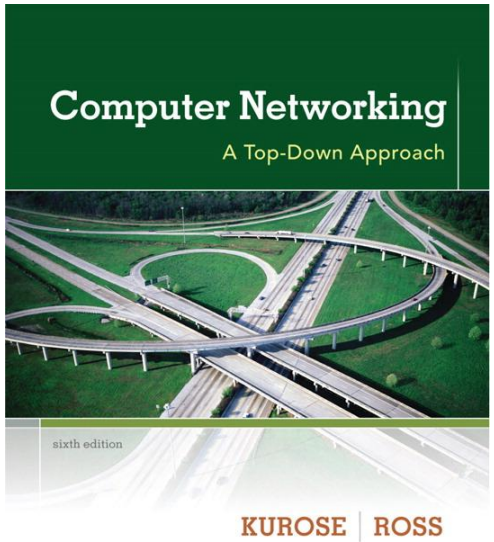
Poglavlje 2
Aplikacijski sloj



Computer Networking: A Top Down Approach
6th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2012

Predavanja prate gradivo knjige *Computer Networking: A Top Down Approach*, 6th edition, Jim Kurose, Keith Ross Addison-Wesley, March 2012

Poglavlje 2: sadržaj

2.1 principi na kojima se zasnivaju mrežne aplikacije

2.2 Web i HTTP

2.4 elektronička pošta

- SMTP, POP3, IMAP

2.3* Prijenos datoteka

2.5 DNS

2.6 P2P aplikacije

2.7 socket programiranje s UDP-om i TCP-om

Poglavlje 2: aplikacijski sloj

ciljevi:

- konceptualni i implementacijski aspekti aplikacijskih protokola
  - odabir modela usluga transportnog sloja
  - paradigma klijent-poslužitelj
  - paradigma ravnopravnih čvorova (P2P)
- učiti o protokolima na primjerima popularnih aplikacijskih protokola
  - HTTP
  - FTP
  - SMTP / POP3 / IMAP
  - DNS
- izrada mrežnih aplikacija
  - socket API

Neke mrežne aplikacije

- e-pošta
- web
- tekstualne poruke
- remote login
- P2P dijeljenje datoteka
- višekorisničke mrežne igre
- strujanje pohranjenog videa (YouTube, Hulu, Netflix)
- Internet telefonija (npr. Skype)
- video konferencije u realnom vremenu
- društvene mreže
- tražilice
- …
- …

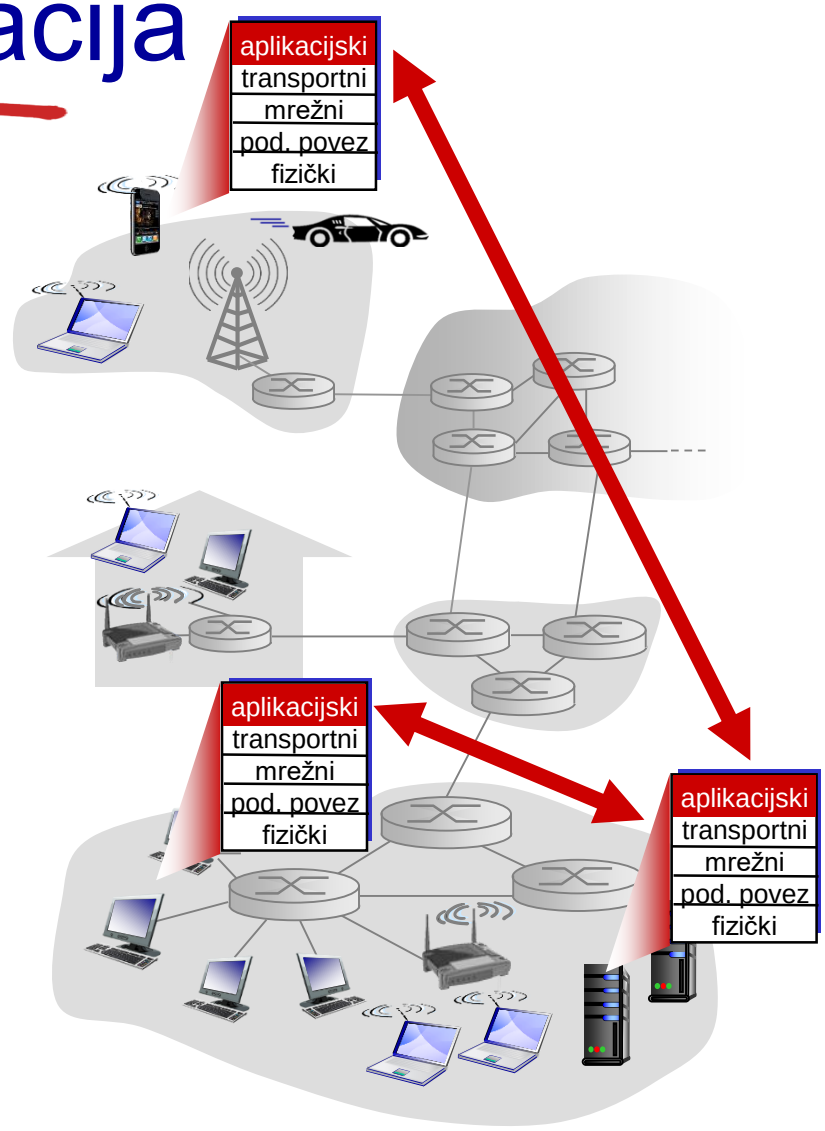
Izrada mrežnih aplikacija

izrada programa koji:

- se izvode na (različitim) *krajnjim sustavima*
- komuniciraju preko mreže
- npr. program web poslužitelja koji komunicira s programom web preglednika

nije potrebno pisati programe za uređaje u jezgri mreže

- na uređajima mrežne jezgre se ne izvode korisničke aplikacije
- izvođenje aplikacija na krajnjim sustavima omogućuje brzi razvoj aplikacija

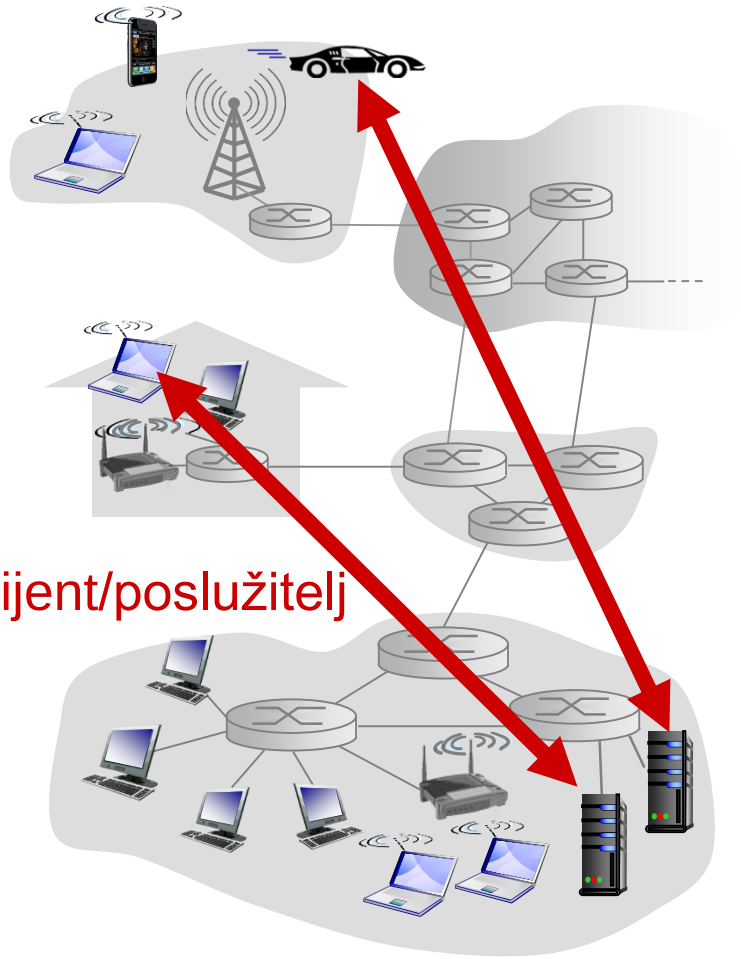


Arhitekture aplikacija

moguće strukture aplikacija:

- klijent-poslužitelj (*client-server*)
- ravnopravni-čvorovi (*peer-to-peer*, P2P)

Arhitektura klijent-poslužitelj



poslužitelj:

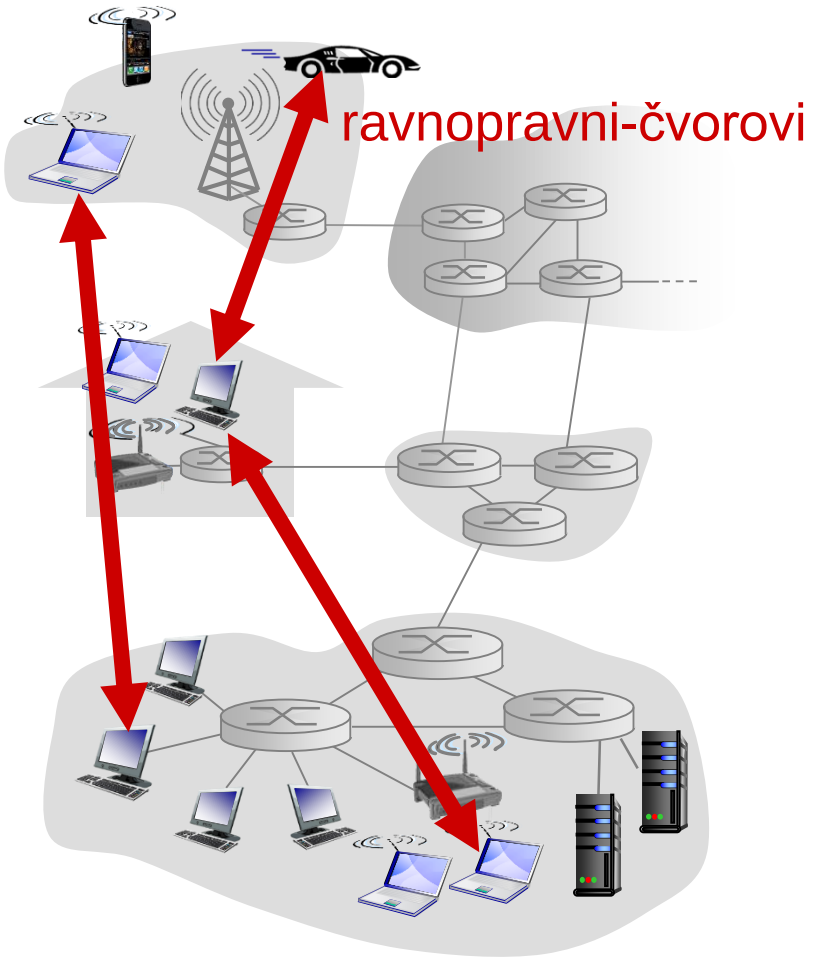
- uvijek uključen
- trajna IP-adresa
- problem skaliranja
  - podatkovni centri

klijent:

- komunicira s poslužiteljem
- može se spajati po potrebi
- može imati dinamički promjenljivu adresu IP-a
- klijenti međusobno ne komuniciraju izravno

P2P arhitektura

- *nema* stalno uključenih poslužitelja
- proizvoljni *krajnji sustavi* komuniciraju izravno
- čvor traži uslugu od drugog čvora, a također i pruža uslugu drugim čvorovima
  - *jednostavno skaliranje* – novi čvorovi povećavaju kapacitet usluge, a također i potražnjom za uslugama
- ravnopravni čvorovi se povremeno povezuju, a mijenjaju im se i IP-adrese
  - složeno upravljanje



Procesna komunikacija

proces: program koji se izvodi na računalu

- unutar istog računala, dva procesa komuniciraju koristeći međuprocesnu komunikaciju (omogućuje OS)
- procesi na različitim računalima komuniciraju razmjenjujući poruke

klijent i poslužitelj

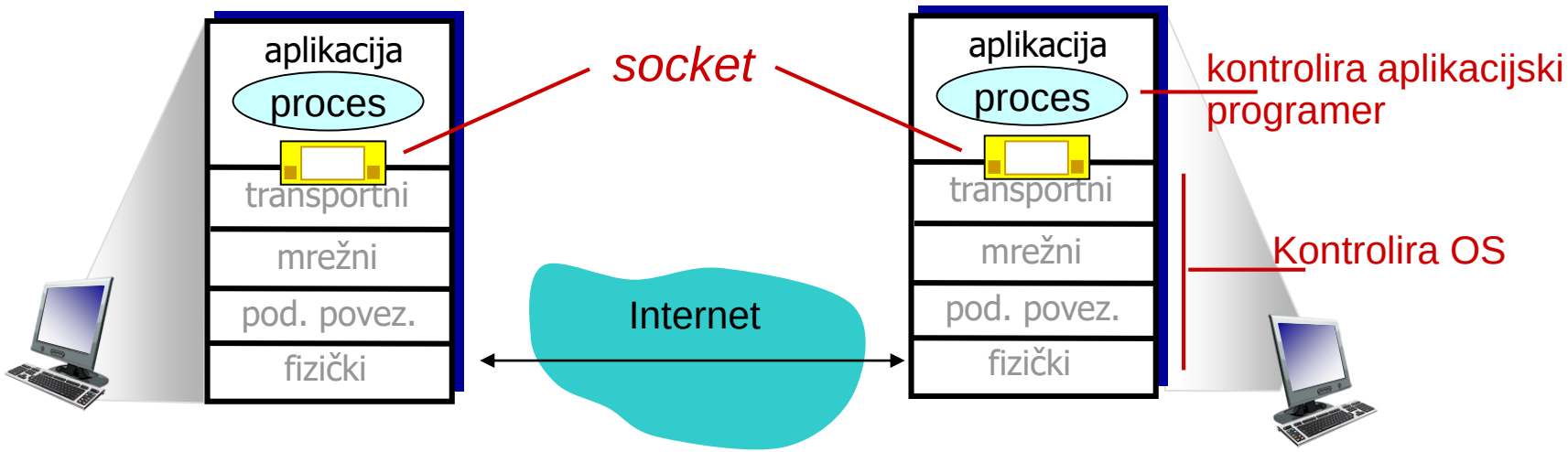
klijentski proces: proces koji započinje komunikaciju

poslužiteljski proces: proces koji čeka na zahtjeve klijenta

- usput: aplikacije u arhitekturi ravnopravnih čvorova (P2P) imaju i klijentske procese i poslužiteljske procese

Socketi

- proces šalje/prima poruke u/od svog socketa
- socketi su analogni vratima
  - proces šalje tako da poruci „pokaže” izlazna vrata
  - proces koji šalje poruku se oslanja na transportnu infrastrukturu, da ona dostavi poruku socketu procesa primatelja



Adresiranje procesa

- da bi primio poruku, proces mora imati *identifikator*
- računala imaju jedinstvene 32-bitne IP-adrese
- *P:* je li IP-adresa računala na kojem se izvodi proces dovoljna za identifikaciju tog procesa?
  - *O:* ne, *puno* se procesa može izvoditi na istom računalu
- *identifikator* uključuje i IP-adresu i broj porta pridružen procesu na računalu.
- Primjeri brojeva portova:
  - HTTP-ov poslužitelj: 80
  - poslužitelj e-pošte: 25
- za slanje HTTP-poruke web-poslužitelju gaia.cs.umass.edu:
  - IP-adresa: 128.119.245.12
  - broj porta: 80
- više o tome uskoro…

Protokoli aplikacijskog sloja definiraju:

- tipove poruka
  - npr. zahtjev, odgovor
- sintaksu poruke:
  - koja polja u poruci
  - kako se polja razdvajaju
- semantika poruke
  - značenje informacija u poljima
- pravila
  - kada i kako procesi šalju i primaju poruke

otvoreni protokoli:

- definirani RFCovima
- omogućuju interoperabilnost
- npr. HTTP, SMTP

vlasnički protokoli:

- npr. Skype

Koje transportne usluge trebaju aplikacije?

pouzdan prijenos

- neke aplikacije (npr. prijenos datoteka, web transakcije) zahtijevaju pouzdan prijenos podataka
- neke aplikacije (npr. audio) mogu tolerirati manje gubitke

kašnjenje

- neke aplikacije (npr. Internet telefonija, interaktivne igre) zahtijevaju malo kašnjenje da bi bile "upotrebljive"

propusnost

- neke aplikacije (npr. multimedija) zahtijevaju minimalnu propusnost da bi bile "upotrebljive"
- druge aplikacije ("elastične aplikacije") koriste onoliku propusnost koja im je na raspolaganju

sigurnost

- kriptiranje, integritet podataka …

Zahtjevi za transportnim uslugama: primjeri

aplikacija	gubici podataka	propusnost	osjetljiva na kašnjenje
prijenos datoteka	bez gubitaka	elastična	ne
e-pošta	bez gubitaka	elastična	ne
Web dokumenti	bez gubitaka	elastična	ne
multimedija u realnom vremenu	prihvatljivi	audio: 5kb/s-1Mb/s video:10kb/s-5Mb/s	do100-tinjak ms
strujanje pohranjene multimedije	prihvatljivi	-||-	do nekoliko sek.
interaktivne igre	prihvatljivi	nekoliko kb/s ili više	do 100-tinjak ms
tekstualne poruke	bez gubitaka	elastična	da i ne
automatizacija	bez gubitaka	nekoliko kb/s	često stroge granice (npr. nekoliko ms)

Usluge protokola transportnog sloja na Internetu

usluge TCP-a:

- *pouzdan prijenos* između procesa pošiljatelja i primatelja
- *kontrola toka:* da pošiljatelj ne preplavi primatelja
- *kontrola zakrčenja:* pošiljatelj usporava slanje u slučaju preopterećenja mreže
- *ne nudi:* vremenske garancije, garantiranu minimalnu propusnost, sigurnost
- *orijentiran na vezu:* nužna uspostava veze između procesa klijenta i poslužitelja

usluge UDP-a:

- *nepouzdan prijenos podataka* između procesa pošiljatelja i primatelja
- *ne nudi:* pouzdan prijenos, kontrolu toka, kontrolu zakrčenja, vremenske garancije, garantiranu minimalnu propusnost, sigurnost ni uspostavu veze

P: Zašto onda postoji UDP?

