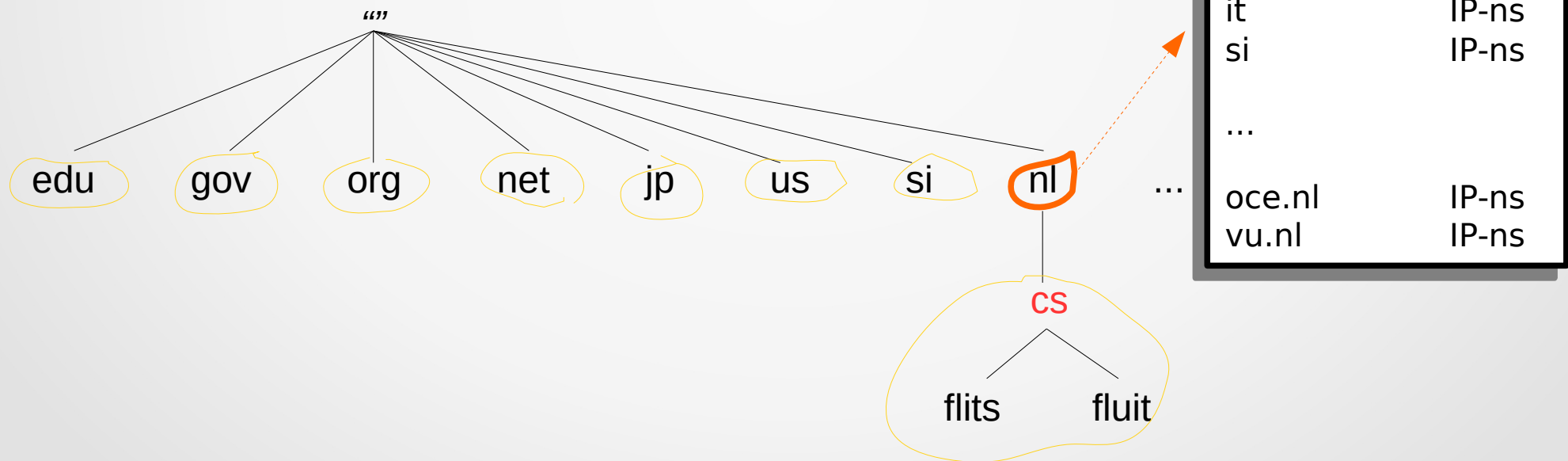




Struktura DNS - Cone (VII)

Vrhnje domene – Top-level domains (TLD): cone, ki nimajo nadrejenih con

- Imenski strežnik vrhnje domene mora poznati tudi:
 - **Imena** vseh TLD strežnikov
 - **IP naslove** vseh TLD strežnikov



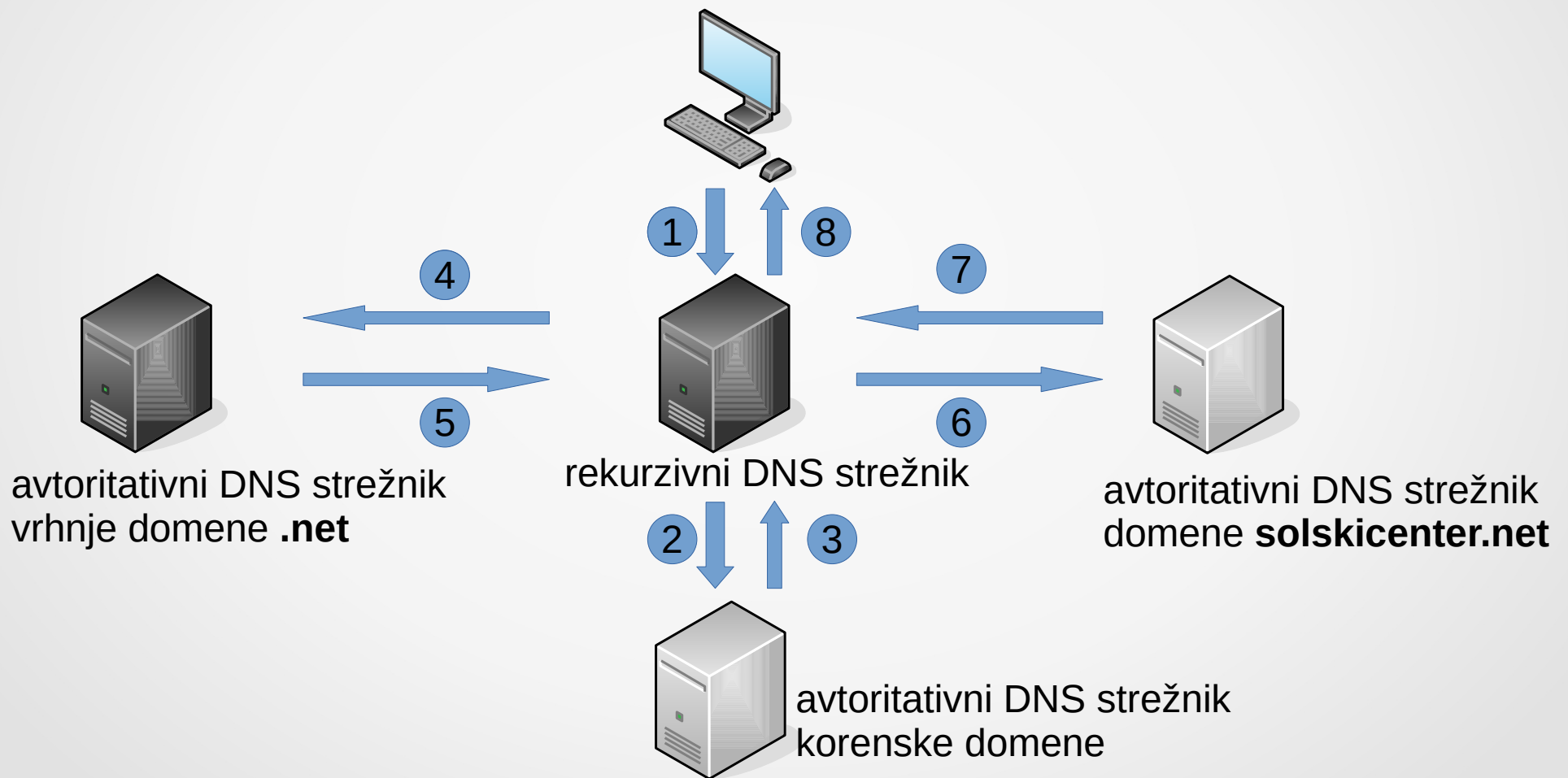
Razreševanje imen (I)