Internet aplikacije: aplikacijski i transportni protokoli

aplikacija	protokol aplikacijskog sloja	korišteni transportni protokol
e-pošta	SMTP [RFC 2821]	TCP
pristup udaljenom računalu	Telnet [RFC 854], SSH [RFC4251]	TCP
Web	HTTP [RFC 2616]	TCP
prijenos datoteka	FTP, SFTP, FTPS	TCP
strujanje multimedije	HTTP (e.g., YouTube), RTP [RFC 1889]	TCP ili UDP
Internet telefonija	SIP, RTP, vlasnički (npr. Skype)	TCP ili **UDP**
Interaktivne igre	WOW, FPS (vlasnički)	UDP ili TCP

Sigurnost i TCP

TCP & UDP

- nema kriptiranja
- lozinke poslane socketu u jasnom obliku, putuju Internetom u jasnom obliku

SSL/TLS

- pruža kriptiranu TCP-vezu
- integritet podataka
- autentifikacija krajnjih točaka

SSL je na aplikacijskom sloju

- aplikacije koriste SSL biblioteke, koje “pričaju” s TCP-om

SSL socket API

- lozinke poslane socketu u jasnom obliku, putuju Internetom kriptirane
- … → poglavlje 8

Poglavlje 2: sadržaj

2.1 principi na kojima se zasnivaju mrežne aplikacije

2.2 Web i HTTP

2.4 elektronička pošta

- SMTP, POP3, IMAP

2.3* Prijenos datoteka

2.5 DNS

2.6 P2P aplikacije

2.7 socket programiranje s UDP-om i TCP-om

Web i HTTP

Prvo, pregled…

- *web stranica* se sastoji od *objekata*
- objekt može biti HTML datoteka, JPEG slika, Java *applet*, audio datoteka,…
- web stranica se sastoji od *osnovne HTML-datoteke* koja može sadržati *više referenciranih objekata*
- svaki objekt se može adresirati *URL-om,* npr:

`www.unizg.hr/nekiOdjel/slika.gif`

ime računala — put do datoteke

URL

- HTTP koristi uslugu TCP-a,TCP koristi uslugu IP-a
- uz naziva i smještaj objekta na poslužitelju, iz URL-a treba znati i:
  - broj TCP porta poslužitelja
  - IP-adresu poslužitelja
- općenitiji oblik URL-a:

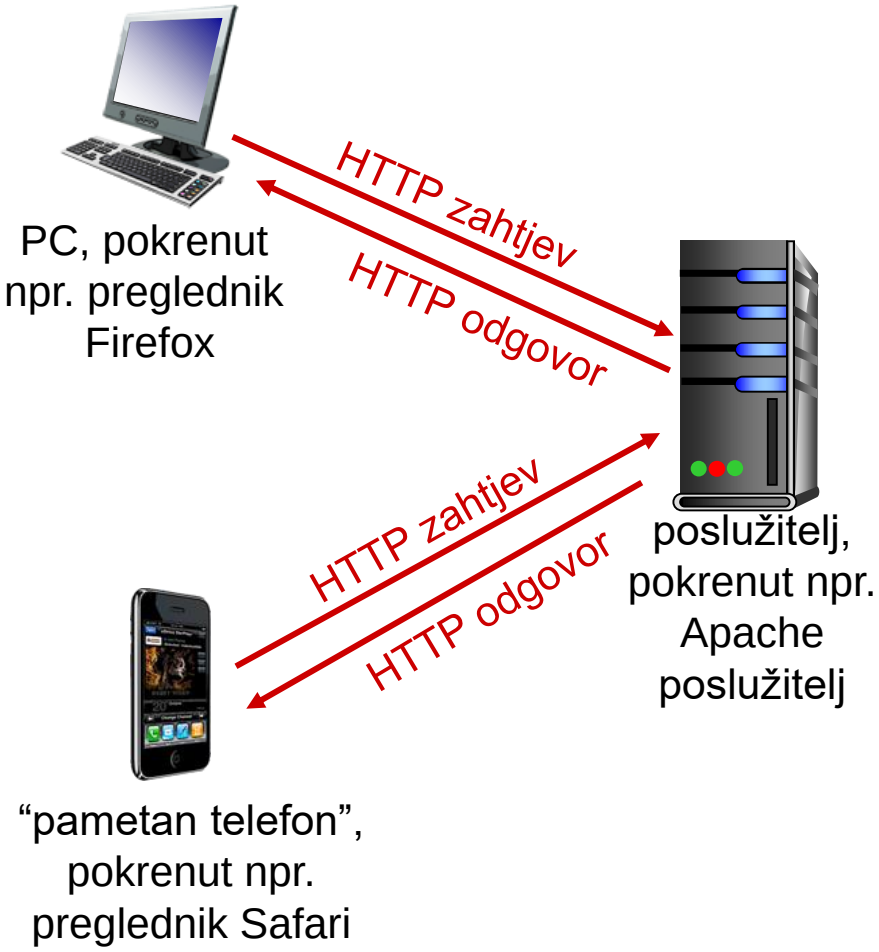
```
protokol://ime_računala[:port]/putanja/datoteka
```

- ime računala se mora prevesti u IP-adresu → to obavlja DNS
- ako je broj porta izostavljen, web preglednik pretpostavlja da se radi o TCP-portu 80

HTTP: pregled

HTTP: hypertext transfer protocol

- protokol aplikacijskog sloja kojeg koristi Web
- model klijent/poslužitelj
  - *klijent:* preglednik koji zahtjeva, prima, (koristeći protokol HTTP) i prikazuje Web objekte
  - *poslužitelj:* Web poslužitelj šalje (koristeći protokol HTTP) objekte koje klijent zahtjeva



HTTP: pregled (…nastavak)

koristi TCP:

- klijent traži uspostavu TCP-veze (stvara socket) prema poslužitelju, na port 80
- poslužitelj prihvaća TCP-vezu od klijenta
- HTTP poruke (poruke protokola aplikacijskog sloja) se razmjenjuju između preglednika (klijent HTTP-a) i web-poslužitelja (poslužitelj HTTP-a)
- na kraju raskid TCP-veze

HTTP je protokol "bez stanja"

- poslužitelj ne pamti informacije o prethodnim zahtjevima

usput

protokoli koji pamte "stanja" su složeni!

- protekla prošlost (stanje) se mora održavati
- ako poslužitelj/klijent ispadne, njihova percepcija "state" može biti različita, potrebno definirati postupak oporavka

HTTP veze

neperzistenan HTTP

- prenosi se najviše jedan objekt po TCP-vezi
  - veza se zatim raskida
- dohvaćanje više objekata zahtjeva više TCP-veza

perzistentan HTTP

- više objekata se mogu poslati preko jedne TCP-veze (uspostavljene između klijenta i poslužitelja)

Neperzistentan HTTP

neka korisnik unese URL:
`www.unizg.hr/nekiOdjel/home.index`

(sadrži tekst, referencira 10 jpeg slika)

1a. klijent HTTP-a zahtjeva uspostavu TCP-veze s HTTP poslužiteljem (procesom) na www.unizg.hr na portu 80

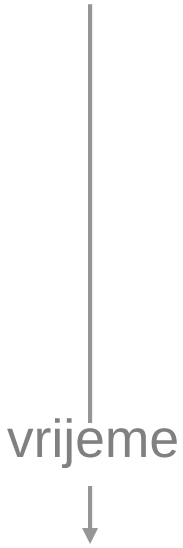
1b. poslužitelj HTTP-a na računalu www.unizg.hr čeka na TCP-vezu na portu 80. "Prihvaća" vezu i o tome obavještava klijenta

2. klijent HTTP-a šalje HTTP *poruku zahtjeva* (sadrži URL) u TCP socket. Klijent u poruci traži objekt **`nekiOdjel/home.index`**

3. poslužitelj HTTP-a prima poruku zahtjeva, formira *poruku odgovora* koja sadrži objekt i šalje je u socket

vrijeme

Neperzistentana HTTP (nastavak)



4. poslužitelj HTTP-a zatvara TCP-vezu

5. klijent HTTP-a prima poruku odgovora koju sadrži html datoteku te ju prikazuje. Analizira html datoteku, pronalazi 10 referenci na jpeg objekte

6. Koraci 1-5 se ponavljaju za svaki od 10 jpeg objekata

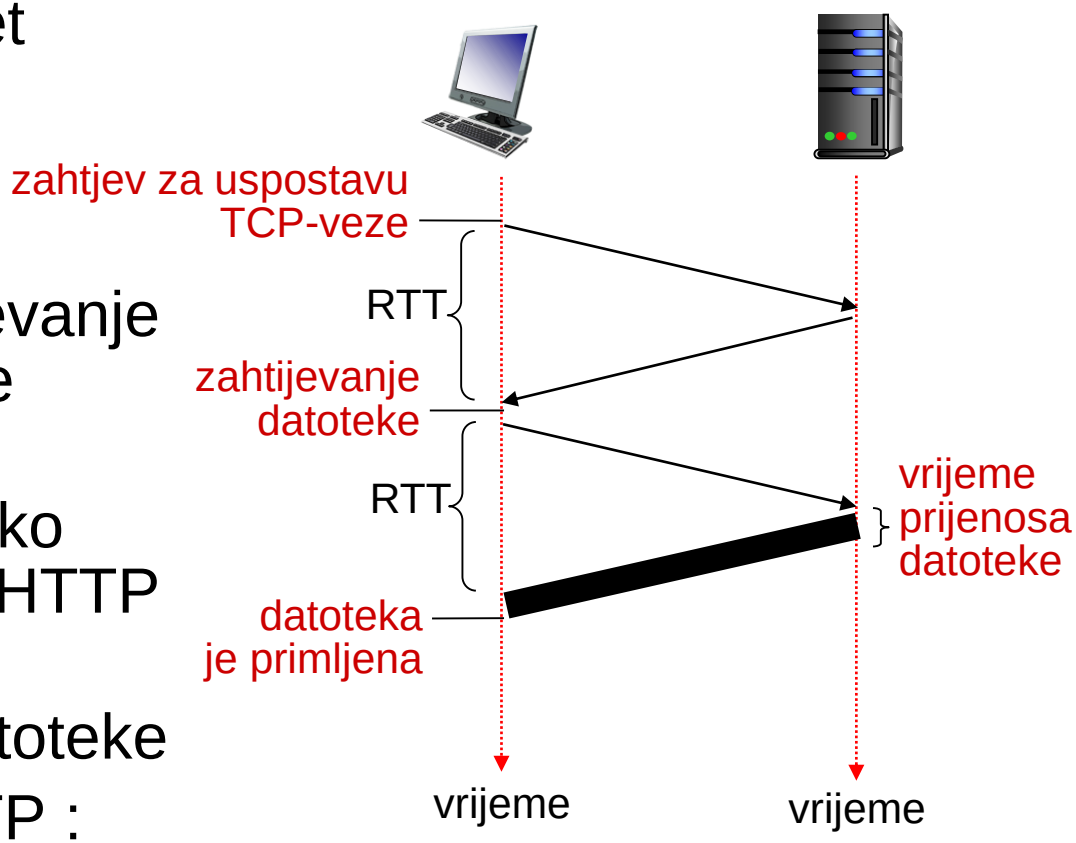
Neperzistentan HTTP: vrijeme odziva

RTT (definicija): potrebno vrijeme da mali paket putuje od klijenta do poslužitelja i natrag

HTTP vrijeme odziva:

- jedan RTT za zahtijevanje uspostave TCP-veze
- jedan RTT za HTTP zahtjev i prvih nekoliko bajtova za povratak HTTP odgovora
- vrijeme prijenosa datoteke
- ne-perzistentan HTTP : vrijeme odziva =

  2RTT+ vrijeme prijenosa datoteke



Perzistentan HTTP

nedostaci neperzistentnog HTTP-a:

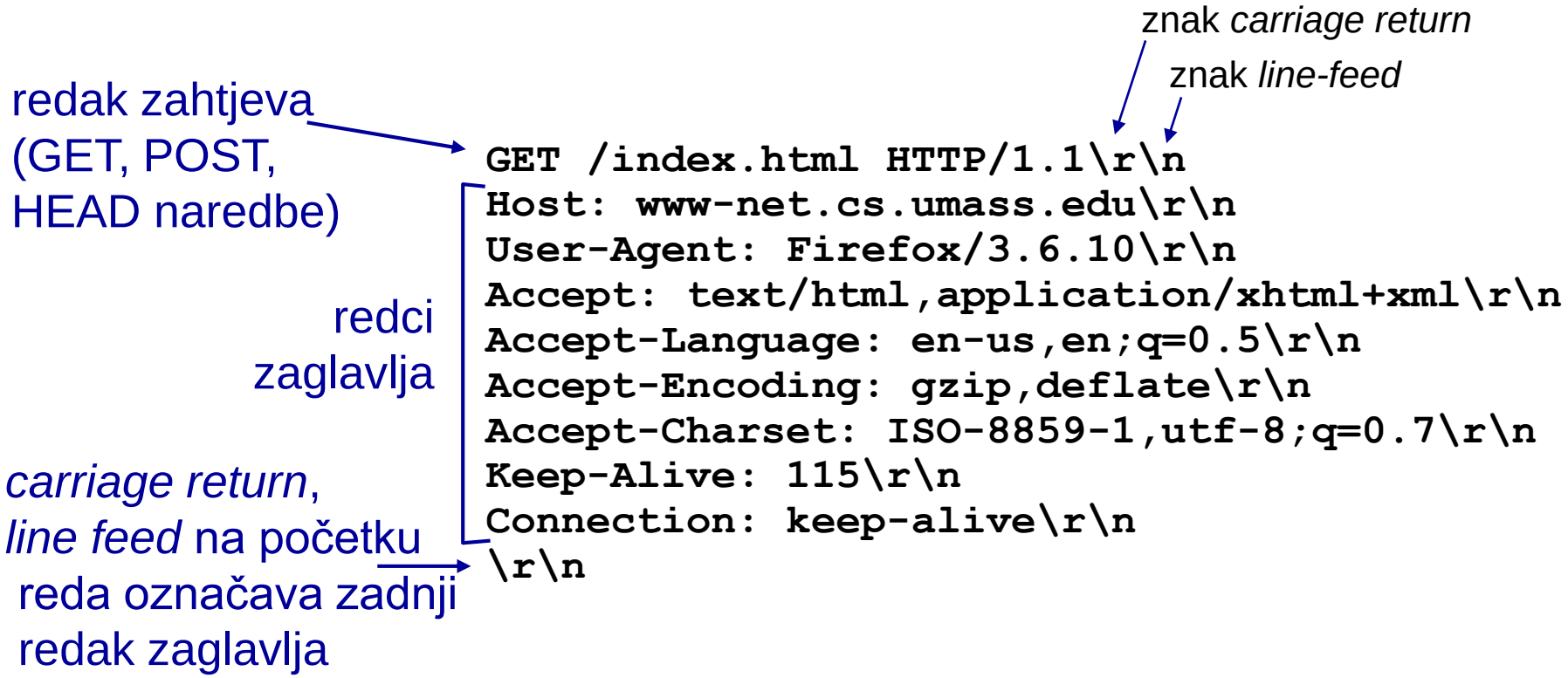
- zahtjeva 2 RTTa po objektu
- opterećenje OS-a za *svaku* TCP-vezu
- preglednici često otvaraju paralelne TCP-veze za dohvaćanje svakog referenciranog objekta

perzistentan HTTP:

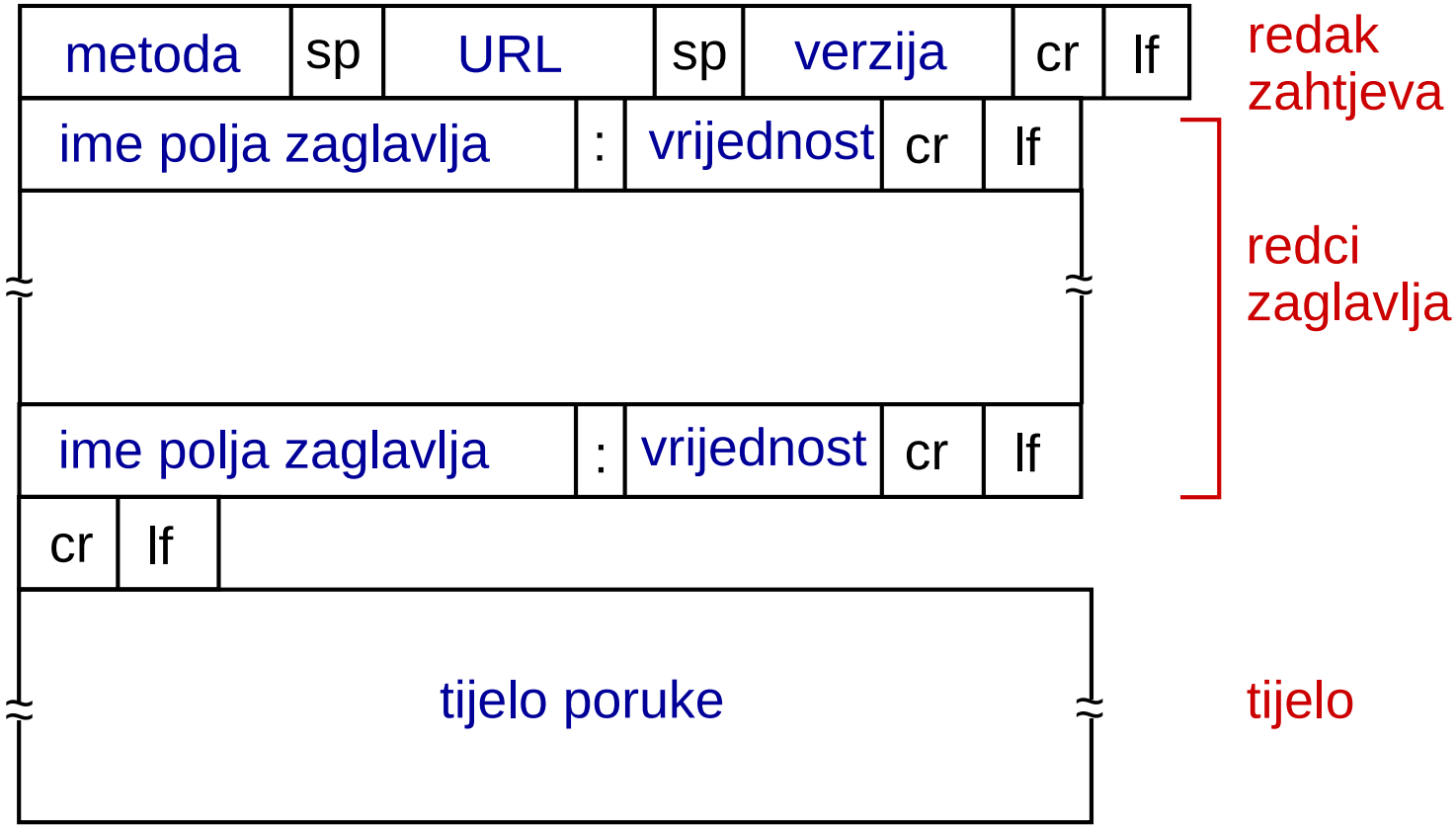
- poslužitelj ostavlja otvorenu vezu, nakon slanja odgovora
- sljedeće HTTP poruke između istih klijenta/poslužitelja šalju se preko otvorene veze
- klijent šalje zahtjev čim naiđe na referencu objekta
  - cjevovodno / ne-cjevovodno
- dohvaćanja svih referenciranih objekata moguće uz samo jedan RTT

HTTP: poruka zahtjeva

- dva tipa poruka HTTP-a: *zahtjev*, *odgovor*
- poruka zahtjeva HTTP-a:
  - ASCII (ljudima čitljiv oblik)



Poruka zahtjeva HTTP-a: opći format



Slanje podataka poslužitelju

POST metoda:

- web stranice često sadrže *forme* za unos
- podaci se prenose na poslužitelj u tijelu poruke

URL GET metode:

- koristi GET metodu
- podaci su ugrađeni u URL polje u retku zahtjeva:

```
www.nekomjesto.hr/zivotinje?majmuna=2&banana=7
```

- parovi (tip, vrijednost)
  - … ?tip1=vrijednost1&tip2=vrijednost2

Poruka odgovora HTTP-a

statusni redak (protokol statusni kôd statusna fraza)

redci zaglavlja

podaci, npr., zahtijevana HTML datoteka

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02
   GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-
   1\r\n
\r\n
data data data data data ...
```

HTTP: statusni kôdovi odgovora

- statusni kôd se javlja u 1. retku poruke odgovora koju poslužitelj šalje klijentu
- neki primjeri kôda:

200 OK

- zahtjev je uspio, zahtijevani objekt se nalazi u toj poruci

301 Moved Permanently

- zahtijevani objekt je preseljen, nova lokacija je navedena u toj poruci (Location:)

400 Bad Request

- poslužitelj ne razumije poruku zahtjeva

404 Not Found

- traženi dokument nije pronađen na poslužitelju

505 HTTP Version Not Supported

Tipovi metoda

HTTP/1.0:

- GET
- POST
- HEAD
  - traži od poslužitelja da u odgovoru izostavi traženi objekt

HTTP/1.1:

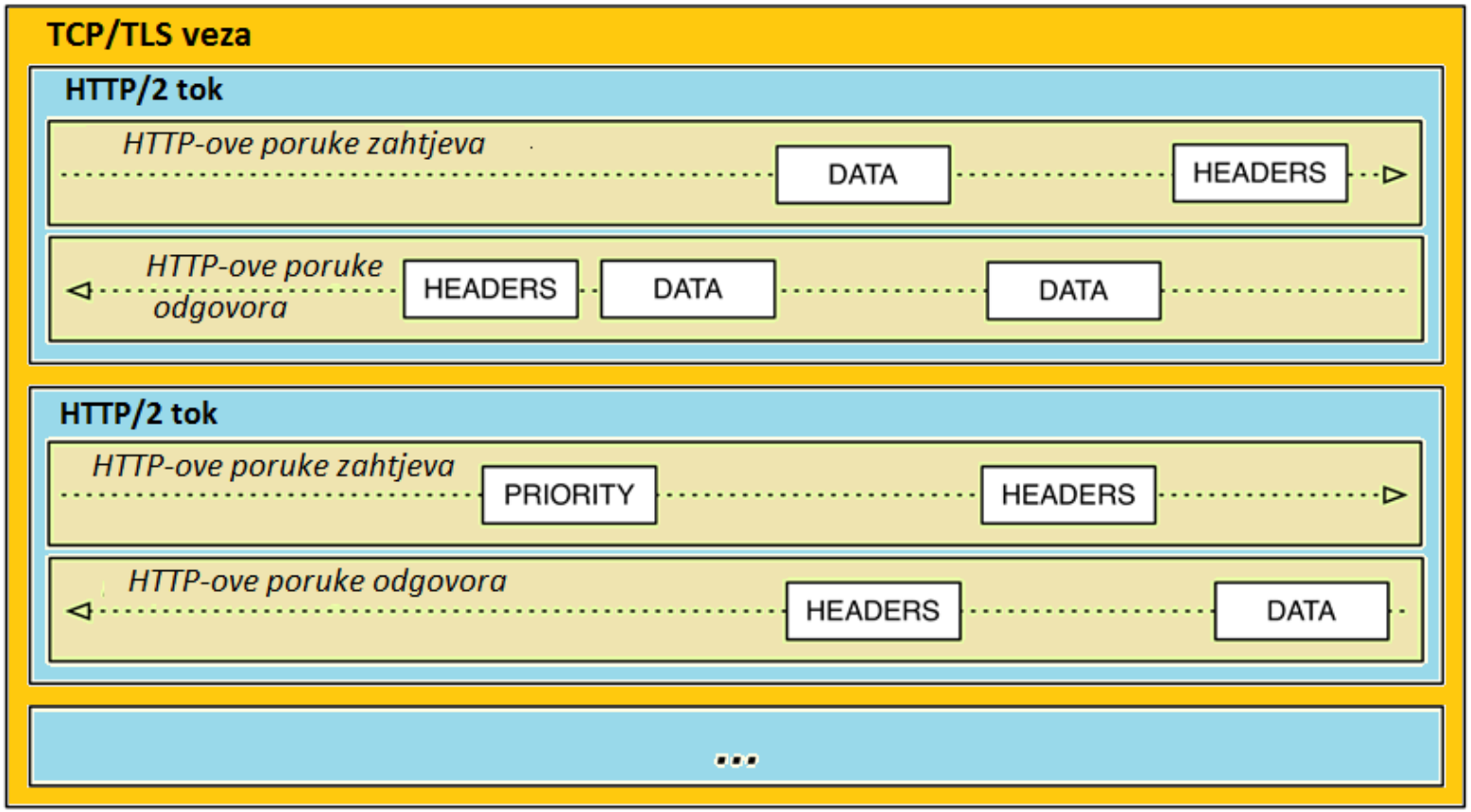
- GET, POST, HEAD
- PUT
  - *upload* datoteke iz tijela poruke na putanju navedenu u URL polju
- DELETE
  - brisanje datoteke navedene u URL polju

HTTP/2 (RFC 7540)

- ***cilj***: smanjiti latenciju i vrijeme potrebno za učitavanje stranice u web pregledniku te zadržati kompatibilnost unazad
- multipleksiranje i konkurentnost
  - smanjiti broj RTT-ova
  - riješiti HOL
  - ukloniti nedostatke paralelnih TCP-veza
  - koriste se dvosmjerni tokovi/strujanja

HTTP/2: tokovi, poruke i okviri

- jedan tok ↔ analogno jednoj TCP-vezi kod korištenja paralelnih TCP-veza



HTTP/2: zaglavlje okvira

- poruke se prenose unutar HTTP/2 okvira
  - 1 poruka ->1 ili više okvira
  - binaran oblik
  - Tipovi okvira: DATA , HEADERS, SETTINGS, PRIORITY, ...
- kompresija zaglavlja „HPACK”, binarni format
  - ubrzati obradu zaglavlja
  - skratiti vrijeme prijenosa
- TLS 1.2 ili noviji
  - enkripcija je prema standardu opcionalna, ali je web-preglednici uglavnom zahtijevaju

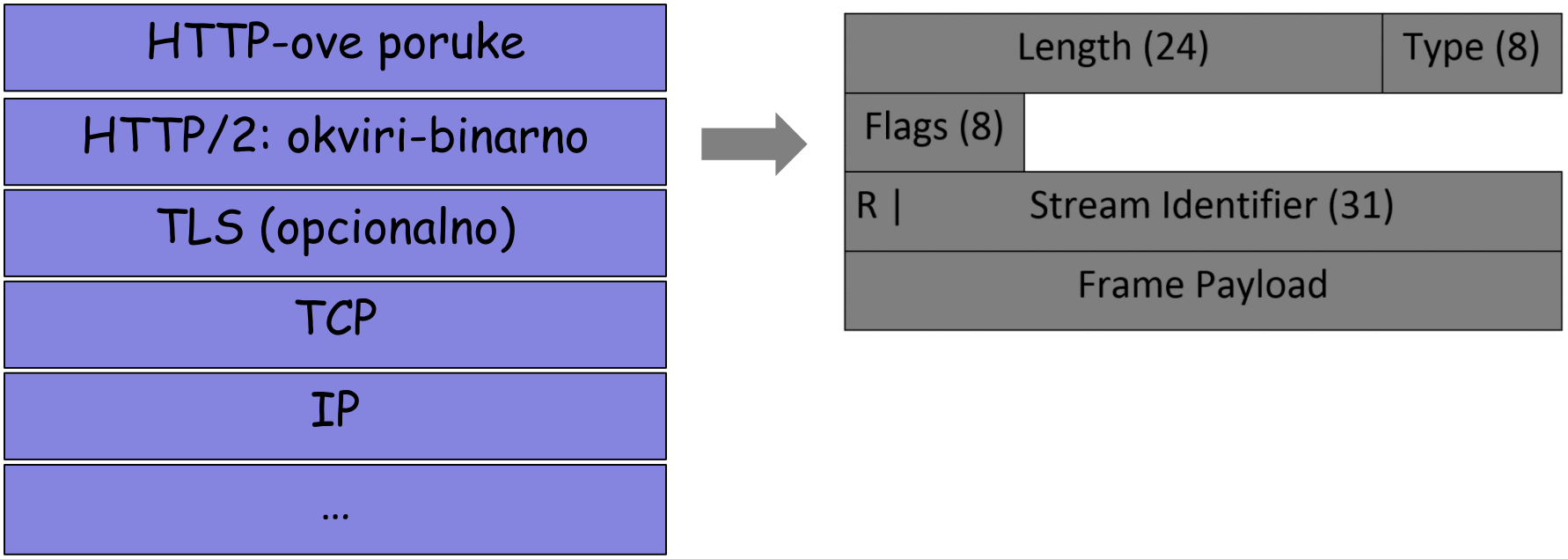
Od HTTP/2 prema HTTP/3

Glavni cilj: smanjiti kašnjenje kod zahtijevanja više objekata

HTTP/2 preko jedne TCP-veze znači:

- oporavak od izgubljenih paketa i dalje zadržavaju prijenos svih objekata
  - kao i kod HTTP 1.1, web preglednici mogu otvarati paralelne TCP-veze kako bi smanjili kašnjenje i povećali ukupnu propusnost
- obična TCP-veza ionako ne pruža sigurnost
- HTTP/3: dodaje sigurnost, kontrola pogrešaka i zakrčenja na razini objekata (više je cjevovodno) preko UDP-a
  - više o HTTP/3 na transportnom sloju

HTTP/2: okviri



Kako isprobati HTTP (strana klijenta)

1. Spojite se Telnetom na neki Web poslužitelj:

```
telnet cis.poly.edu 80
```

otvara TCP-vezu na port 80
(predef. HTTP port poslužitelja) na cis.poly.edu.
sve što se utipka se šalje
na port 80 na cis.poly.edu

2. Utipkajte zahtjev GET HTTP:

```
GET /~ross/ HTTP/1.1
Host: cis.poly.edu
```

Nakon utipkavanja (pritisnite *carriage return* dvaput). Time ste poslali minimalan (ali potpun) HTTP GET zahtjev poslužitelju

3. Pogledajte poruku odgovora koju je poslao HTTP poslužitelj!

(ili pomoću Wiresharka pogledajte snimljene HTTP zahtjeve/odgovore)

Dinamički sadržaj

izvodi se programski kôd koji dinamički generira sadržaj

na strani poslužitelja

- poslužitelj izvodi programski kôd i dinamički generira html objekt koji šalje klijentu
- CGI, PHP, ASP.NET, ...

na strani klijenta

- klijnent prima html kod u kojem je ugrađena skripta
- klijent interpretira skriptu i dinamički mijena sadržaj/izgled stranice
  - Javascript, …

Bilježenje stanja: kolačići (*cookies*)

mnoga web sjedišta koriste kolačiće

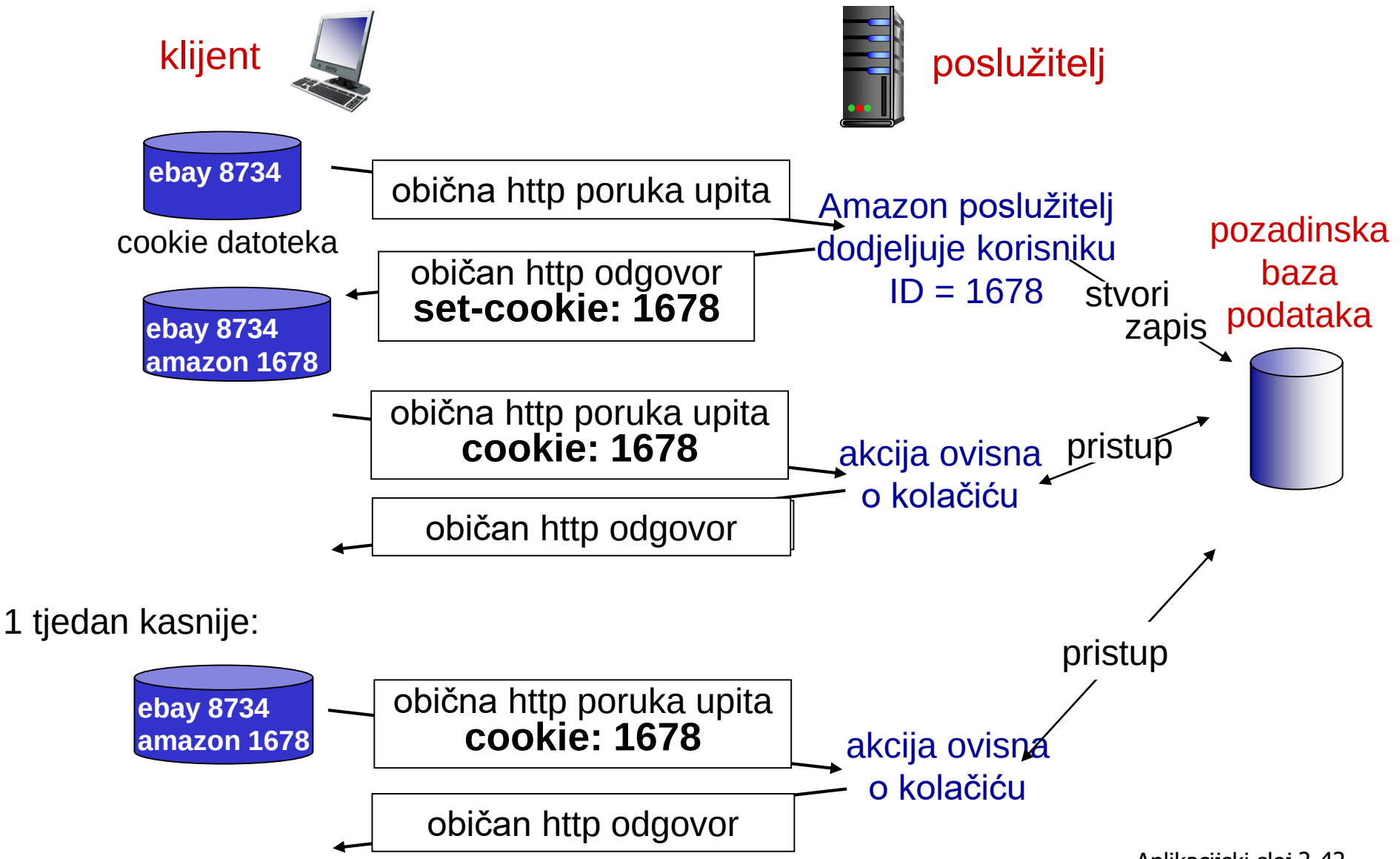
četiri komponente:

1) kolačićev redak zaglavlja u poruci HTTP odgovora
2) kolačićev redak zaglavlja u sljedećoj poruci HTTP upita
3) kolačićeva datoteka spremljena na računalu korisnika, kojom upravlja web preglednik
4) baza podataka na web sjedištu

primjer:

- Suzana uvijek pristupa Internetu sa svog PC-a
- prvi put posjećuje određenu web-trgovinu
- kada inicijalni HTTP zahtjev stigne u sjedište, sjedište stvara:
  - jedinstveni ID
  - zapis u svojoj bazi podataka i koristi ID kao ključ

Kolačići: održavanje “stanja”(nastavak)



Kolačići (…nastavak)

za što se kolačići mogu koristiti:

- autorizaciju
- košarica za kupovinu
- preporuke/reklame
- stanje korisničke sjednice (web e-mail)

kako pamtiti "stanje":

- krajnje točke protokola: održavaju stanje višestrukih transakcija kod pošiljatelja/primatelja
- kolačići: HTTP poruke prenose informaciju o stanju

usput

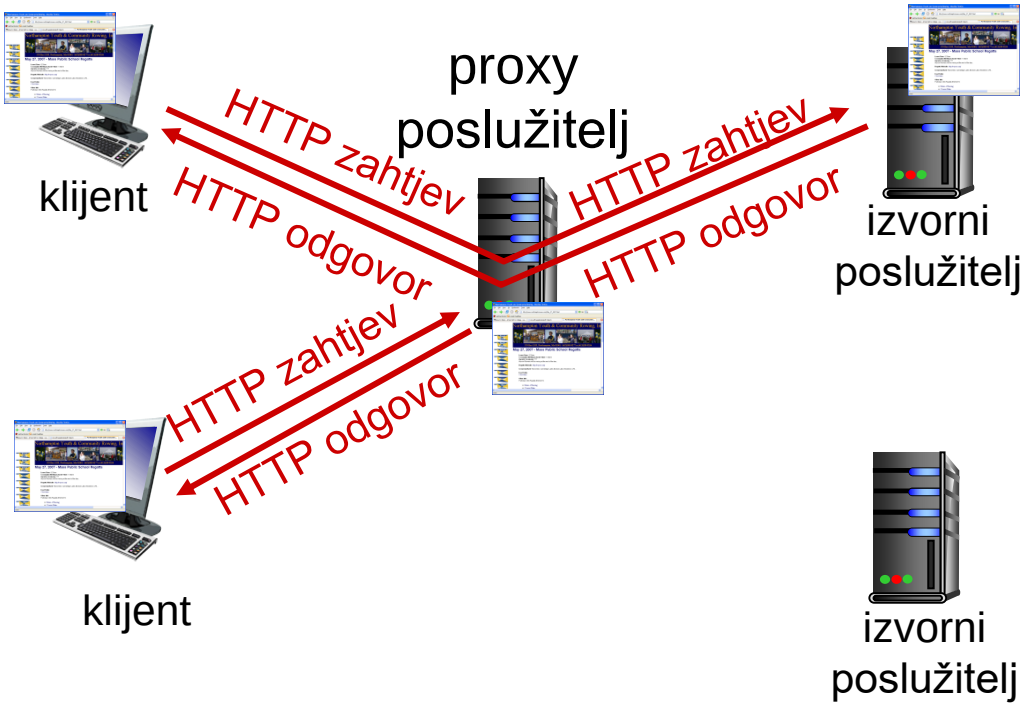
kolačići i privatnost:

- pomoću kolačića web-sjedišta mogu naučiti mnogo o korisnicima
- korisnik je nekad prethodno možda unio svoje ime i e-adresu

Web cache (*proxy* poslužitelj)

cilj: odgovoriti na zahtjev klijenta bez da se uključuje izvorni poslužitelj

- korisnik podešava preglednik: web pristup preko *cachea*
- preglednik šalje sve HTTP zahtjeve *cacheu*
  - **objekt je u *cacheu***: *cache* vraća objekt
  - **inače**: *cache* zahtjeva objekt od izvornog poslužitelja te ga po primitku šalje klijentu



Još o web *cachingu*

- cache se ponaša i kao klijent i kao poslužitelj
  - poslužitelj je za klijenta koji mu šalje zahtjeve
  - kada upućuje zahtjeve izvornom poslužitelju onda je u ulozi klijenta
- uobičajeno je *cache* instalira ISP (sveučilište, kompanija, rezidencijalni ISP)

zašto web caching?

- smanjuje vrijeme odgovora na klijentov zahtjev
- smanjuje promet na pristupnom vodu institucije
- Internet je pun *cache*a: omogućuje "slabom" pružatelju sadržaja da isporuči sadržaj (slično je i P2P dijeljenje datoteka)

Primjer *cachinga*:

pretpostavke:

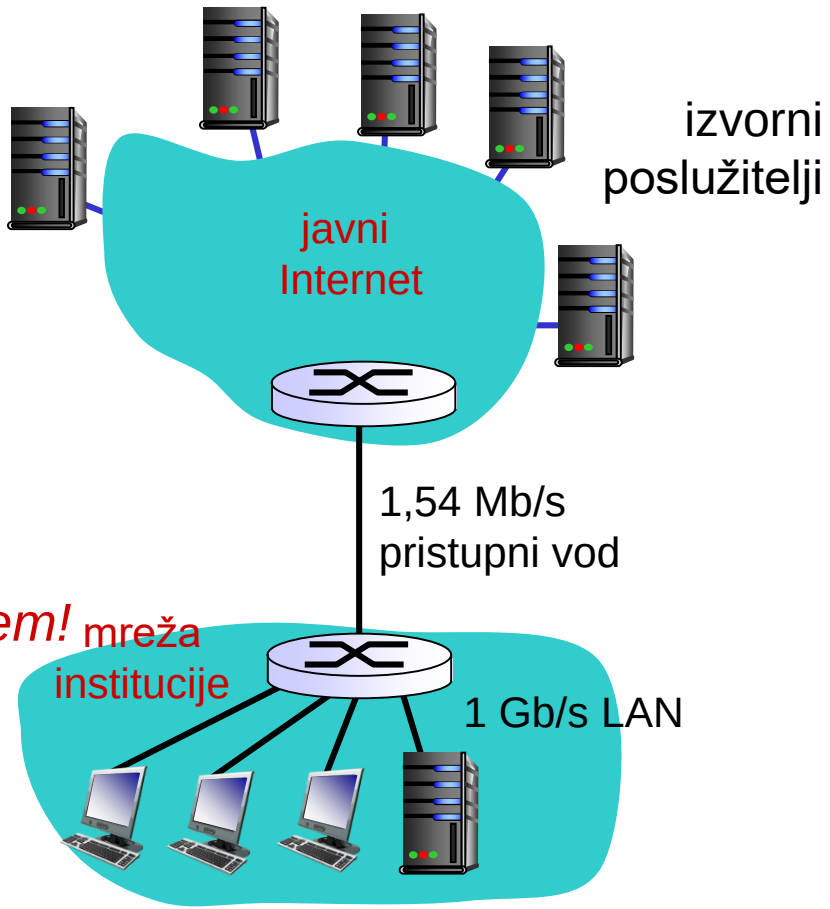
- prosj. veličina objekta: 100 Kb
- prosj. broj zahtjeva koje preglednici šalju poslužitelju:15 /sekundi
- prosj. podataka prema preglednicima: 1,50 Mb/s
- RTT od institucijskog usmjernika do bilo kojeg odredišnog poslužitelja: 2 s
- kapacitet pristupnog voda 1,54 Mb/s

posljedice:

- iskorištenost LAN-a: 0,15%
- iskorištenost pristupnog voda = 97%
- ukupno kašnjenje = "Internet kašnjenje" + "pristupno kašnjenje" + "LAN kašnjenje"

  = 2 s + minute + μsekunde

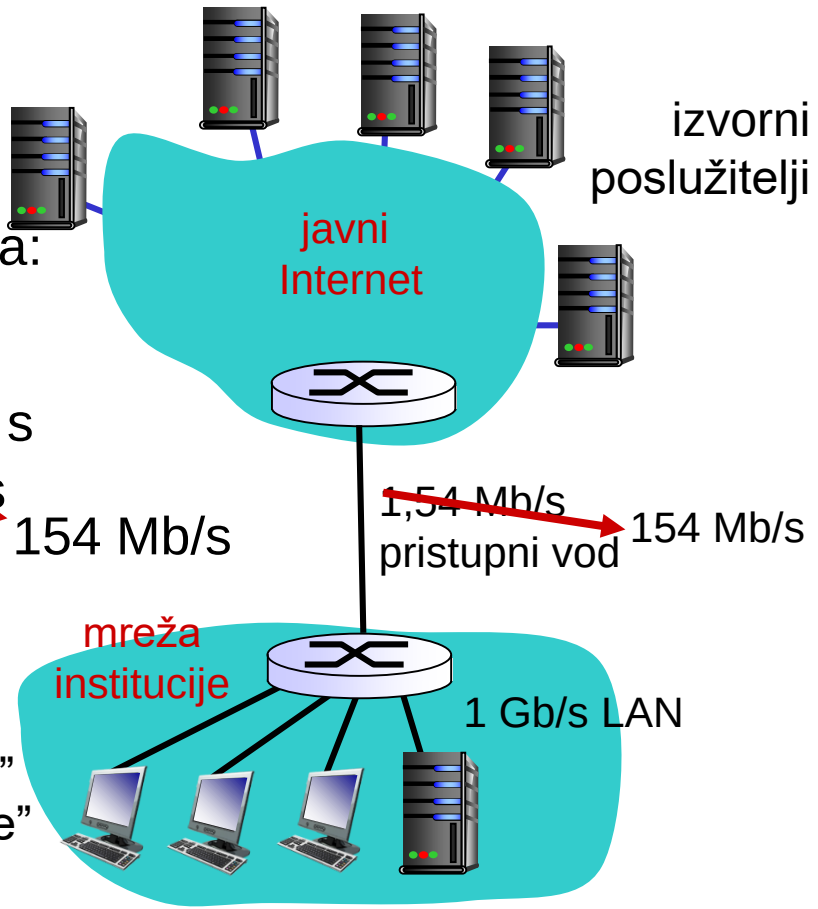
problem!



Primjer *cachinga*: brži pristupni vod

pretpostavke:

- prosj. veličina objekta: 100 Kb
- prosj. broj zahtjeva koje preglednici šalju poslužitelju:15 /sekundi
- prosj. podataka prema preglednicima: 1,50 Mb/s
- RTT od institucijskog usmjernika do bilo kojeg odredišnog poslužitelja: 2 s
- kapacitet pristupnog voda 1,54 Mb/s



posljedice:

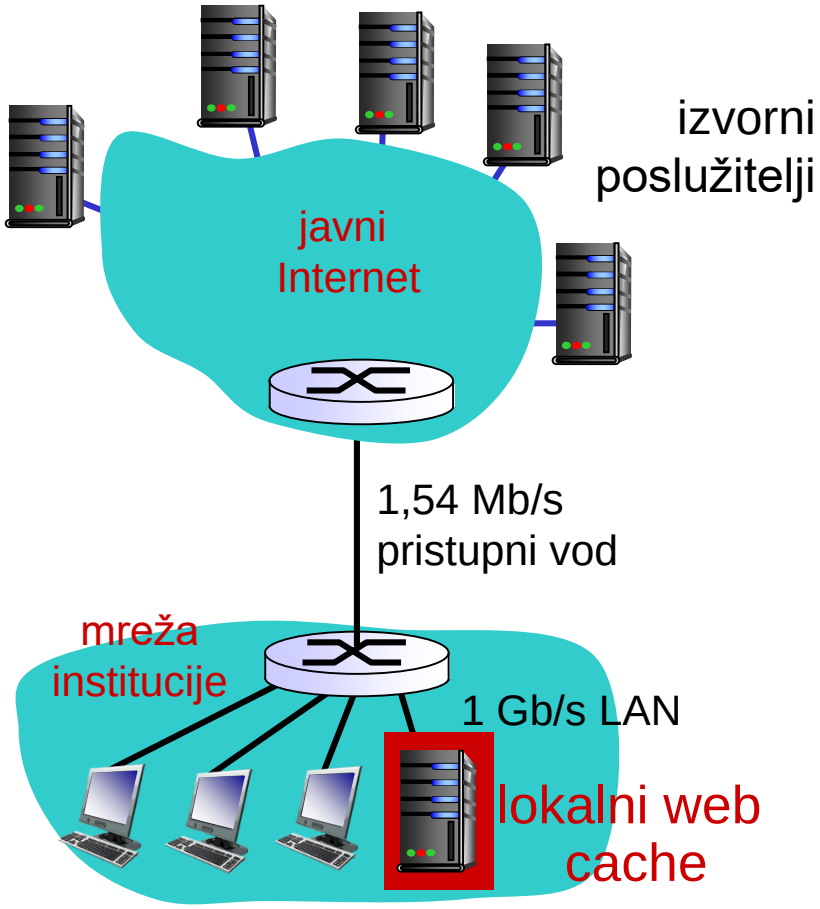
- iskorištenost LAN-a: 0,15%
- iskorištenost pristupnog voda = ~~97%~~ 0,97%
- ukupno kašnjenje = "Internet kašnjenje" + "pristupno kašnjenje" + "LAN kašnjenje"
  = 2 s + ~~minute~~ + μsekunde
  msekunde

Cijena: povećanje kapaciteta pristupnog voda (nije jeftino!)

Primjer *cachinga*: ugradnja lokalnog cachea

pretpostavke :

- prosj. veličina objekta: 100 Kb
- prosj. broj zahtjeva koje preglednici šalju poslužitelju:15 /sekundi
- prosj. podataka prema preglednicima: 1,50 Mb/s
- RTT od institucijskog usmjernika do bilo kojeg odredišnog poslužitelja: 2 s
- kapacitet pristupnog voda 1,54 Mb/s



posljedice:

- iskorištenost LAN-a: 0,15%
- iskorištenost pristupnog voda = ?
- Ukupno kašnjenje = ?

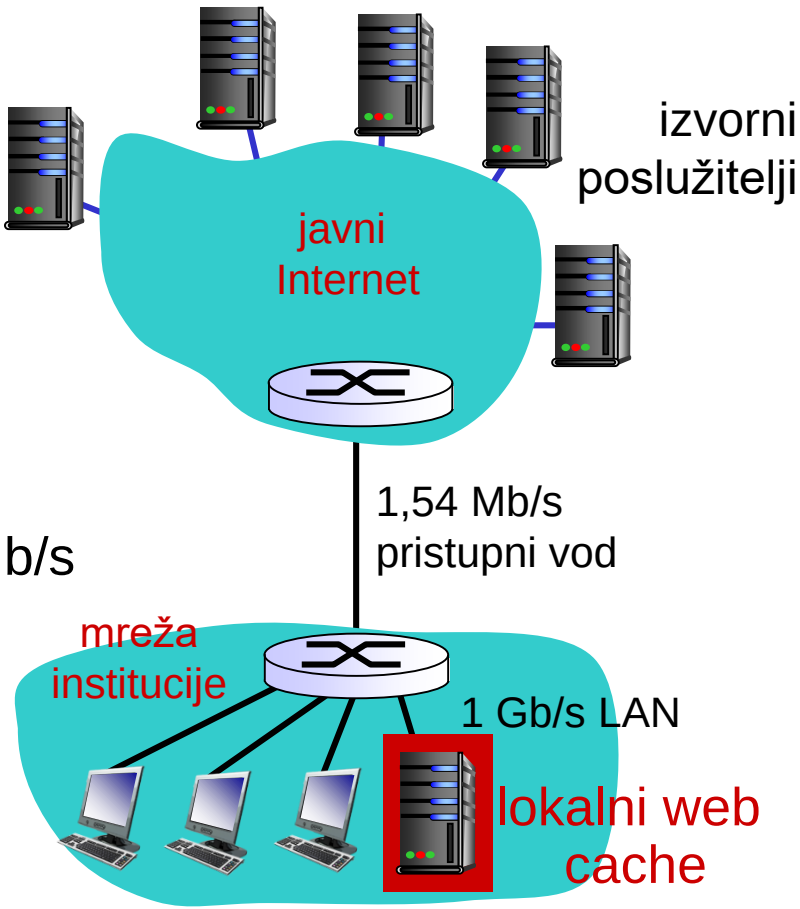
Kako izračunati iskoristivost voda i kašnjenje?

Cijena: web cache (jeftino!)

Primjer *cachinga*: ugradnja lokalnog cachea

Računanje iskoristivosti lokalnog voda i kašnjenja za cache:

- neka je omjer pogodaka cachea 0.4
  - 40% zahtjeva ide iz *cachea*, 60% zahtjeva ide od izvornih poslužitelja
- iskoristivost pristupnog voda:
  - 60% zahtjeva koristi pristupni vod
- podaci preko pristupnog voda do preglednika = 0.6*1,50 Mb/s = 0,9 Mb/s
  - iskorištenost = 0,9/1,54 = 0,58
- ukupno kašnjenje
  - = 0,6 * (kašnjenje od izvornih poslužitelja) +0.4 * (kašnjenje iz cachea)
  - = 0,6 (2,01) + 0,4 (~ms)
  - = ~ 1,2 s
  - manje nego sa vodom od154 Mb/s (i jeftinije također!)

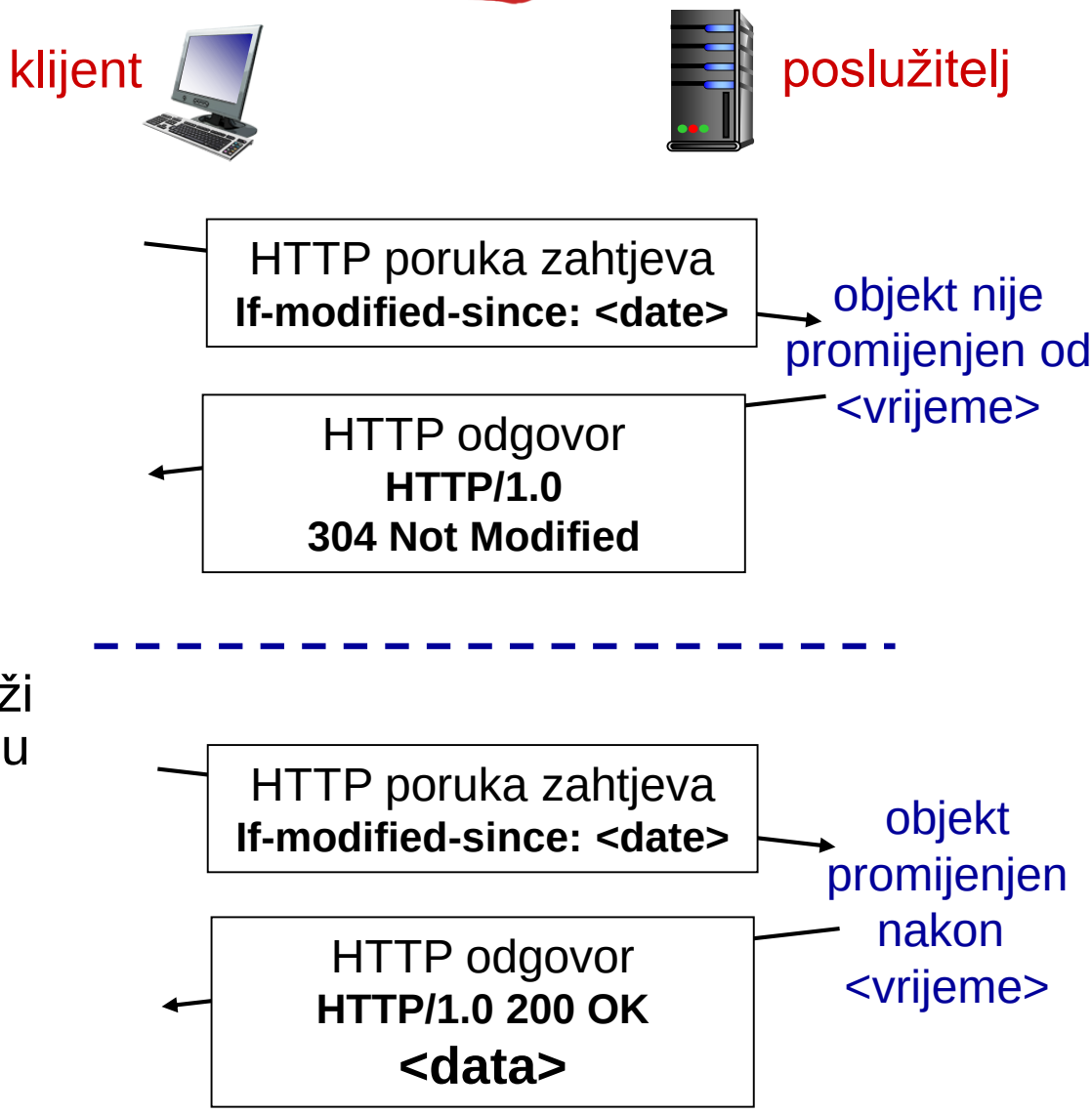


Uvjetni GET (*Conditional GET*)

- *Cilj:* ne šalji objekt ako je u *cacheu* aktualna verzija
  - nema kašnjenja zbog vremena prijenosa objekta
  - manja iskorištenost voda
- *cache:* naznači u HTTP zahtjevu vrijeme kada je objekt spremljen u cache

  `If-modified-since: <vrijeme>`
- *poslužitelj:* odgovor ne sadrži objekt ako je kopija u cacheu ažurna:

  `HTTP/1.0 304 Not Modified`



Gdje smjestiti *cache*?

- različite mogućnosti
  - na klijentu
    - klijent pohranjuje objekte koje je već dohvatio
  - *forward proxy*
    - nalazi se blizu klijenta