- Na splošno poznamo dve vrsti DNS strežnikov:
 - **avtoritativni strežnik** in
 - **rekurzivni strežnik**
- Prvi ima podatke o domeni, drugi pa lahko te podatke poišče. Za vsako domeno naj bi bilo več avtoritativnih strežnikov. Avtoritativni strežniki nastopajo v dveh vlogah:
 - **gospodar cone (ang. Zone Master) ali Primarni DNS conski strežnik** = domenski strežnik, ki ima osnovno kopijo podatkov o domeni, in
 - **suženj cone (ang. Zone Slave) ali Sekundarni DNS conski strežnik** = domenski strežnik, ki ima kopijo podatkov o domeni (kopijo pridobi iz primarnega strežnika).
- Priporočljivo je, da poleg primarnega strežnika obstaja vsaj še en sekundarni strežnik, ki je po možnosti na neki drugi lokaciji.
- Na svetu je na tisočine rekurzivnih DNS strežnikov. Večina ljudi uporablja rekurzivne DNS strežnike, ki jih upravlja **Ponudnik Internetnih Storitv - Internet Service Provider (ISP)** in jih nikoli ne zamenjajo.

Razreševanje imen (II)

- Ko v spletni brskalnik vnesemo na primer `www.solskicenter.net`, se izvedejo sledeče DNS poizvedbe
- 1. Računalnik v rekurzivni DNS strežnik ISP-ja, nastavljen za internetno povezavo, pošlje poizvedbo o ustreznih zapisih za `www.solskicenter.net`
- 2. Rekurzivni strežnik posreduje zahtevek avtoritativnim strežnikom za korensko domeno `.` (root)
- 3. Strežniki `.` (root) vrnejo spisek strežnikov DNS za `.net`
- 4. Rekurzivni strežnik nato posreduje zahtevek avtoritativnim strežnikom za `.net`
- 5. Avtoritativni strežniki za `.net` vrnejo rekurzivnemu strežniku odgovor avtoritativnih strežnikov za `solskicenter.net`
- 6. rekurzivni strežnik nato posreduje zahtevek avtoritativnim strežnikom za `solskicenter.net`
- 7. ker avtoritativni strežniki za `solskicenter.net` imajo odgovor na našo poizvedbo, vrnejo rekurzivnemu strežniku ustrezne podatke (`www.solskicenter.net` je alias za `solskicenter.net`, ki ima IP-naslov `194.249.2.4`)
- 8. rekurzivni strežnik računalniku posreduje odgovor, ki ga je prejel (`solskicenter.net` ima IP-naslov `194.249.2.4`)

Razreševanje imen (III)



DNS Cache (I)

- Pri razreševanju imen, rekurzivni strežnik, ki se nahaja v coni Z lahko izve za RR cone Z1, ki ni njegova cona.
- Shrani vse pridobljene zapise v cache pomnilniku – predpomnilniku
- Sprejme zahtevek za ime /:
 - Če je / v coni Z odgovori, sicer pa
 - **Če je / v predpomnilniku odgovori**, sicer pa
 - Posreduje zahteveke drugemu imenskemu strežniku
- Učinkovitejši odgovor
 - odgovor dobimo prej
 - breme je porazdeljeno na več strežnikov
- Upa se, da je pravkar pridobljeni zapis še veljaven

DNS Cache (II)

- V resnici imamo več (stopenj) DNS predpomnilnikov, tj. Pomnilnikov, v katerih so shranjeni zapisi virov:
 - v brskalniku,
 - v operacijskem sistemu
 - v rekurzivnem DNS strežniku
- Za brskalnik Chrome lahko izbrišemo vsebino DNS predpomnilnika tako, da v vrstico naslova vnesemo besedilo **chrome://net-internals/#dns** in kliknemo na gumb **Clear host DNS**
- Za operacijski sistem Windows, lahko izbrišemo vsebino DNS predpomnilnika tako, da v ukazni vrstici (command prompt) vnesemo besedilo **ipconfig /flushdns** in pritisnemo na **Enter**

Time To Live

- Vsak RR zapis ima polje (stolpec) **Time To Live - TTL**
- Naravno število med 0 in 86400
- TTL označuje, **koliko časa so podatki v bazi DNS veljavni**. Ta čas se uporablja tudi pri predpomnjenju domen v različnih DNS strežnikih, operacijskih sistemih in brskalnikih. Po preteku časa (TTL) se predpomnilnik obnovi.

Name	Time To Live	Value
mail.google.com	86400	192.113.21.27
ftp.cs.cornell.edu	86400	115.116.123.11
...
...