    - klijent je svjestan *proxya* kojem upućuje zahtjeve
  - *reverse proxy*
    - nalazi se blizu poslužitelja
    - "glumi" poslužitelja (klijent ga smatra poslužiteljem)
    - smanjuje opterećenje poslužitelja

Web-usluge (*web services*)

- web
  - komunikacija između čovjeka i stroja
  - korisnik određuje zahtjev (upisuje adrese, odabire hiperpoveznice)
- web-usluge
  - komunikacija između stroja i stroja
  - automatiziraju se postupci npr. u poslovanju
  - zahtijeva uzajamno djelovanje (interoperabilnost) različitih sustava koji u pravilu nisu razvijani da bi međusobno komunicirali
  - dvije različite arhitekture, neformalno SOAP i REST

SSDP (*Simple Service Discovery Protocol*)

- služi za otkrivanje uključi-i-koristi (*plug & play*) uređaja
- dio je UPnP-a (*Universal Plug and Play*)
- koristi se HTTP-ov format poruka koji se šalje preko UDP-a, a ne preko TCP-a
- otkrivanje usluge
  - klijent višeodredišno šalje (HTTPMU) zahtjev otkrivanja na UDP port 1900 (IP-adresa 239.255.255.250)
  - uređaj/usluga koja se prepozna u zahtjevu, jednoodredišno šalje (HTTPU) odgovor
- oglašavanje usluge
  - uređaj/usluga može oglašavati svoju prisutnost, višeodredišnim slanjem (HTTPMU) poruke

Poglavlje 2: sadržaj

2.1 principi na kojima se zasnivaju mrežne aplikacije

2.2 Web i HTTP

2.4 elektronička pošta

- SMTP, POP3, IMAP

2.3* Prijenos datoteka

2.5 DNS

2.6 P2P aplikacije

2.7 socket programiranje s UDP-om i TCP-om

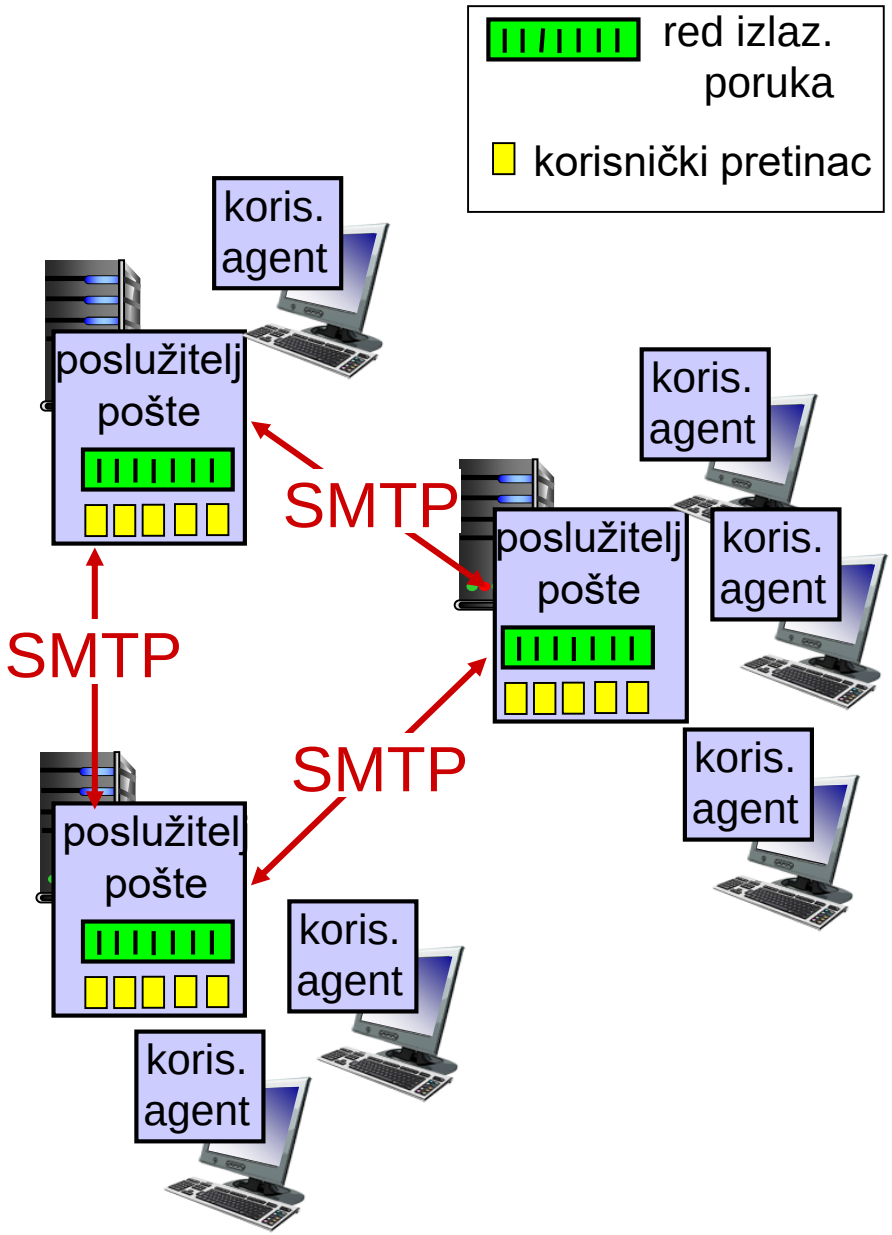
Elektronička pošta

Tri glavne komponente:

- korisnički agenti
- poslužitelji e-pošte
- simple mail transfer protocol: SMTP

Korisnički agent (User agent)

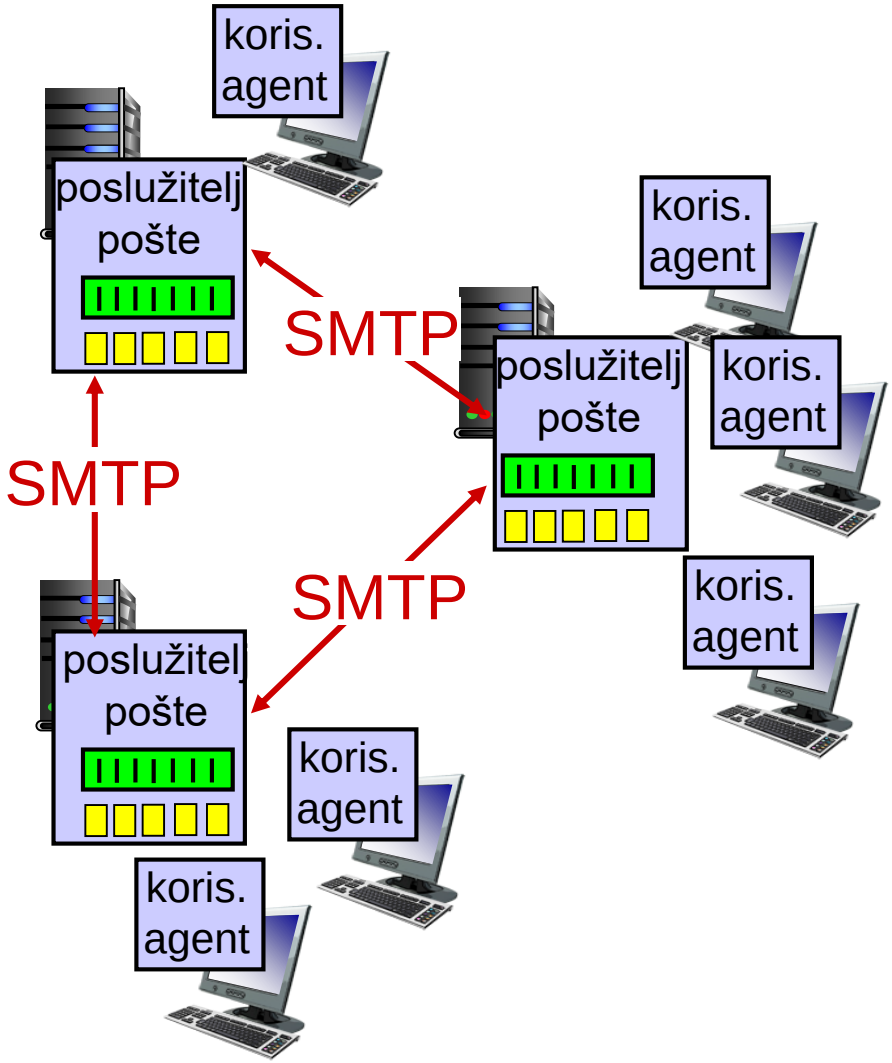
- poznat i kao “čitač pošte (mail reader)”
- sastavljanje, uređivanje čitanje poruka pošte
- npr. Outlook, Thunderbird…
- odlazne i dolazne poruke su pohranjene na poslužitelju



Elektronička pošta: poslužitelj pošte

poslužitelji pošte (mail servers):

- *pretinac (mailbox)* sadrži pristigle poruke za korisnika
- *red poruka (message queue)* odlaznih poruka (koje treba poslati)
- *protokol SMTP* za slanje poruka između poslužitelja pošte
  - klijent: poslužitelj koji šalje poruku
  - "poslužitelj": poslužitelj koji prima poruku

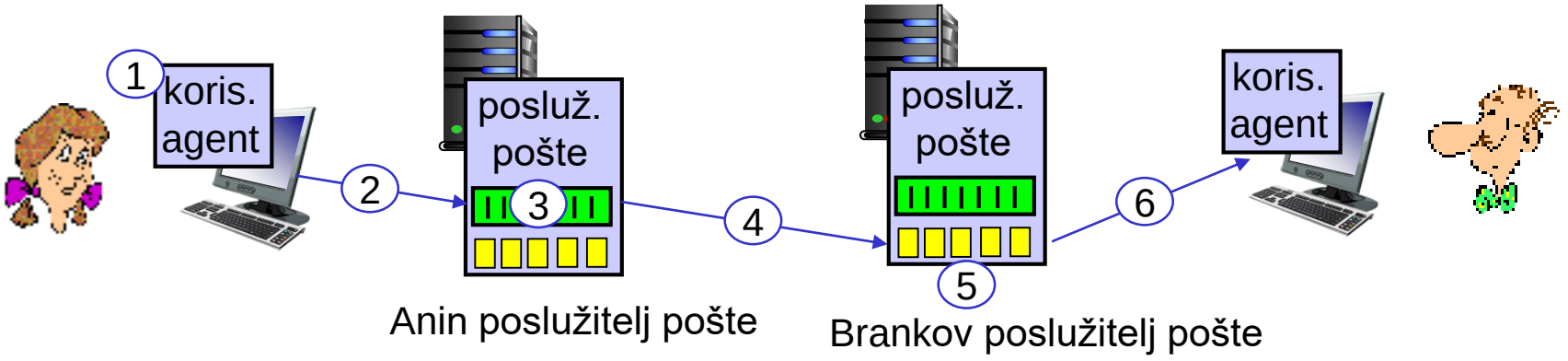


Elektronička pošta: SMTP [RFC 2821]

- koristi TCP za pouzdan prijenos poruka e-pošte od klijenta do poslužitelja, port 25
- izravan prijenos: poslužitelj pošiljatelj do poslužitelja primatelja
- tri faze prijenosa
  - uspostava veze (rukovanje)
  - prijenos poruka
  - zatvaranje veze
- interakcija naredba/odgovor (kao HTTP, FTP)
  - naredbe: ASCII tekst
  - odgovor: statusni kôd i fraza
- poruke moraju biti u 7-bitnom ASCII kôdu

Scenarij: Ana šalje poruku Branku

1) Ana koristi korisnički agent sastavljanje poruke "za" `branko@nekaskola.hr`
2) Anin korisnički agent šalje poruku do njenog poslužitelja pošte; poruka se stavlja u *red poruka*
3) klijentska strana SMTP-a uspostavlja TCP vezu s Brankovim poslužiteljem pošte
4) SMTP klijent šalje Aninu poruku preko TCP veze
5) Brankov poslužitelj pošte stavlja poruku u Brankov poštanski pretinac
6) Branko pokreće svog korisničkog agenta da bi čitao poruke



Primjer SMTP interakcije

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250  Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Isprobajte sami SMTP interakciju:

- **`telnet servername 25`**
- pogledajte 220 odgovore od poslužitelja
- unesite naredbe HELO, MAIL FROM, RCPT TO, DATA, QUIT

na ovaj način se može alati e-mail bez da se koristi klijent e-pošte

SMTP: završne napomene

- SMTP koristi perzistentne veze
- SMTP zahtjeva da poruke (zaglavlje & tijelo) budu u 7-bit ASCII obliku
- SMTP poslužitelj koristi `CRLF.CRLF` za određivane kraja poruke

usporedba s HTTP-om:

- HTTP: dohvaća (pull)
- SMTP: šalje (push)
- oboje koriste ASCII naredba/odgovor interakciju i statusne kôdove
- HTTP: svaki objekt je ugrađen u svoju vlastitu poruku odgovora
- SMTP: više objekata se šalje u višedijelnoj poruci

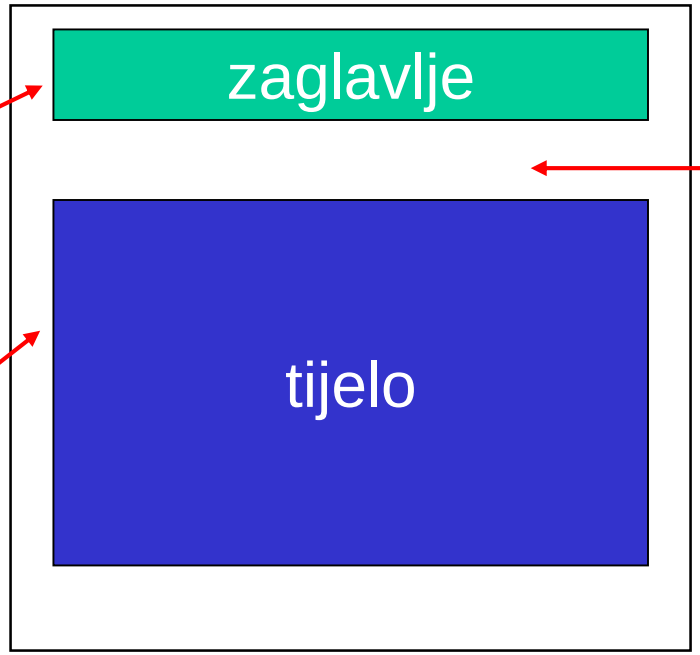
Format poruke pošte

SMTP: protokol za razmjenu poruka e-pošte

RFC 822, RFC 2822: standard za format tekstualne poruke:

- redci zaglavlja, npr.:
  - To:
  - From:
  - Subject:

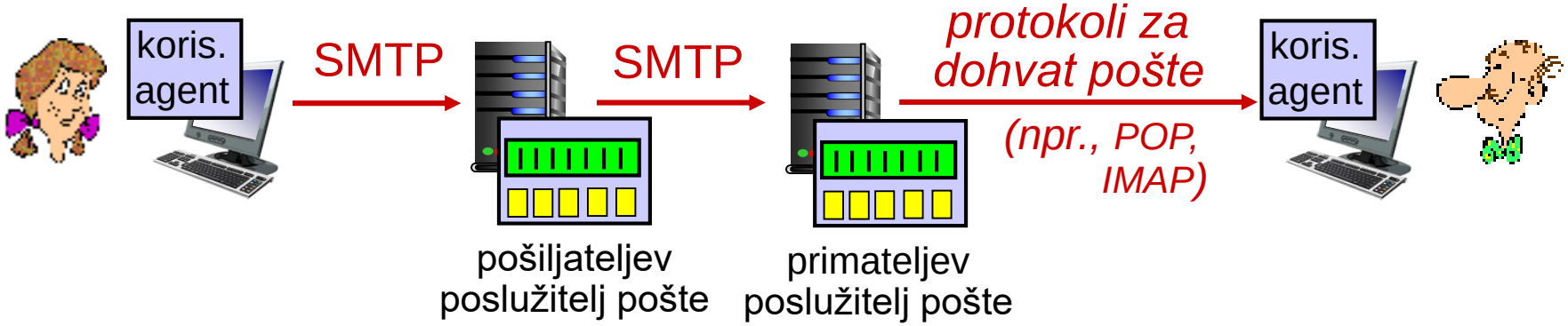
  različito od naredbi SMTP MAIL FROM, RCPT TO!
- tijelo: "sama poruka"
  - isključivo ASCII znakovi



MIME (Multipurpose Internet Mail Extensions)

- kako poslati poruku koja nije 7-bitni ASCII tekst?
  - kodirati druge oblike u 7-bitni ASCII -> MIME
- funkcija sloja prikaza ISO/OSI modela
- MIME se sastoji od tri osnovna dijela:
  - linije zaglavlja koje nadograđuju polazni RFC 822
    - npr. MIME-Version, Content-Description, Content-Type, Content-Transfer-Encoding
  - skup tipova i podtipova sadržaja
    - npr. image/gif, image/jpeg, text/plain, text/richtext, application/postscript, application/msword, multipart
  - način enkodiranja različitih tipova podataka da se mogu prenositi e-poštom u ASCII obliku
- ASCII i MIME su u jasnom obliku; što ako želimo sadržaj poruke učiniti tajnim? -> (Mreže računala 2)

Protokoli za dohvat pošte (mail access protocols)



- SMTP: isporuka/pohrana na poslužitelj primatelja
- protokol za dohvat pošte: dohvaćanje pošte s poslužitelja
  - POP: Post Office Protocol [RFC 1939]: autorizacija, skidanje
  - IMAP: Internet Mail Access Protocol [RFC 1730]: više mogućnosti, uključujući upravljanje porukama koje su pohranjene na poslužitelju
  - HTTP: gmail, Hotmail, Yahoo! Mail, itd.

Protokol POP3

faza autorizacije

- naredbe klijenta:
  - **`user:`** izjaviti korisničko ime
  - **`pass:`** lozinka
- odgovori poslužitelja
  - **`+OK`**
  - **`-ERR`**

faza transakcije, klijent:

- **`list:`** lista brojeva poruka
- **`retr:`** dohvaćanje poruke prema broju
- **`dele:`** brisanje
- **`quit:`** kraj

```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

POP3 (…natavak) i IMAP

još o POP3

- prethodni primjer koristi POP3 način “skidanja i brisanja”
  - Bob ne može ponovo čitati isto e-pismo, ako je promijenio klijenta
- POP3 “skini-i-ostavi”: kopije poruka na različitim klijentima
- POP3 ne čuva stanja između sjednica

IMAP

- drži sve poruke na jednom mjestu: na poslužitelju
- korisnici mogu organizirati poruke u mape
- pamti stanje korisnika između sjednica:
  - imena mapa i smještaj poruka (prema ID-u poruke) unutar mapa

Poglavlje 2: sadržaj

2.1 principi na kojima se zasnivaju mrežne aplikacije

2.2 Web i HTTP

2.4 elektronička pošta

- SMTP, POP3, IMAP

2.3* Prijenos datoteka

2.5 DNS

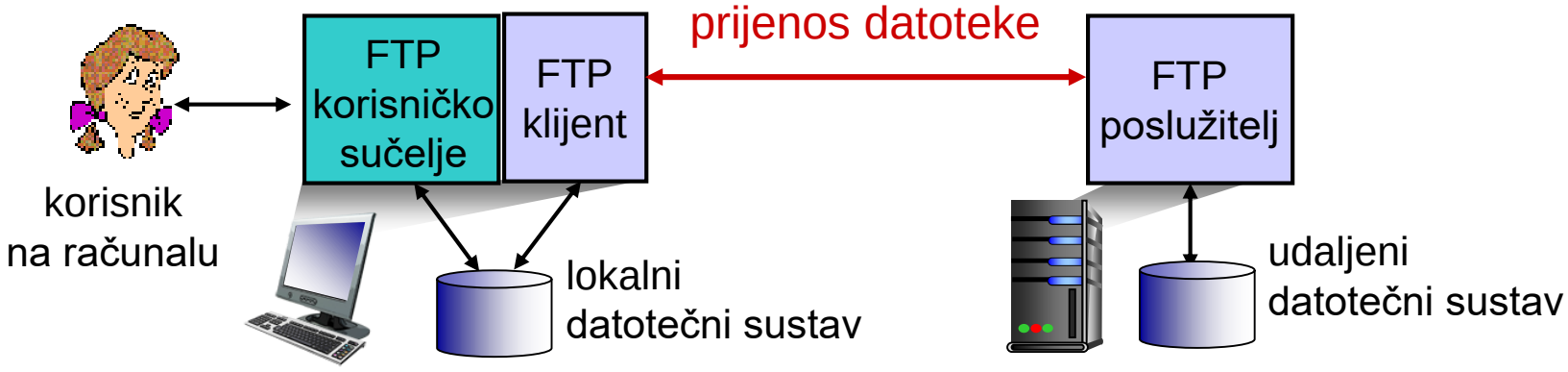
2.6 P2P aplikacije

2.7 socket programiranje s UDP-om i TCP-om

Prijenos datoteka

- Web i e-pošta: prijenos za specifične namjene
  - dohvaćane (*download*) datoteke s web-poslužitelja
  - slanje (*upload*) datoteke npr. metodom POST
  - datoteke kao dodaci u e-pošti
- Prijenos datoteka i direktorija na udaljeni datotečni sustav: FTP, SFTP, FTPS,…
- Sinkronizacija datoteka i mapa: *DropBox*, *pCloud*, *Sync.com*,…

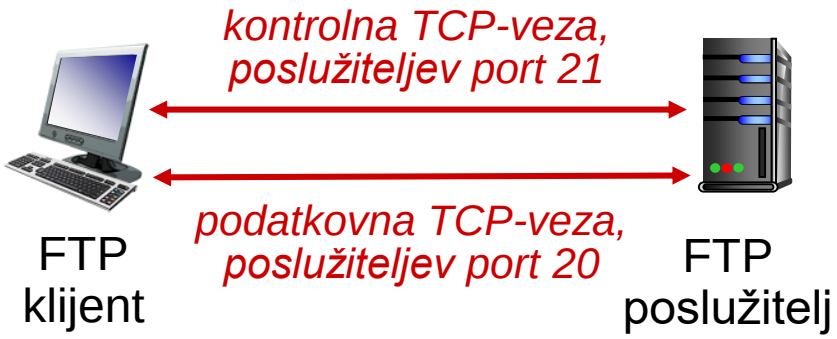
FTP: *file transfer protocol* (RFC 959)



- prijenos datoteke od/do udaljenog računala
- model klijent/poslužitelj
  - *klijent:* strana koja inicira prijenos (od/do udaljenog rač.)
  - *poslužitelj:* udaljeno računalo (tcp port 21)
- FTPS prijenos poruka preko TLS-a koji koristi TCP
  - ftps ostvarena sigurnost
  - običan FTP koristi TCP izravno

FTP: odvojene veze za upravljanje i podatke

- FTP klijent kontaktira FTP poslužitelj na TCP portu 21
- autorizacija klijenta preko veze za upravljanje
- klijent pregledava udaljeni direktorij i šalje naredbe preko kontrolne veze
- kada poslužitelj primi naredbu za prijenos datoteke, *poslužitelj* otvara drugu TCP-vezu za podatke (za datoteku) *prema klijentu*
- nakon prijenosa datoteke, poslužitelj zatvara vezu za slanje podataka



- poslužitelj otvara još jednu TCP-vezu za prijenos još jedne datoteke
- kontrolna veza: *"izvan osnovnog pojasa (out of band)"*
- FTP poslužitelj održava "stanje": tekući direktorij, prethodna autentifikacija

FTP: naredbe, odgovori

primjeri naredbi:

- šalju se kao ASCII tekst kroz kontrolni kanal
- **USER *username***
- **PASS *password***
- **LIST** vraća listu datoteka u tekućem direktoriju
- **RETR filename** dohvaćanje datoteke
- **STOR filename** sprema datoteku na udaljeno računalo

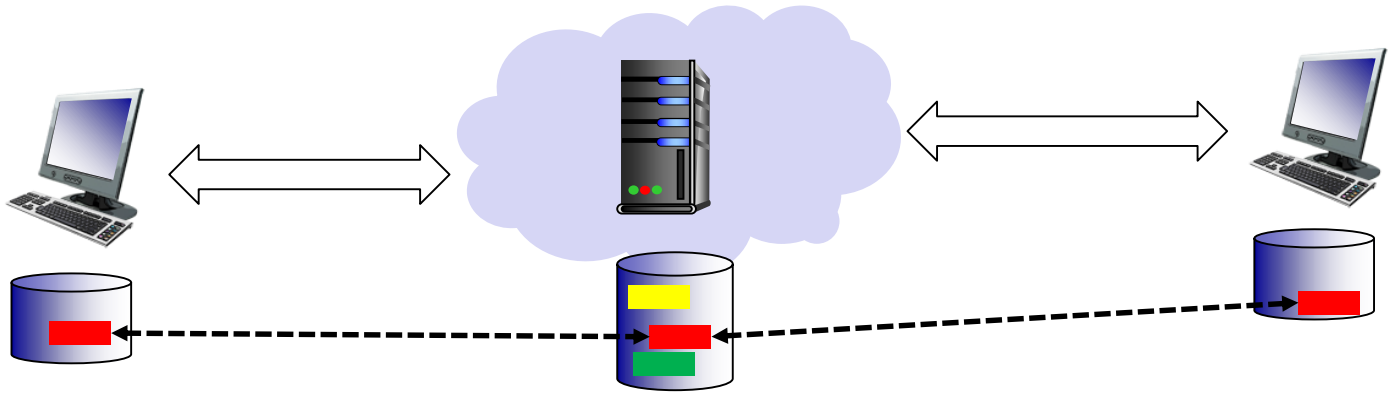
primjeri povratnih kôdova

- Statusni kod i fraza (kao i u HTTP-u)
- **331 Username OK, password required**
- **125 data connection already open; transfer starting**
- **425 Can't open data connection**
- **452 Error writing file**

SFTP: *SSH File Transfer Protocol*

- slično kao i FTP:
  - prijenos datoteke od/do udaljenog računala
  - upravljanje datotekama i direktorijima na udaljenom datotečnom sustavu
  - arhitektura klijent-poslužitelj
  - klijenti sa slikovnim ili tekstovnim sučeljem
- koristi sigurnost SSH, tcp port 22
- neke naredbe:
  - `ls` – sadržaj tekućeg direktorija na udaljenom računalu (n.u.r)
  - `put` – kopira datoteke s lokalnog na udaljeno računalo
  - `get` – kopira datoteke s udaljenog na lokalno računalo
  - `exit` ili `quit` – naredba za raskidanje veze
  - `mkdir` – stvara direktorij n.u.r
  - `cd` – promjena tekućeg direktorija n.u.r
  - `chmod` – promjena prava pristupa n.u.r
  - `rm` / `rmdir` – brisanje datoteke / direktorija n.u.r
  - `lls`, `lcd`, `lmkdir` kao `ls`, `cd` i `mkdir` samo na lokalnom računalu

Sinkronizacija datoteka i mapa



pohrana datoteka često u oblaku

- *DropBox*, *pCloud*, *Sync.com, …*
- prevladavaju vlasnički aplikacijski protokoli
- za transport TLS i TCP
- arhitektura klijent-poslužitelj
- *LAN Sync* – P2P prijenos unutar LAN-a (*DropBox*, *pCloud*)
  - *Dropbox LanSync Protocol*: tcp port 17500
  - *Dropbox LanSync Discovery*: udp port 17500

Sinkronizacija datoteka i mapa

- *DropBox*
  - automatski se sinkronizira mapa
  - promjena datoteke – dovoljno slati samo dijelove (jedan komad 4MB → sha-256)
  - TLS: zaštita od prisluškivanja komunikacijskih poveznica, ali kompanija (i drugi) vide sadržaj!
- *pCloud*
  - virtualni disk smješten u oblaku
  - mogu se sinkronizirati i mape
  - TLS: zaštita od prisluškivanja kom. poveznica
  - *Crypto* – dodatak: nazivi i sadržaj datoteka se kriptiraju na klijentu prije slanja poslužitelju

Poglavlje 2: sadržaj

2.1 principi na kojima se zasnivaju mrežne aplikacije

2.2 Web i HTTP

2.4 elektronička pošta

- SMTP, POP3, IMAP

2.3* Prijenos datoteka

2.5 DNS

2.6 P2P aplikacije

2.7 socket programiranje s UDP-om i TCP-om

DNS: domain name system

ljudi: mnogo identifikatora:

- OIB, JMBG, ime, broj putovnice,…

Internet: računala i usmjernici:

- adresa IP-a za adresiranje datagrama
- "ime", npr. www.yahoo.com → koriste ga ljudi

P: kako preslikati ime u adresu IP-a i obrnuto?

Domain Name System:

- *distribuirana baza podataka* implementirana hijerarhijski pomoću mnogo *imenskih poslužitelja (name servers)*
- *protokol aplikacijske razine:* računala i imenski poslužitelji komuniciraju kako bi *razriješili (resolve)* imena (preveli imena u adrese)
  - napomena: funkcija jezgre Interneta, ostvarena kao protokola aplikacijskog sloja
  - složenost "na rubu" mreže

DNS: usluge i struktura

usluge DNS-a

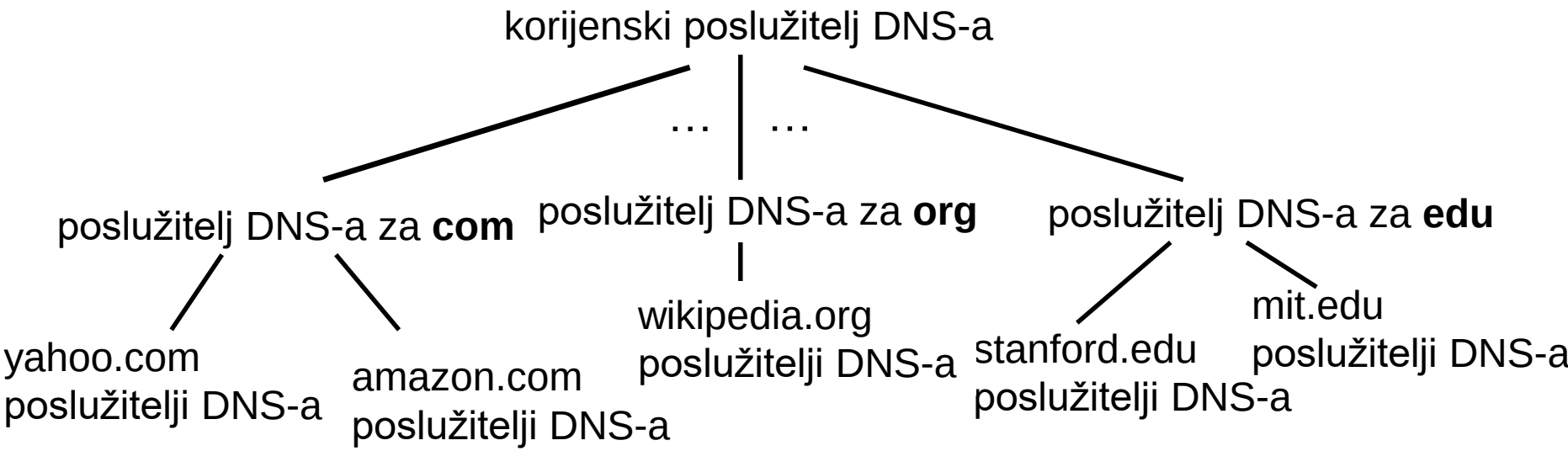
- prevođenje imena u adrese IP-a
- alternativna imena/ nadimci/alias računala
  - kanonska i alias imena
- alias poslužitelja pošte
- raspodjela opterećenja
  - replicirani poslužitelji weba: mnogo adresa IP protokola odgovara jednom imenu

zašto ne centralizirani DNS?

- jedna točka kvara
- količina prometa
- udaljenost do centralne baze
- održavanje

O: ne skalira se dobro!

DNS: distribuirana, hijerarhijska baza podataka

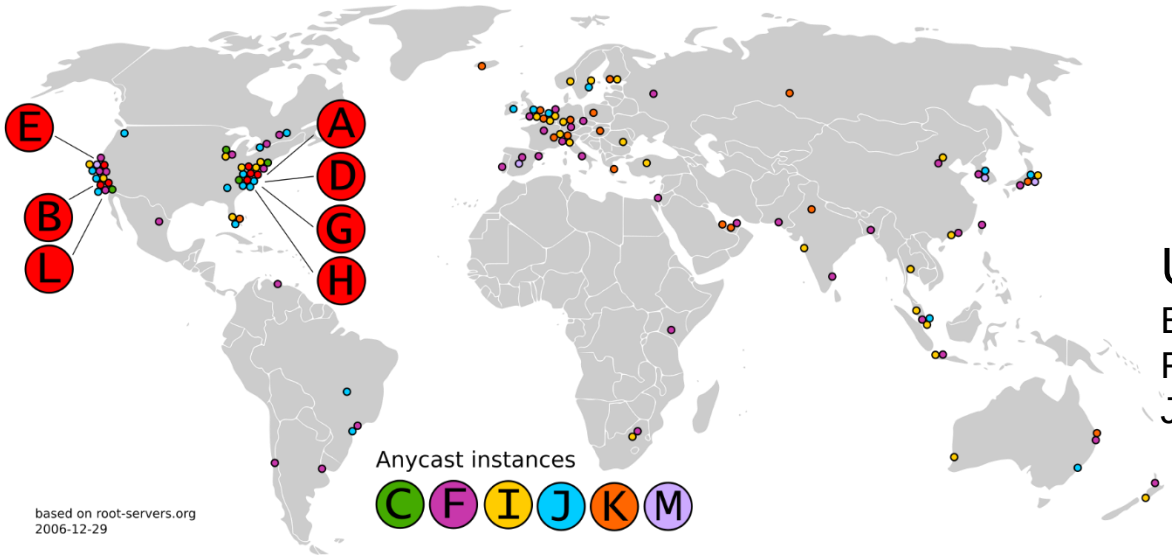


klijent želi adresu IP-a za www.amazon.com; 1. aproksimacija:

- klijent pita korijenski poslužitelj da nađe poslužitelja DNS-a za com
- klijent pita poslužitelja DNS-a za .com da pronađe poslužitelj za amazon.com
- klijent pita poslužitelja DNS-a za amazon.com da pronađe adresu IP-a za www.amazon.com

DNS: korijenski imenski poslužitelji

- kontaktira ih lokalni imenski poslužitelj koji ne može razriješiti ime
- Iznimno važan za funkcioniranje Interneta:
  - Internet ne bi radio bez njih!
  - DNSSEC – pruža sigurnost (autentifikacija i integritet poruke)
- ICANN (*Internet Corporation for Assigned Names and Numbers*) upravlja korijenskom DNS domenom



13 korijenskih imenskih „poslužitelja" u svijetu – svaki od njih repliciran mnogo puta

U HRVATSKOJ (2019)

E – ZG (*NASA Ames Research Center*)
F – ZG (*Internet Systems Consortium, Inc.*)
J – ZG (*Verisign, Inc.*)

TLD, autoritativni poslužitelji

poslužitelji vršne domene (top-level domain, TLD):

- odgovorni za com, org, net, edu, aero, jobs, museums i sve vršne domene zemalja, npr.: hr, uk, fr, de, ca, jp
- Network Solutions održava poslužitelje za .com TLD
- CARNet upravlja .hr TLD

autoritativni poslužitelji DNS-a:

- poslužitelji DNS-a koji pripadaju organizaciji
  - pruža autoritativno preslikavanje imena i adresa računala svoje organizacije
- može ga održavati organizacija ili njezin pružatelj usluga

Lokalni imenski poslužitelj

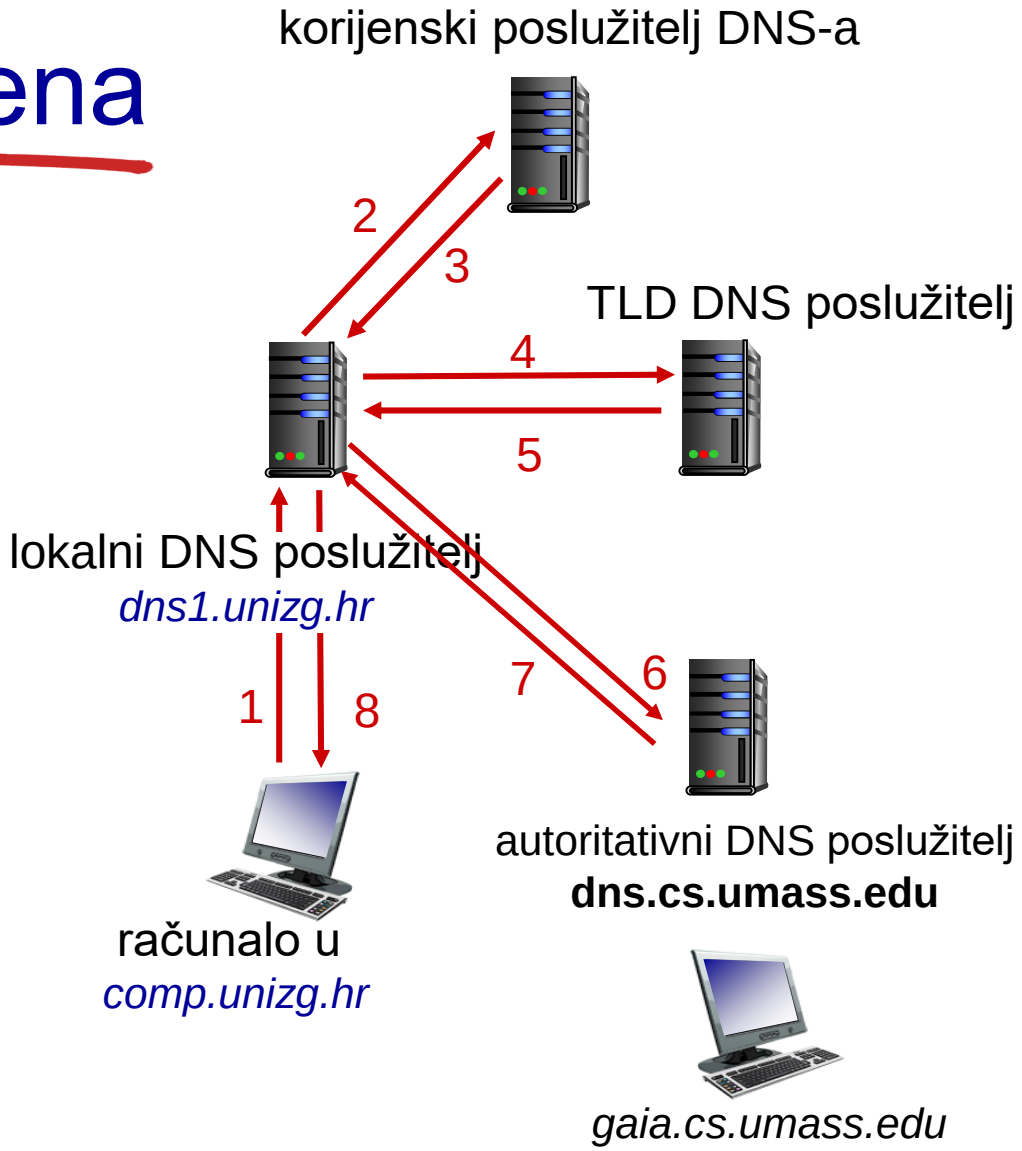
- striktno gledajući ne spada u hijerarhiju
- svaki ISP (rezidencijalni ISP, kompanija, sveučilište) ih ima
  - poznat i kao "*default name server*"
- kada računalo napravi DNS upit, taj upit šalje svom lokalnom imenskom poslužitelju
  - sadrži lokalni *cache* od nedavno korištenih parova preslikavanja ime-u-adresu (ali mogu biti zastarjeli!)
  - ponaša se kao proxy, prosljeđuje upite u hijerarhiju

DNS: primjer prevođenja imena

- računalo u poddomeni comp.unizg.hr želi saznati adresu IP-a od računala gaia.cs.umass.edu

iterativni upit:

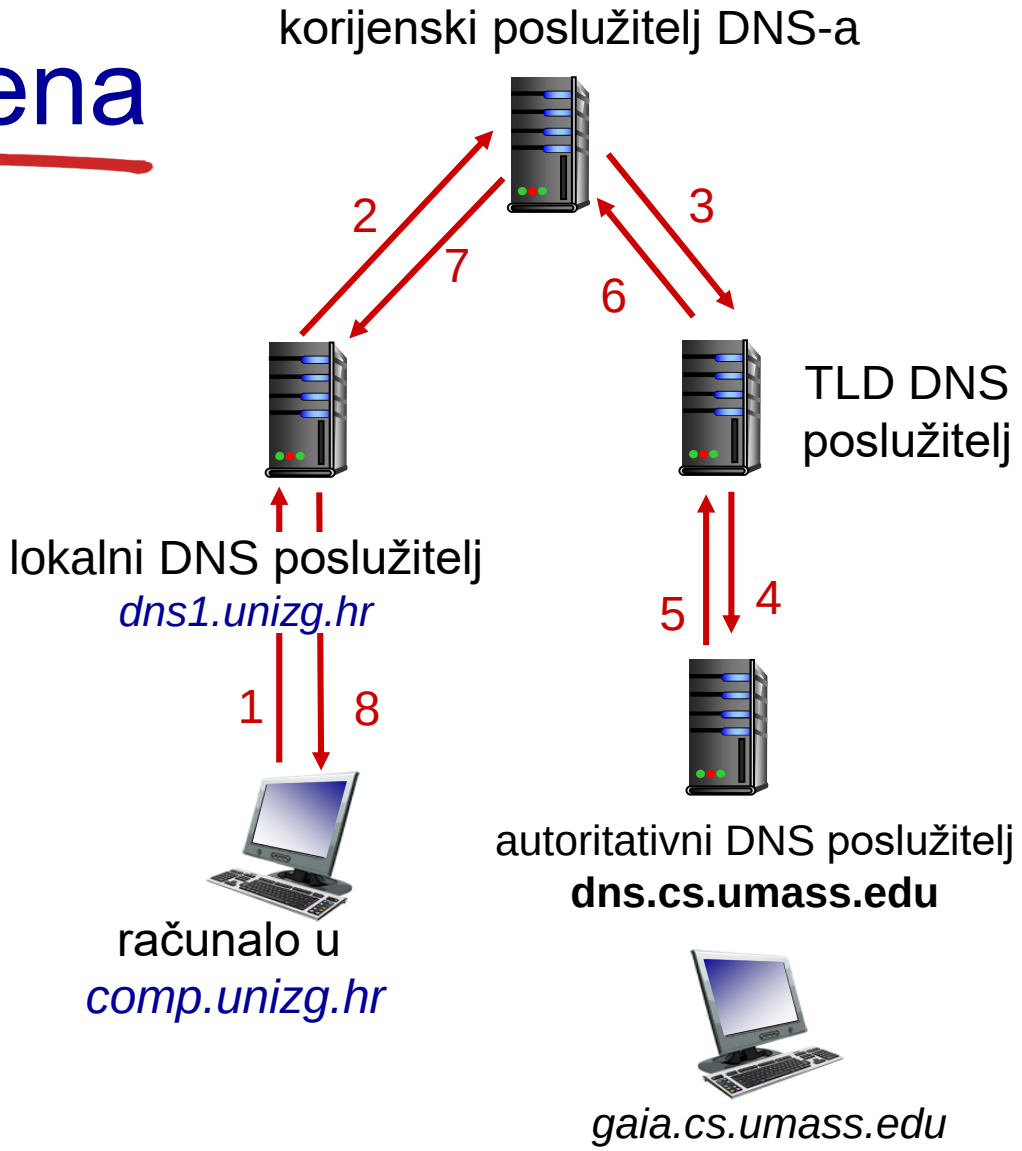
- poslužitelj odgovara s imenom sljedećeg poslužitelja kojega treba kontaktirati
- "ne znam to ime, ali pitaj ovaj poslužitelj"



DNS: primjer prevođenja imena

rekurzivan upit:

- teret prevođenja imena je na poslužitelju kojem je poslan upit
- teže opterećenje za više razine hijerarhije?



DNS: *caching* i ažuriranje zapisa

- jednom kada poslužitelj nauči preslikavanje, to preslikavane sprema u *cache*
  - zapisi u cacheu imaju vremensko trajanje (TTL), nakon čega se brišu
  - TLD poslužitelji su obično u *cacheu l*okalnog poslužitelja
    - pa se korijenskim poslužiteljima rijetko postavljaju upiti
- zapisi u *cacheu* mogu *zastariti* (prevođenje ime-u-adresu na principu najbolje što može!)
  - ako računalo promijeni adresu IP-a, to ne mora biti poznato na Internetu dok ne isteknu TTL-ovi
- ažuriraj/obavijesti mehanizme predložio IETF
  - RFC 2136

DNS: zapisi

DNS: distribuirana baza podataka pohranjuje zapise (resource records, RR)

RR format: **`(ime, vrijednost , tip, ttl)`**

tip=A

- **`ime`** je ime rač. (hostname)
- **`vrijednost`** je IP adresa

tip=NS

- **`ime`** je domena (npr., foo.com)
- **`vrijednost`** je *hostname* autoritativnog imenskog poslužitelja za tu domenu

tip=CNAME

- **`ime`** je alias nekog "kanonskog" imena
- **`www.ibm.com`** je zapravo alias za **`servereast.backup2.ibm.com`**
- **`vrijednost`** je kanonsko ime

tip=MX

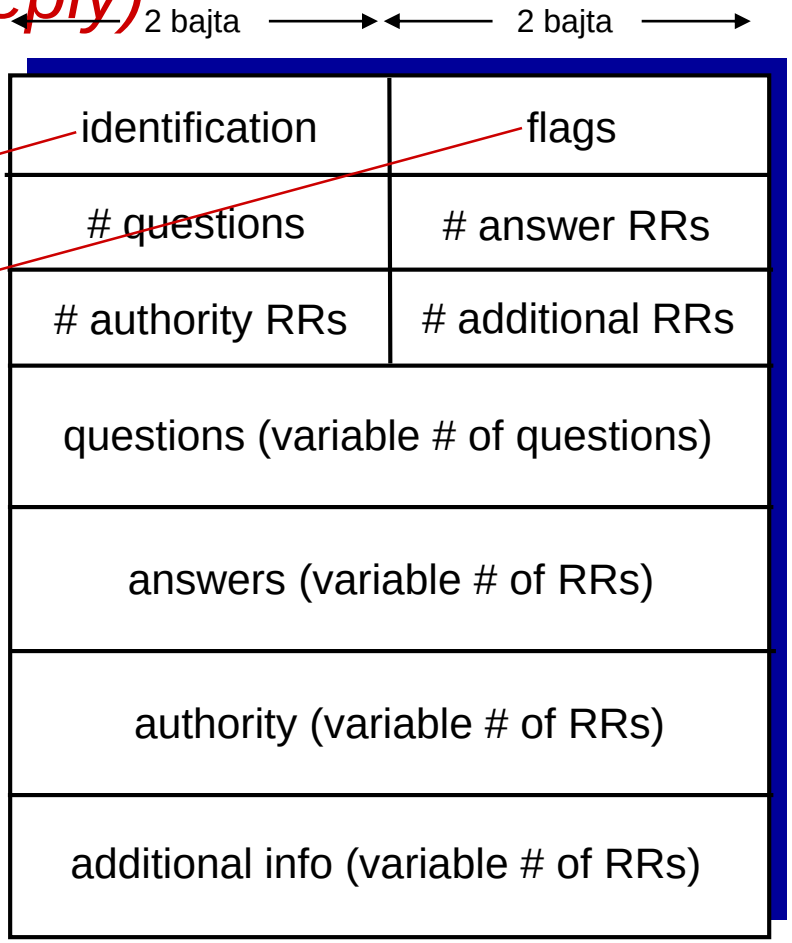
- **`vrijednost`** je ime poslužitelja pošte vezanog za **`ime`**

Poruke protokola DNS

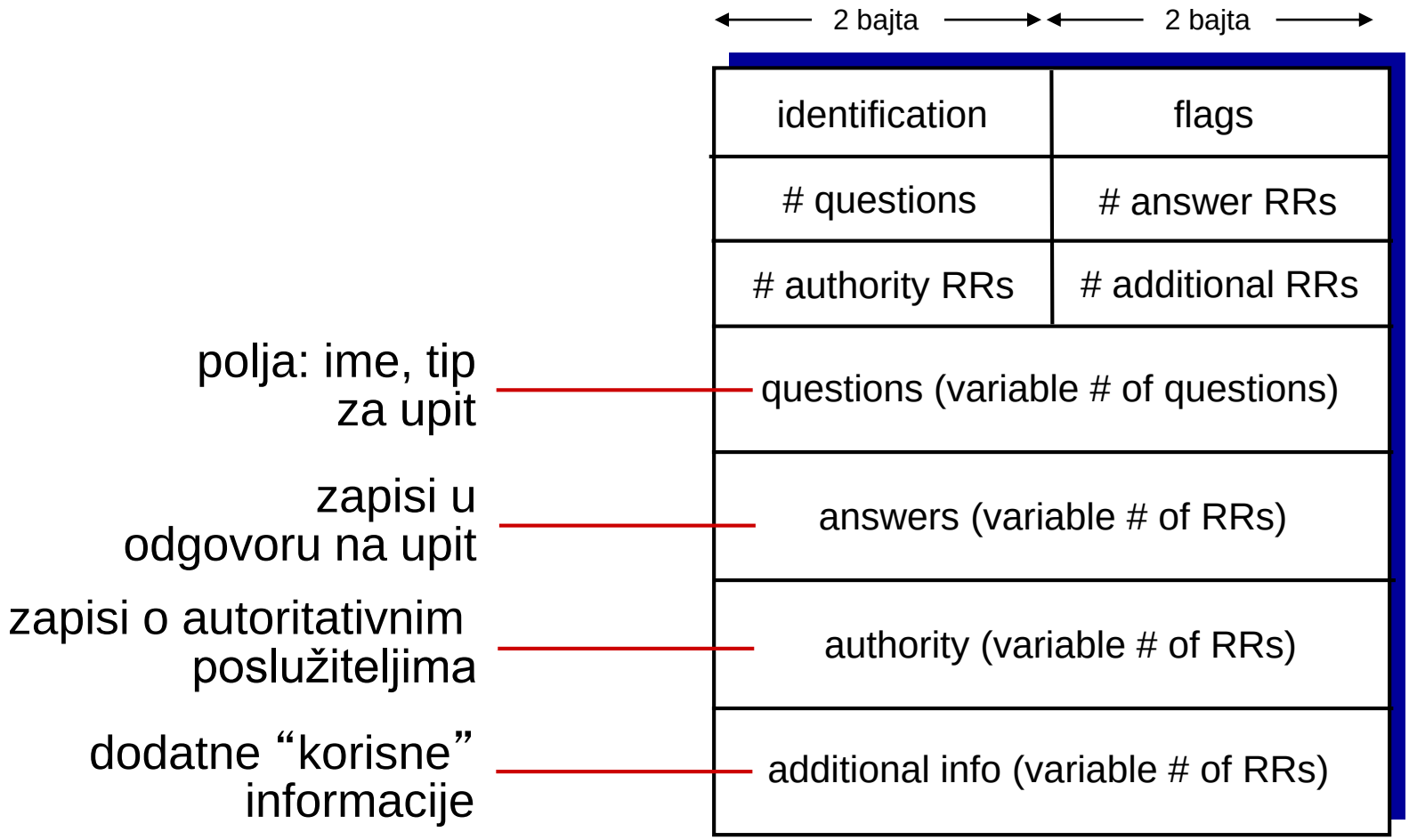
- isti *format poruke* za poruke *upita (query)* i *odgovora (reply)*

zaglavlje poruke

- identification: 16-bitni broj upita, odgovor na njega koristi isti broj
- zastavice:
  - upit ili odgovor
  - zahtjeva se rekurzija
  - rekurzija moguća
  - odgovor je autoritativan



Poruke protokola DNS



Dodavanje zapisa u DNS

- primjer: nova organizacija “Network Utopia”
- registrira ime networkuptopia.com kod *DNS registratora* (npr. Network Solutions)
  - dostavlja ime, IP-adrese od autoritativnog imenskog poslužitelja (primarnog i sekundarnog)
  - registrator dodaje dva zapisa u .com TLD poslužitelj:
    (networkutopia.com, dns1.networkutopia.com, NS)
    (dns1.networkutopia.com, 212.212.212.1, A)
- u autoritativni imenski poslužitelj dodati
  - zapis tipa A za www.networkuptopia.com
  - zapis tipa MX za networkutopia.com

Napadi na DNS

DDoS napadi

- Bombardiranje korijenskog poslužitelja s prometom
  - do sada nije uspjelo
  - filtriranje prometa
  - lokalni DNS poslužitelji stavljaju u cache IP adrese TLD poslužitelja, što omogućuje zaobilaženje korijenskog poslužitelja
- Bombardiranje TLD poslužitelja
  - potencijalno opasnije

Napadi preusmjeravanja

- čovjek-u-sredini
  - presretanje upita
- DNS trovanje
  - slanje lažnih odgovora DNS poslužitelju, koji se spremaju u *cache*

Zloupotreba DNS-a za DDoS

- slanje upita s lažnim izvorišnim IP adresama: postaje odredišna IP adr.
- zahtjeva pojačavanje

Poglavlje 2: sadržaj

2.1 principi na kojima se zasnivaju mrežne aplikacije

2.2 Web i HTTP

2.3 FTP

2.4 elektronička pošta

- SMTP, POP3, IMAP

2.5 DNS

2.6 P2P aplikacije

2.7 socket programiranje s UDP-om i TCP-om

Poglavlje 2: sažetak

- arhitekture aplikacija
  - klijent-poslužitelj
  - ravnopravni-čvorovi (P2P)
- zahtjevi za uslugama :
  - pouzdanost, propusnost, kašnjenje
- odabir Internet transportnog modela usluga
  - orijentiran-na-vezu, pouzdan: TCP
  - nepouzdan, datagrami: UDP
- pojedini protokoli:
  - HTTP
  - FTP
  - SMTP, POP, IMAP
  - DNS
  - P2P: BitTorrent, DHT
- socket programiranje: TCP, UDP socketi

Poglavlje 2: sažetak

- tipična razmjena poruka zahtjev/odgovor :
  - klijent zahtjeva podatke ili uslugu
  - poslužitelj odgovara s podacima i statusnim kôdom
- formati poruka

važne teme:

- kontrolne i podatkovne poruka
  - in-band, out-of-band
- centraliziran ili decentraliziran
- “sa stanjem” ili “bez stanja“
- nepouzdan ili pouzdanom prijenos poruka
- “složenost na rubu mreže”